Journal of Artificial Intelligence Research 70 (2021) 955-1001 Submitted 06/2020; published 03/2021

# Constrained Multiagent Markov Decision Processes: a Taxonomy of Problems and Algorithms

**Frits de Nijs** FRITS.NIJS@MONASH.EDU
*Dept. of Data Science and AI, Faculty of IT, Monash University*
*20 Exhibition Walk, 3168 Clayton, Australia*

**Erwin Walraven** E.M.P.WALRAVEN@TUDELFT.NL
**Mathijs M. de Weerdt** M.M.DEWEERDT@TUDELFT.NL
**Matthijs T. J. Spaan** M.T.J.SPAAN@TUDELFT.NL
*Delft University of Technology*
*Van Mourik Broekmanweg 6, 2628 XE Delft, The Netherlands*

## Abstract

In domains such as electric vehicle charging, smart distribution grids and autonomous warehouses, multiple agents share the same resources. When planning the use of these resources, agents need to deal with the uncertainty in these domains. Although several models and algorithms for such constrained multiagent planning problems under uncertainty have been proposed in the literature, it remains unclear when which algorithm can be applied. In this survey we conceptualize these domains and establish a generic problem class based on Markov decision processes. We identify and compare the conditions under which algorithms from the planning literature for problems in this class can be applied: whether constraints are soft or hard, whether agents are continuously connected, whether the domain is fully observable, whether a constraint is momentarily (instantaneous) or on a budget, and whether the constraint is on a single resource or on multiple. Further we discuss the advantages and disadvantages of these algorithms. We conclude by identifying open problems that are directly related to the conceptualized domains, as well as in adjacent research areas.

## 1. Introduction

The recent arrival of Artificial Intelligence (AI) as a household term comes from the success of well-known examples like digital personal assistants (Hoy, 2018), media recommender systems (Möller et al., 2018), and game-playing AI capable of surpassing human experts (Silver et al., 2018). However, these examples have in common that they interact one-on-one with the user. We expect that increased uptake of AI systems will quickly also lead to more *multiagent* systems, in which multiple intelligent agents interact while making decisions. In such systems the agents are likely to encounter constraints due to the presence of shared resources, which limits their potential decisions.

Multiagent systems with resource constraints arise naturally in several parts of our society nowadays. For example, in autonomous warehouses there are multiple robots which collect items for delivery, and these robots are constrained by, e.g., time constraints, locations, and facilities they use (Claes et al., 2017). Another example can be found in power distribution grids, in which aggregators control charging of multiple autonomous electric vehicles which are not allowed to violate distribution grid constraints (De Weerdt et al.,

2018). Multiagent decision making with constraints also occurs in online advertising, in which a limited advertising budget should be assigned to online users in order to maximize conversion (Boutilier & Lu, 2016). Finally, monitoring tasks at airports need to be performed with limited security staff while considering multiple autonomous adversaries (Pita et al., 2008). Besides the existence of *i)* more-or-less independent agents and *ii)* resource constraints, all aforementioned applications include *iii)* sources of uncertainty which need to be considered while making decisions where these decisions may affect each other sequentially, and *iv)* there is an opportunity for coordination in advance. These four properties identify the type of problems under discussion in this article.

Markov decision processes (MDPs) provide a well-studied framework for decision-making problems with uncertainties (Puterman, 1994; Boutilier, 1996). Over the years many extensions have been proposed which augment the framework with additional constraints on the optimal policy. Altman (1999) gives a comprehensive overview of the properties of constrained Markov decision processes with discounted or expected average costs. In this paper we survey work that builds on these models by taking a multiagent perspective, emphasizing the coordination aspect between agents regarding resources. Additionally, we include work that defines constraints that must hold absolutely, under any circumstance. These different definitions of constraints have mostly been studied in isolation, which makes it unclear how the algorithms proposed in literature compare against each other, and under what circumstances they can be used when solving planning problems.

In this survey we aim to establish a general understanding of problems that can be modeled as constrained multiagent Markov decision processes and the associated algorithms that have been proposed in the literature. A first aim is to provide further understanding which (type of) algorithm works best for which type of domains. Moreover, the overview enables us to identify unexplored areas in the field as well as promising research directions that can be pursued in future research.

### 1.1 Contributions

We present a general conceptualization of constrained multiagent planning problems based on Markov decision processes. This conceptualization describes all aspects of resources and planning domains that need to be considered when performing planning in environments with shared constraints. This leads to taxonomy in which all relevant techniques developed in the literature can be placed.

Second, we give an extensive overview of solution algorithms that appeared in the planning literature, and we identify and compare the conditions under which these algorithms can be applied. We further discuss the advantages and disadvantages of these algorithms.

Third, we identify open research directions that can be pursued to advance the research field in the future. This discussion identifies open problems that currently exist, as well as research areas that are closely related. These areas have strong connections with the research area covered by this survey, and both areas may strengthen each other in the future.

From a more general perspective, our survey also aims to provide a comprehensive overview for new researchers in the field, such that they can quickly grasp the main concepts and solutions that have been developed in the literature. This enables them to understand

the constrained planning field as it is right now, and it ensures that they can quickly identify open problems and research directions to work on.

### 1.2 Outline

The structure of this survey is as follows. In Section 2 we start from motivating application domains involving multiple agents, uncertainty and constraints, analyze the properties of these domains, and introduce general potential solution approaches which enable agents to coordinate their actions subject to the availability of resources informally. In Section 3 we introduce formal models that capture the identified domain properties. The potential solution approaches provide the starting point for Section 4, in which we describe several algorithms that can be used to compute or implement solutions in multiagent systems with resource constraints. In Section 5 we describe related problem variants and algorithms that are outside the main focus of this survey. In Section 6 we describe open research directions and important observations regarding the general field of constrained planning. Finally, we summarize and conclude the survey in Section 7.

## 2. Characteristics of Constrained Multiagent Markov Decision Problems

In this section, we first introduce motivating examples of problems that can be modeled as constrained multiagent Markov decision processes (CMMDPs). Using these examples, we discuss which properties these problems have in common, but also which properties vary over the problems. Depending on the specific properties a problem has, different solution approaches are applicable.

### 2.1 Motivating Application Domains

We find CMMDPs in several prominent domains including energy, advertising and maintenance. First we give some example domains with autonomous decision makers that need to coordinate the use of a shared resource or infrastructure. After that we introduce two example domains where there is a single, centralized decision-maker that reasons about all the agents in the domain and their resource allocation. Both categories of domains motivate the need for sophisticated algorithms to solve these problems.

#### 2.1.1 Autonomous Agents Coordinating their Resource Usage

In order to avert runaway climate change, there is now significant momentum towards using renewable sources like the sun and wind to generate electricity, and simultaneously to move as many energy-consuming activities as possible to use electricity as a power source. Notable examples of this trend include electric vehicles (EVs) as replacement for combustion engine vehicles, and electric heat pumps as replacement for gas-based house central heating systems.

However, this energy transition causes several practical problems, by placing additional demands on the electricity generation and transportation infrastructure. Firstly, the introduction of new high-power loads like EVs can cause the daily peak demand to exceed the transmission capacity of the local distribution network. Secondly, supply from renewable sources depends on the weather conditions, which cannot be predicted perfectly. However,

supply and demand of electricity must be balanced at all times to maintain grid stability, which requires that fluctuations in wind speed or cloud cover must be compensated.

Both these problems could be alleviated or reduced by *smart grid control*, for example by using flexibility in the demand for electricity. Several types of (potentially) flexible load have been identified: EVs typically need shorter to charge than the total time they spend parked, and can thus have their charge rate modulated or time periods shifted (De Weerdt et al., 2018). Heat pumps and air conditioners may also be shifted in time, because the indoor temperature changes gradually due to thermal inertia. Because people experience a range of temperatures as comfortable, there is typically some time before the temperature deviates too far from its setpoint (see, e.g., De Nijs, Spaan, & De Weerdt, 2015).

However, because using consumer appliances for demand-side flexibility affects the comfort of the user, its usage should be optimized to minimize its impact. Optimizing the control decisions for all appliances in a neighborhood of devices subject to total power consumption constraints, and/or to network constraints are important examples of problems that can be modeled as CMMDPs, with slightly different properties, such as whether there is sufficient time and communication reliability to coordinate online, and whether these limitations are strict or some (short) violation of these constraints is allowed.

Moreover, controllers for flexible loads need to reason about several sources of uncertainty. In the first place, the power supply itself is subject to uncertainty, or in the case of dealing with network constraints, the available amount may be uncertain due to other loads. Secondly, in the case of charging, the arrival and departure times of EVs are uncertain. Finally, simplified models of thermal inertia may not capture all aspects of the actual temperature transition, in particular due to the behavior of people in a building.

Similar examples can be found in other situations where a common resources or infrastructure is shared with multiple actors, such as barges going through locks, trains from different companies using a shunting yard, car owners finding a parking place, search and rescue parties coordinating use of resources, and the en-route charging of EVs (De Weerdt et al., 2016). In some domains, the decisions to be supported are not directly made by the agents in the system, but the agents are influenced by a centralized decision maker.

#### 2.1.2 Coordination of Agents by a Centralized Decision-Maker

*Online advertising* involves presenting advertisements to online users which browse the internet (Boutilier & Lu, 2016), aiming to seduce these users to use the advertised good or service (called conversion). In the related planning problem, these users are modeled by agents and the decisions on which agents to serve ads to is (stochastically) based on their potential for conversion, subject to the advertising budget of the advertiser.

As such, each agent has states corresponding to its level of interest in the advertised product, ranging from uninterested, to searching, to interest in either the advertiser's or the competitor's product, all with various levels of intensity. The level of interest of real online users cannot be measured accurately, and therefore the state of the system is partially observable from the viewpoint of the advertiser. Global reward is obtained when the agent moves to the conversion state of the advertiser's product. The advertiser influences the state transitions of the agent by selecting the intensity of the campaign directed at each browser. At the lowest level, no ads are shown and no costs are incurred. The next levels

use progressively more resources to increasingly influence the transition function in favor of reaching the advertiser's conversion state.

The advertiser has only a single resource, a budget for funding the advertising actions. Budget constraints mean that consumption in all time steps is counted towards a single resource constraint. In principle such a budget is a hard constraint, but the large-scale, high-speed nature of online advertising means that it may be more practical for an advertisement aggregator to aim to meet their clients' budget constraints in expectation, in which case the constraint can be seen as a soft resource constraint.

Since the advertiser models each individual user and optimizes the advertisements the system shows, there is no communication required between online users that browse the internet. Instead, the advertiser can decide in each time instant which advertisements are shown, after which these advertisements are sent to the online user.

Another example of centrally making decisions about the allocation of limited resources to agents is in the context of *condition-based maintenance*. Systems for which the condition and performance deteriorate over time require maintenance in order to prevent damage and failures. Performing maintenance on a regular basis in fixed intervals can be costly, and therefore condition-based maintenance has been proposed to reduce maintenance cost and operational cost in general (Jardine et al., 2006). Condition-based maintenance uses inspections and sensor diagnostics to decide about the maintenance to perform. Condition-based maintenance emerges in several areas, including wind turbine maintenance (Byon & Ding, 2010), bridges and infrastructure (Neves & Frangopol, 2005; Van den Boomen et al., 2020) and aircraft components (Harman, 2002). From a decision-making point of view, the complexity can be found in deciding when and how maintenance is performed.

In condition-based maintenance the main objective is keeping multiple objects in a good condition given a fixed maintenance budget. Each object that requires maintenance can be seen as an agent whose condition behaves stochastically over time, and inspections and maintenance can be performed on these objects to maintain its condition. The current condition of an object is partially observable, because sensor readings and diagnostics do not provide perfect information about the actual condition of the object. The actual condition becomes known after performing manual inspections, which can be costly or time-consuming in practice. The model that can be used for maintenance planning of an individual object consists of states that reflect the condition of the object, and actions that represent either inspection or maintenance actions. The costs associated with these actions represent monetary cost of inspections and maintenance.

Typically there is a fixed budget available for performing maintenance, which represents a resource constraint that needs to be considered while performing planning of maintenance. This constraint can be a global constraint that spans a long period, but the problem can also be formulated with multiple short-term budget constraints or constraints that affect only a subset of objects. Depending on the actual objects and the business, the constraints can be considered either hard or soft.

### 2.2 Analysis of Problem Domain Properties

While the 'decision maker' in the problems described in the previous sections may be either a centralized entity or a true collection of individual agents, both types of problems can be

approached with essentially the same solution methods. This is because the problems have three fundamental properties in common, which are the key elements of CMMDPs.

**Common Properties of CMMDPs** In the first place, each problem has multiple entities (agents) that are functionally *independent*, except for their *resource consumption*. Secondly, the decisions need to be made sequentially, while taking *uncertainty* over their consequences into account. Thirdly, in each domain, there is an opportunity to *coordinate in advance*, allowing to plan a course of action that takes all entities into account.

Apart from the aspects that are common across these domains, they also vary along problem domain properties that affect what types of solution approaches are useful. These properties form the core of the conceptualization of CMMDPs. We discuss them here in turn and indicate which of the motivating application domains have which properties, summarized in Table 1.

**Persistence of Communication** One approach to handle uncertainty is to make coordinated decisions when sufficient information is available. Coordinating the decision-making regarding resource usage during plan execution requires a central decision maker, or a persistent communication channel between the agents and sufficient time between decisions to allow for computing a recourse solution. Such coordination may be assumed in the domains of online advertising and condition-based maintenance.

However, not every domain can provide for persistent, reliable communication. The environment in which agents operate may impose restrictions on the communication capabilities of the agents. For example, during rescue missions in hostile environments agents may not be able to communicate with each other while deciding about actions that require shared resources. In the case of smart grid control, localized power outages can bring down communication infrastructure, which means that relying on always-available communication reduces system robustness.

When communication is persistently available or the domain allows for a centralized decision-maker, we say that a domain is *connected*. Otherwise, a domain is *disconnected*. In disconnected domains solutions may need to be more conservative with the allocation of resources, especially when the domain also has hard constraints.

**Partial Observability of the State** Whether the available information is accurate and complete is an important assumption. When not all relevant information about the current state is known, we say the domain is partially observable. In some domains, such as in smart grids with a reliable measurement and communication infrastructure this is reasonable, but in other domains, such as in condition-based maintenance and in the advertising domain information is incomplete and sensor readings can be noisy. Algorithms that explicitly model the uncertainty regarding the state and that this is only partially observable often perform better in these domains than algorithms that work based on the assumption that all information is correct and fully observable.

**Strictness of Constraints** The presence of uncertainty means that the outcomes of plans are themselves random variables. As such, a decision maker has to decide whether it optimizes its behavior for the expected return, or for avoiding worst-case situations. As the impact of a plan on constraints is also stochastic, the same dilemma appears for satisfying constraints in expectation or also in the worst case.

| | Connected | PO | Strictness | Timespan | | |
|---|---|---|---|---|---|---|
| | | | | Budget | Inst. | Multi |
| Smart grid control | * | * | hard/soft | | ✓ | ✓ |
| Online advertising | ✓ | ✓ | hard/soft | ✓ | | |
| Condition-based maintenance | ✓ | ✓ | hard/soft | ✓ | | |

Table 1: Motivating application domains and their properties: "Connected" is the property that agents can reliably communicate during execution, PO stands for whether the current state is only partially observable, "Hard" and "Soft" refer to whether constraints need to be met always or in expectation, and budget/inst./multi whether the constraint holds across the whole timespan, only a single instant, or that there are multiple constraints, respectively. A '*' indicates that it depends on the specifics of the use case.

While it may seem natural to assume that constraints must always be satisfied, there are many domains in which satisfying constraints in expectation is a better choice. For example, constraints may be used as a proxy for multiple objectives, by setting constraints in order to attain some minimum performance level for each secondary objective. In this case, it makes sense to treat all objectives in the same way. In some other domains, such as grid congestion management, it can be acceptable to exceed capacity constraints occasionally; briefly exceeding capacity limits may heat up equipment, which may degrade material life if it persists for a long time. However, if the situation normalizes quickly, no harm is done. In online advertising budget constraints need to be met, but since this budget is allocated for a certain period, and the process is likely repeated many times, meeting the budget constraint in expectation could be quite reasonable. Finally, in settings like maintenance, the probability of impacting the constraint (i.e., the chance of simultaneously finding multiple major defects) may be so low that planning for worst-case situations leads to extremely conservative plans. In this case, it may be better to employ plan repair to handle unlikely constraint violations.

When the domain requires that constraints are met in the worst case, we call such constraints *hard*. If constraints only need to be met in expectation, we instead call them *soft*. We will see later that soft constraints are significantly easier to plan for, even allowing for efficient optimal algorithms in specific cases. A few algorithms for soft constraints bridge the gap, by providing bounds on the risk that actual resource use exceeds the limits.

**Timespan of Constraints** The application domains presented as motivation in Section 2 highlight the importance of time in the definition of a constraint. For example, the maximum electrical power that can be supplied by renewable generators implies a constraint that applies to a specific instant in time. This stands in contrast to optimizing maintenance activities under a fixed budget, which defines a constraint that is evaluated over a long-run horizon. In the subsequent we will refer to these two types as *instantaneous* and *budget* constraints, respectively.

The timespan of constraints can shape algorithm design; while instantaneous constraints and budget constraints both impose a requirement on agents to coordinate their actions

(e.g., who draws power), budget constraints also require them to keep track of the budget remaining (e.g., remaining state of charge in the battery). In this sense, budget constraints are inherently more general than instantaneous constraints, since we could define an instantaneous constraint as a budget constraint that can only be impacted by actions in one particular time step. On the other hand, problems with instantaneous constraints typically contain *multiple* constraints (e.g., a maximum bandwidth that applies in all time steps), which introduces game-like strategic considerations around trading resources of unequal value from a particular agent's viewpoint.

In the next section we show how the problems may be mathematically formalized using the framework of sequential decision-making under uncertainty.

## 3. Modeling Constrained Multiagent Decision-Making

First we present multiagent Markov decision processes, a standard modeling framework for these types of problems. Then we introduce a model for resources and constraints on these resources.

### 3.1 Multiagent Decision-Making under Uncertainty

To model decision making under uncertainty, we employ the typical Markov Decision Process (Bellman, 1957; Puterman, 1994, MDP) framework. An MDP model specifies how a stochastic environment behaves as a decision maker interacts with it: every discrete time step $t$, the decision maker is asked to choose an action $a$ from the finite set of available actions $A$ on the basis of the current state $s$. This action then induces a stochastic state transition in the environment, resulting in a subsequent state $s'$ sampled according to transition function $T(s, a)$. By choosing $a$, the decision maker is rewarded with an instantaneous reward $R(s, a)$. A multiagent MDP (Boutilier, 1996, MMDP) generalizes this modeling framework with the notion of multiple cooperative actors, each with their own action sets, as defined in Definition 1.

**Definition 1** (MMDP)**.** *A finite-horizon multiagent Markov decision process $M$ is defined by the tuple $\langle \alpha, S, \{A_i\}_{i\in\alpha}, T, R, h\rangle$, containing*

- *a finite set of agents $\alpha$ identified by $i \in \{1, 2, \ldots, n\}$,*
- *a finite set of states $S$,*
- *a finite set of actions $A_i$ for each agent $i$,*
- *a joint transition function $T : S \times A_1 \times \ldots \times A_n \times S \to [0, 1]$, inducing the probability mass function $T(s, a_1, \ldots, a_n, s') = \mathrm{P}(s' \mid s, a_1, \ldots, a_n)$,*
- *a joint reward function $R : S \times A_1 \times \ldots \times A_n \to \mathbb{R}$, and*
- *a finite time horizon $h$ of discrete time steps $t \in \{1, 2, \ldots, h\}$.*

Decision making in an MMDP allows for either a centralized or a decentralized perspective: in the centralized view, a single decision maker prescribes the actions for all the actors to take, while in the decentralized view there is one decision maker for each agent. These

two perspectives match the two main application domain categories of either autonomous decision makers that need to coordinate, or a single centralized decision maker (as identified in Section 2.1). In either case, rational decision maker(s) will plan to select actions that maximize the *expected* cumulative reward of their choices.

A solution to an (M)MDP takes the form of a policy $\pi : \{1, 2, \ldots, h\} \times S \to A$, with $A$ representing the set of *joint* actions in the multiagent case: $A = \times_{i \in \alpha} A_i$. The expected value of a specific policy $\pi$ is given by the value function $V_\pi$, defined recursively through the Bellman (1957) equation:

$$V_\pi(t, s) = \begin{cases} R(s, \pi(t, s)) + \sum_{s' \in S} T(s, \pi(t, s), s') \cdot V_\pi(t + 1, s') & 1 \leq t \leq h, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

An optimal policy $\pi^*$ selects an action that maximizes expected value, meaning

$$\pi^*(t, s) = \arg\max_{a \in A} \left[ R(s, a) + \sum_{s' \in S} T(s, a, s') \cdot V_{\pi^*}(t + 1, s') \right]. \tag{2}$$

The recursive structure of the optimal policy $\pi^*$ allows for a dynamic programming algorithm: first compute and memoize the values $V_{\pi^*}(t+1, \cdot)$, then use those values to compute $V_{\pi^*}(t, \cdot)$. This algorithm takes time $O(h \cdot A \cdot S^2)$ to compute the optimal policy, and is therefore polynomial in the size of an MDP. In the MMDP case, action set $A$ is itself exponentially sized, making this algorithm inefficient for general multiagent problems. However, this does not hold for MMDPs where the agents are *independent*, as defined below.

### Factored States and Independent Agent Models

In an MMDP, each agent sees the entire system state $s$. In large-scale multiagent systems, the requirement that all agents can observe everything may be too restrictive (Becker et al., 2004). Instead, we can consider the case where the state space is factored into per-agent sets, $S = \times_{i \in \alpha} S_i$. When agents must condition their policy only on a locally observed factor of the state space $s_i \in S_i$, the model becomes a decentralized MDP (Dec-MDP). Computing an optimal policy for a Dec-MDP is NEXP-complete (Bernstein et al., 2002), unless the transition and reward functions are *independent* (Becker et al., 2004). In this case the transition function and reward function are factored into per-agent components $T_i$, $R_i$ without dependence on other agents' local state:

$$T(s, a, s') = \prod_{i=1}^{n} T_i(s_i, a_i, s'_i), \tag{3}$$

$$R(s, a) = \sum_{i=1}^{n} R_i(s_i, a_i). \tag{4}$$

When states, transitions, and rewards are all factored, we have a model where each agent $i$ is represented by its own MDP, $\alpha_i = \langle S_i, A_i, T_i, R_i, h \rangle$. Then the optimal policy for each agent can also be computed separately from the other agents. This independence of models may seem to make them uninteresting from a multiagent perspective. However, this situation changes when the agents are faced with joint constraints, as we will see later, in Section 3.3.

### Partial State Observability

In partially observable settings, we model (in)visibility of the state space by assuming that an agent only has access to (noisy) observations, correlated with the state. Under partial observability, if the decision maker selects action $a$ and the environment state transitions to $s'$, then the agent observes $o$ with probability $\Omega(a, s', o) = P(o \mid a, s')$. While the probability to observe $o$ is dependent on the successor state $s'$, the agent never receives explicit information about $s'$. A Partially Observable MDP (Kaelbling, Littman, & Cassandra, 1998; Spaan, 2012, POMDP) thus generalizes the regular MDP model by adding a finite set of observations $o \in O$, and an observation function $\Omega : A \times S \times O \to [0, 1]$, resulting in tuple $\langle S, A, O, T, R, \Omega, h\rangle$.

In fully observable MDPs the state provides a Markovian signal on which to base optimal decisions. However, in POMDPs an individual observation does not provide sufficient information to make optimal decisions. All executed actions and observations encountered in the past can affect the knowledge the agent has about the current state, and hence a notion of memory is necessary to define an optimal decision making policy.

A Markovian planning signal can be recovered by using belief states $b$ in place of actual states $s$. A belief state $b$ is a vector of length $|S|$ defining the probability that the current environment state is $s$, i.e. $b(s) = \mathrm{P}(s)$. In other words, the vector characterizes the current belief of the decision maker regarding the actual environment state. A belief is a sufficient statistic for the full history of actions and observations, and therefore no other representations would provide the decision maker with more information about the history.

While interacting with the environment, the decision maker needs to update its belief $b$ after every action-observation pair $a, o$. The prior probability $\mathrm{P}(s' \mid b, a)$ that the environment transitions to state $s'$ is obtained by enumerating all possible ways $s'$ could be reached according to the current belief,

$$\mathrm{P}(s' \mid b, a) = \sum_{s \in S} \mathrm{P}(s' \mid s, a) b(s) = \sum_{s \in S} T(s, a, s') b(s). \tag{5}$$

Then, the posterior probability $b_a^o(s')$ that the system is in state $s'$ after taking action $a$ and observing $o$ is obtained through an application of Bayes' rule:

$$b_a^o(s') = \mathrm{P}(s' \mid b, a, o) = \frac{\mathrm{P}(o \mid a, s') \cdot \mathrm{P}(s' \mid b, a)}{\sum_{s'' \in S} \mathrm{P}(o \mid a, s'') \cdot \mathrm{P}(s'' \mid b, a)} = \frac{\Omega(a, s', o) \cdot \mathrm{P}(s' \mid b, a)}{\sum_{s'' \in S} \Omega(a, s'', o) \cdot \mathrm{P}(s'' \mid b, a)}.$$

Given this belief updating rule, the optimal policy can again be obtained using dynamic programming. However, due to the continuous nature of the space of beliefs $b$ this becomes more complicated. Dynamic programming algorithms for finite-horizon POMDPs make use of a representation that encodes the value function $V_\pi$ as a set of $|S|$-dimensional vectors. Based on this representation the optimal value function can be computed using incremental pruning (Cassandra et al., 1997; Walraven & Spaan, 2017). Unfortunately, computing an optimal policy for a POMDP is a PSPACE-complete problem (Papadimitriou & Tsitsiklis, 1987), making it intractable for reasonably sized models. Therefore it is common in practice to use point-based value iteration algorithms which compute an approximate solution by executing backups on a finite set of beliefs (Pineau et al., 2003; Spaan & Vlassis, 2005; Smith & Simmons, 2005; Kurniawati et al., 2008; Poupart et al., 2011).

The generalization from single-agent to multiagent POMDPs is the same as in the MDP case. Multiagent POMDPs can also be made decentralized, by factoring both the

state and observation spaces, such that agents only receive individual observations. Dec-POMDPs (Oliehoek & Amato, 2016) share NEXP-completeness with Dec-MDPs, unless they are transition, reward, and *observation* independent; the last one being formally defined analogously to transition independence.

### 3.2 Multiagent Planning with Shared Resources

The planning models presented in the previous section optimize a single objective, namely the expected value obtained by executing the policy. However, in most practical situations, the control policy should achieve this goal subject to some constraints. Constrained versions of single-agent MDPs have therefore been studied extensively, dating back at least to the works of Rossman (1977), Kallenberg (1983), and Beutler and Ross (1985). Moreover, Altman (1999) provided an overview of the theory and algorithms to solve constrained MDPs.

In this survey we focus on planning problems which include resources that impose constraints on the behavior of agents that are otherwise completely independent. A resource is an asset that agents should use in order to complete a planning task. If agents share the resources and if the availability of these resources is limited, then such resource constraints need to be considered while solving the planning problems of the agents. In the remainder of this section we formalize the basic terminology that we use for resources throughout this survey. Furthermore, we introduce various types of resources and multiple types of constraints imposed by these resources, which is important to understand and compare algorithms in the remaining parts of this survey. We start with a basic definition of the notion of resources in the context of planning:

**Definition 2** (Resource)**.** *A resource $j$ is a shared asset that agents have to use when executing actions during the execution of their policies.*

As an example of a resource, one can think about power that needs to be available in order to run a machine. Another example of a resource is money that needs to be invested when making planning decisions. Resources can affect individual agents, such as agents that carry a battery during plan execution. If agents share a global financial budget then the resource affects multiple agents. In both cases, limited availability of the resources imposes constraints on the behavior of the agents during plan execution, which needs to be considered by planning algorithms.

Consumption of resources during plan execution can be formalized using a resource consumption function, which defines the resource consumption when executing an action in a state. For agent $i$ the instantaneous consumption of resource $j$ is defined using a function $c_{i,j} : S_i \times A_i \to [0, c_{\max,i,j}]$, where $c_{\max,i,j}$ denotes the maximum consumption of resource $j$ by agent $i$. This function has the same form as the regular reward function, and it defines the consumption of resource $j$ for each state-action pair of agent $i$.

Two types of resources can be distinguished, which both lead to a different type of constraint on the behavior of the agents. The first resource type is a non-replenishable resource, for which a bounded quantity is available that is (partially) consumed over time until it is depleted, at which point no further consumption is allowed. Such resources can be seen as 'budgets', such as an amount of money, or the remaining energy stored in a battery. A key characteristic of this resource type is that it spans the entire plan execution, and

the total resource quantity represents a 'budget' that the agents can use when executing their policies. Therefore, we formalize the constraint imposed by this type of resource as a so-called budget constraint, as defined below.

**Definition 3** (Budget constraint)**.** *A budget constraint is a constraint imposed by a non-replenishable resource $j$ for which a bounded quantity $L_j$ is available during the entire plan execution. Resource consumption at time $t$ reduces the resource availability during the remaining time steps $t' > t$.*

Agents are collectively constrained to use at most $L_j$ units of the resource, which means that a constraint violation occurs if the agents collectively use more units during plan execution. More formally, the budget constraint imposed by resource $j$ is violated when it holds that

$$\sum_{t=1}^{h} \sum_{i=1}^{n} c_{i,j}(s_{i,t}, \pi_i(t, s_{i,t})) > L_j, \tag{6}$$

in which $\pi_i$ denotes the policy used by agent $i$ and $s_{i,t}$ denotes the state of agent $i$ at time $t$.

The second type of resource that can be distinguished is a replenishable resource for which the same quantity is available during every time instant. For example, the number of vehicles on a road-segment is bounded by the number of lanes, but once the vehicles have passed the lanes are available again. Other examples include bandwidth in a communication network, instantaneous power production and transmission capacity in an electricity grid, but also shared tools or CPU cycles. This resource type imposes so-called instantaneous constraints, which represent the maximum consumption at each decision point.

**Definition 4** (Instantaneous constraint)**.** *An instantaneous constraint is a constraint imposed by a replenishable resource $j$ for which a bounded quantity $L_j$ is available during each time instant. Resource consumption at time $t$ does not affect the resource availability during subsequent time steps $t' > t$.*

For instantaneous constraints the agents violate the constraint on resource $j$ at time $t$ when it holds that

$$\sum_{i=1}^{n} c_{i,j}(s_{i,t}, \pi_i(t, s_{i,t})) > L_j. \tag{7}$$

As can be seen, the only difference with budget constraints is that the resource limit $L_j$ affects only individual time steps, rather than all steps of the execution of a policy. Instantaneous constraints can also be applied in settings where the resource availability is time dependent. In this case the definitions remain unchanged, except that the resource quantity becomes $L_{j,t}$, thereby conditioning the limit of resource $j$ also on the current time step $t$.

Both Definition 3 and 4 consider constraints that are *hard*, or strict, meaning that they should hold for all realizations of uncertainty in the state transition dynamics of the agents. However, another definition for the constraints is possible, where the constraint is *soft*, which means we accept policies that meet a limit $L$ in expectation. Whether a constraint should be modeled as hard or soft depends on the problem domain: soft constraints are appropriate for concepts like quality of service, whereas hard constraints are appropriate when exceeding them causes significant harm to the agent or its environment, for example in grids where exceeding capacity constraints leads to blackouts. What type of constraint is used also has

complexity consequences, as soft constraints can be handled in polynomial time by using relaxations, as we will see later.

### 3.3 Agent Coupling Imposed by Resource Constraints

The resource constraints couple the decision making problems of otherwise independent agents. In other words, the presence of the constraints cause agents to exert influence on the (allowable) decisions of other agents. Constrained but otherwise independent multiagent systems can be characterized as 'weakly coupled' (Meuleau et al., 1998; Adelman & Mersereau, 2008). This claim appeals to the intuitive idea that, from the perspective of one particular agent, the other agents exert anonymous influence (Robbel et al., 2016): for agent $i$ to know whether it can use resources, it only needs to know if the cumulative demand of the other agents leaves sufficient room, not which agents use the resource.

The independence assumption needs to be treated carefully from a modeling point of view. The model assumes that each agent is modeled using a separate MDP, which means that the state transitions and rewards in the MDP do not depend on the transitions and rewards defined by the MDPs corresponding to other agents. However, coupling the agents through shared resources creates dependencies between agents, because state transitions of an agent may affect the resource availability and state transitions of other agents in the remaining steps. In this case an optimal policy for the constrained planning problem is conditional on the joint state and the availability of the shared resources, similar to the joint policies in the multiagent MDP model for unconstrained problems (Boutilier, 1996).

Two types of algorithms are covered in this survey. The first type considers the coupled planning problem, such that agents choose their actions based on the joint state and resource availability. Other algorithms in this survey do not model the dependencies, and they let agents choose their actions based on their individual states while ignoring the behavior of the other agents. Although this assumption potentially sacrifices optimality of the decisions, it typically provides better scalability for problems with many agents.

## 4. Algorithms for Constrained Multiagent Markov Decision Problems

The presence of resource constraints forces agents to coordinate their decisions in order to satisfy them. In this section we identify two main categories of solution approaches which can be used to achieve this coordination between agents.

**Resource Preallocations** One of the central assumptions in CMMDPs is that it consists of multiple independent agents that are only coupled through one or more resources that need to be shared by the agents. The decision processes of the agents can be decoupled by allocating resources to the agents prior to execution using a so-called resource preallocation. Such a preallocation enables the agents to compute a policy that respects the resource allocation, and it is no longer necessary to consider the states of other agents and the resources allocated to other agents. As a result, after preallocating resources the agents can safely choose a resource-consuming action without coordinating this decision with other agents.

The strictness of the constraints influences what type of preallocation is acceptable. Hard constraints require deterministic preallocations, whereas soft constraints can also be met with

stochastic preallocations. A deterministic preallocation is an allocation of resources to agents prior to policy execution, such that each possible realization of the resource consumption of an agent does not exceed the quantity that is allocated to the agent. For example, (Wu & Durfee, 2010; Agrawal et al., 2016) propose methods that use such a solution approach.

Stochastic preallocation approaches allocate resources to agents under the condition that their *expected* resource consumption does not exceed the quantity that is allocated to the agent. This is a common approach, taken for example by (Altman, 1999; Yost & Washburn, 2000; Isom et al., 2008; Kim et al., 2011; Poupart et al., 2015; Walraven & Spaan, 2018; Dolgov & Durfee, 2003; De Nijs et al., 2017).

Both deterministic and stochastic preallocations can be seen as offline solution methods to achieve coordination in constrained multiagent systems. In both cases the resources are assigned to the agents when solving the planning problem, and online communication during policy execution is therefore not required.

**Online and Hybrid Online/Offline Solution Approaches** The communication-free advantage of resource preallocations comes at a cost: the resources are allocated under maximum uncertainty, before any realizations are known. Stochastic system evolution may see an agent end up unable to make use of a preallocated resource as a result. Online solution strategies, on the other hand, may use communication between the agents to make or adjust the resource assignments at the point in time that resources are actually used. These strategies fall into two broad groups, those that coordinate purely online, and hybrid methods that can use communication to improve or repair initial offline coordination.

A fully online strategy sees agents perform their planning and coordination without a precomputed baseline policy. This means that the agents have to decide in a relatively short timeframe what they should do to maximize their expected reward, for example through a forward search of their currently reachable state space. As a result, minimal computational resources are spent on parts of the state space that are not reached in practice (e.g., De Nijs et al., 2015).

Hybrid strategies can be seen to combine an offline planning phase with an online recourse strategy. In the offline phase agents compute an initial decision making policy based on a coordination strategy similar to resource preallocation. During online policy execution, agents coordinate with each other or through a centralized mechanism, to ensure that the resources are deployed to the best effect while respecting the constraints. For example, it may be effective to allocate resources to agents based on the expected value gained or lost due to changing its action from its intended course. Because the online phase guarantees constraint satisfaction, the offline phase can be more aggressive in its resource demands compared with preallocation solutions. Depending on how the online phase is implemented, it may also involve (partial) replanning to optimize future resource usage. Hybrid approaches are taken by Meuleau et al. (1998), Boutilier and Lu (2016), Undurti and How (2010), Lee et al. (2018), among others.

An important advantage of online solutions is that it potentially allows agents to base their decisions on the state of other agents, as well as on the current resource availability. Due to the dependencies between the agents introduced by the resource coupling, this may provide better decisions than the decisions made with offline methods.

| | Connected | PO | Strictness | Timespan |
|---|---|---|---|---|
| Stochastic preallocation (4.1.1 ) | * | | soft | * |
| PO stochastic preallocation (4.1.2) | * | ✓ | soft | * |
| Deterministic preallocation (4.2) | * | | hard/soft | * |
| Online/hybrid allocation (4.3.1) | ✓ | | hard/soft | * |
| PO online/hybrid allocation (4.3.2) | ✓ | ✓ | hard/soft | * |

Table 2: Taxonomy of solution approaches for constrained multiagent systems that makes two main restrictions explicit: i) online methods require coordination while execution ("Connected"), and ii) stochastic preallocations cannot deal with hard constraints.

**Taxonomy of Solution Approaches** In each of the two solution approaches identified above, domain properties such as whether the state is fully observable and whether constraints are instantaneous or define a budget for the whole planning horizon also influence specifics of the algorithms, but to a lesser extent than whether coordination is done during execution. The solution approaches above thus play the most important role in the solution taxonomy for planning and coordination in constrained multiagent systems. In disconnected domains, online solutions cannot be used, because of their reliance on communication to coordinate decisions. However, online solutions exist for both hard and soft resource constraints. For offline methods we can distinguish deterministic and stochastic preallocations, of which only deterministic preallocations can be used to deal with hard constraints. This taxonomy, as visualized in Table 2, thus sketches the general landscape of solution strategies in planning problems with shared resources. The taxonomy can also be used to decide what kind of algorithms need to be applied given a planning problem and given the characteristics of the resources. For this purpose it includes pointers to the next sections, in which we describe the specific planning algorithms in more detail, and where we also further refine this taxonomy to include the timespan of resource constraints.

For each algorithm we provide the intuitions behind it, as well as mathematical details to illustrate how the algorithms operate, using pseudocode where this seems the most effective way to communicate this. Following the in-depth treatment of the algorithms mentioned in the coming subsections 4.1–4.3, organized according to the taxonomy from Table 2, in Section 4.4 we discuss observed trends and gaps in the state of the art of solving constrained multiagent Markov decision problems.

### 4.1 Stochastic Preallocations

Stochastic preallocations algorithms can be used to compute policies for the agents such that the agents do not violate the resource constraints in expectation, as defined in Eq. 8, in which the expectation defines the expected resource consumption for resource $j$ at time $t$ while executing the policy $\pi_i$,

$$E[C^{j,t}_{\pi_i}] \leq L^i_{j,t} \quad \forall i, j, t. \tag{8}$$

The approaches for stochastic preallocations use linear programming to express such resource constraints. We therefore start with a brief introduction to linear programming formulations for MDPs without constraints. Traditional MDP algorithms exploit the fact

that the Bellman equation not only describes optimality, but also prescribes the method to get there through the fixed point: starting from arbitrary initial values, repeated application of the Bellman equation eventually results in the optimal value function. This idea is also used in the 'primal' linear program (LP) for solving MDP policies (Littman et al., 1995):

$$\begin{aligned} \min_{v_{1,s}} \ & \sum_{s \in S} \mathrm{P}(1,s) v_{1,s} \\ \text{s.t. } & v_{t,s} \geq R(s,a) + \sum_{s' \in S} \mathrm{P}(s' \mid s,a) v_{t+1,s'} \quad \forall t < h, s, a \\ & v_{h,s} \geq R(s,a) \qquad \forall s, a. \end{aligned} \tag{9}$$

In this LP the variables $v_{t,s}$ hold the expected value of following the computed policy from time $t$ and state $s$ onward, $v_{t,s} = V[t,s]$. The constraints encode the Bellman equation, by ensuring that the value $v_{t,s}$ is *at least as large* as the expected value of any action (including the best action). By minimizing $v_{1,s}$, the solution is made tight to the strongest constraint, which is given by the value of the best action.

Unfortunately, because the chosen action is implicit in the model, the primal LP is unsuitable to use with constraints on the resource usage of actions. However, by the strong duality theorem, the LP (9) has an equivalent 'dual' LP, which does have variables for actions (Littman et al., 1995):

$$\begin{aligned} \max_{x_{t,s,a}} \ & \sum_{t=1}^{h} \sum_{s \in S} \sum_{a \in A} x_{t,s,a} R(s,a) \\ \text{s.t. } & \sum_{a' \in A} x_{t+1,s',a'} = \sum_{s \in S} \sum_{a \in A} \mathrm{P}(s' \mid s,a) x_{t,s,a} \quad \forall t < h, s' \\ & \sum_{a' \in A} x_{1,s',a'} = \mathrm{P}(1,s') \qquad \forall s'. \end{aligned} \tag{10}$$

In this LP, the $x_{t,s,a}$ variables encode the unconditional probability that the computed policy uses action $a$ in state $s$ at time $t$, $x_{t,s,a} = \mathrm{P}(t,s,a \mid \pi)$. The constraints ensure conservation of flow, meaning that the sum of probability coming out of $s'$ at time $t+1$ equals the total incoming probability as a result of transitions to $s'$. The term $\mathrm{P}(1,s')$ denotes the probability that the initial state is $s'$ in the first time step. The LP optimizes the expected value directly by considering the cumulative rewards discounted by their probability of being awarded. Note that action selection may be randomized, as the probability of an action being selected is given by:

$$\mathrm{P}(a \mid \pi(t,s)) = \frac{x_{t,s,a}}{\sum_{a' \in A} x_{t,s,a'}}. \tag{11}$$

Thus far, we have presented the LPs in the context of planning for a single MDP. In the case of a multiagent MDP with independent agent dynamics, we can add all their models together in a single LP with only a polynomial increase in the total size of the program,

resulting in the following multiagent dual LP:

$$\begin{aligned} \max_{x_{i,t,s,a}} & \sum_{i=1}^{n} \sum_{t=1}^{h} \sum_{s \in S_i} \sum_{a \in A_i} x_{i,t,s,a} R_i(s,a) \\ \text{s.t.} & \sum_{a' \in A_i} x_{i,t+1,s',a'} = \sum_{s \in S_i} \sum_{a \in A_i} \mathrm{P}_i(s' \mid s,a) x_{i,t,s,a} \quad \forall i, t < h, s' \\ & \sum_{a' \in A_i} x_{i,1,s',a'} = \mathrm{P}_i(1,s') \qquad \forall i, s'. \end{aligned} \tag{12}$$

This multiagent dual LP forms the basis for the stochastic preallocation approaches presented in this section. First we discuss two approaches for fully observable models, and then three methods that can be applied also when the state is not fully observable.

#### 4.1.1 Stochastic Preallocations for Fully Observable Models

The first approach below leverages a linear programming formulation based on the Bellman equation for MDPs, and the second approach uses a column generation approach for linear programming in order to split the optimization problem into subproblems that can be solved independently.

Constrained MDPs (CMDPs) leverage the multiagent dual LP in order to handle additional constraints (Altman, 1999), such as the ones we intend to model. In our case, we can add the resource constraints by adding the following constraint to LP (12):

$$\sum_{i=1}^{n} \sum_{s \in S_i} \sum_{a \in A_i} x_{i,t,s,a} \cdot c_{i,j}(t,s,a) \leq L_{j,t} \qquad \forall j, t. \tag{13}$$

Because $x_{i,t,s,a}$ is the probability that agent $i$ reaches state $s$ at time $t$ and takes action $a$, we obtain the expected consumption of the agent by multiplying with the consumption of the action. The resulting LP therefore computes a solution which maximizes the expected value of the agents' joint policy, subject to it satisfying each of the constraints in expectation. Because the model is an LP, its optimal solution can be found in a polynomial time, making this a highly tractable approach for computing a stochastic preallocation.

The preallocation LP that we just discussed decouples the constraints from the agents' planning problems, but it still requires optimizing a single large centralized program. A column generation procedure has been proposed which allows agents to solve their individual subproblems independently. Column generation (Gilmore & Gomory, 1961) is an effective technique for decomposing combinatorial optimization problems, provided there is some method to generate new potential solutions to subproblems efficiently. The technique uses the insight that when a linear program is used to select solutions from an exhaustive set, the simplex algorithm iteratively adds elements to the solution which are not 'priced out' by the $\lambda$ prices computed in the dual solution. An element is priced out if its contribution to the objective per unit of the constraint is less than $\lambda$. If we can generate the optimal element to be selected on the fly, we avoid maintaining the exhaustive set of elements explicitly. Generating the element comes down to optimizing an ancillary problem subject to the $\lambda$ costs.

Yost and Washburn (2000) identified (for POMDPs, but it straightforwardly applies here) that we can compute a policy that optimizes for $\lambda$ efficiently, allowing the use of column generation to solve constrained MDPs. Just as the expected value of a policy is given by a recursive function $V_\pi$, the expected consumption of a policy, which we will denote $C_\pi$, follows the same structure:

$$C_{\pi,j}(t,s) = c_j(t,s,\pi(s)) + \sum_{s' \in S} \big(\mathrm{P}(s' \mid s, \pi(s)) \cdot C_{\pi,j}(t+1, s')\big). \tag{14}$$

When we are searching for the maximally-improving column, we are searching for the column satisfying

$$\max_\pi \left(V_\pi(t,s) - \lambda \cdot C_\pi(t,s)\right) = \max_\pi V^C_{\pi,\lambda}(t,s), \tag{15}$$

where $\cdot$ takes the dot product of the price vector $\lambda$ with the consumption vector $C_\pi$.

Both $V_\pi$ and $C_\pi$ are Markovian, and therefore we can write out the optimization problem from the perspective of a single current state:

$$\begin{aligned}
V^C_{\pi,\lambda}(t,s) = V_\pi(t,s) - \lambda \cdot C_\pi(t,s) = \\
R(s,a) + \sum_{s'} \mathrm{P}(s' \mid s,a) V_\pi(t{+}1, s') - \lambda \cdot \Big(c(t,s,a) + \sum_{s'} \mathrm{P}(s' \mid s,a) C_\pi(t{+}1,s')\Big) = \\
R(s,a) - \lambda \cdot c(t,s,a) + \sum_{s'} \mathrm{P}(s' \mid s,a)\Big(V_\pi(t{+}1,s') - \lambda \cdot C_\pi(t{+}1,s')\Big) = \\
R(s,a) - \lambda \cdot c(t,s,a) + \sum_{s'} \mathrm{P}(s' \mid s,a) V^C_{\pi,\lambda}(t{+}1,s').
\end{aligned}$$

This equation resolves to a resource-priced Bellman-like recursive form, which we can use as the objective function in the traditional dynamic programming algorithm. Therefore, we can compute the optimal column to be selected at the same complexity as planning a regular MDP policy.

In the multiagent case a newly computed policy optimized for objective (15) is then added to the set of potential policies $Z_i$ for each agent $i$, which together form the search space of the column generation 'master LP' selecting the optimal mix of policies subject to constraints:

$$\begin{aligned}
\max_{x_{i,k}} \; & \sum_{i=1}^{n} \sum_{\pi_k \in Z_i} x_{i,k} V_{\pi_k}(1, s_1), \\
\text{s.t.} \; & \sum_{i=1}^{n} \sum_{\pi_k \in Z_i} x_{i,k}\, C_{\pi_k,j,t}(1,s_1) \le L_{j,t} \quad \forall j,t, \\
& \sum_{\pi_k \in Z_i} x_{i,k} = 1 \qquad \forall i, \\
& x_{i,k} \ge 0 \qquad \forall i,k.
\end{aligned} \tag{16}$$

Putting the master LP and the planning subroutines together results in Algorithm 1. The resulting solution defines a probability distribution over agent policies, such that the probability that agent $i$ will follow policy $\pi_{i,k}$ over the entire horizon is given by:

$$\mathrm{P}(\pi_i = \pi_{i,k}) = \frac{x_{i,k}}{\sum_{\pi_{k'} \in Z_i} x_{i,k'}}. \tag{17}$$

**Algorithm 1** Column generation for CMDP $M$ (Yost & Washburn, 2000)

$\lambda = 0$, $\lambda' = \infty$, $Z = \emptyset$
1: **while** $\lambda \neq \lambda'$ **do**
2: $\quad \lambda \leftarrow \lambda'$
3: $\quad \forall i\colon \pi_{i,\text{new}} \leftarrow \text{PLAN}(M_i, \lambda)$ ▷ Eq. 15
4: $\quad Z_i \leftarrow Z_i \cup \{\pi_{i,\text{new}}\}$
5: $\quad \langle x, \lambda' \rangle \leftarrow \text{SOLVELP}(Z)$ ▷ Eq. 16
6: **end while**
7: **return** $\langle x, Z \rangle$

Algorithm 1, like the Constrained MDP LP, computes optimal joint policies which satisfy the constraints in expectation. It does so in the same worst-case complexity, but the algorithm has two practical scalability benefits: (i) planning the individual agent policies on line 3 can be done fully in parallel, and (ii) the dynamic programming algorithm optimizing Eq. 15 directly exploits the time-recursive structure present in the MDP. Therefore, it can be expected that in practice column generation will be significantly more scalable than directly solving the CMDP dual LP of Eq. 12.

#### 4.1.2 Stochastic Preallocations for Partially Observable Models

Stochastic preallocation algorithms for POMDPs can be grouped into three categories. The first category consists of algorithms for unconstrained POMDPs, augmented with additional constraints on the computed policy. The second category takes the opposite approach, and augments algorithms for constrained MDPs with partial observability. The third category formalizes the constrained planning problem as a sequence of unconstrained planning problems, similar to the column generation algorithm for MDPs. In the remainder of this section we describe the algorithms belonging to these three categories in more detail.

Isom et al. (2008) consider optimal single-agent POMDP planning in domains with an infinite planning horizon and a resource constraint that should be satisfied in expectation. Optimal unconstrained POMDP solutions are typically computed using exact value iteration (Cassandra et al., 1997), which executes dynamic programming iterations to compute a value function $V(b)$. This value function can be represented using a finite set of $|S|$-dimensional vectors (Sondik, 1978). A value function represented by such vectors can be formalized as follows:

$$V(b) = \max_{\alpha \in V} \alpha \cdot b, \tag{18}$$

in which we intentionally overload $V$ to represent both the value function $V$ and the vector set that defines this function.

Before the resource constraint can be integrated in exact value iteration, it is important to know the expected resource consumption of the policy computed by the algorithm. In order to keep track of expected resource consumption in exact value iteration, the value function is defined using pairs $(\alpha_r, \alpha_c) \in V$, in which $\alpha_r$ represents reward and $\alpha_c$ represents resource consumption. Based on these pairs the value function can be formalized as follows:

$$V(b) = \max_{(\alpha_r, \alpha_c) \in V} \alpha_r \cdot b. \tag{19}$$

If the pair $(\alpha_r, \alpha_c)$ is the maximizing pair in this equation, then $\alpha_r \cdot b$ denotes the expected reward and $\alpha_c \cdot b$ the expected resource consumption. Keeping track of additional resource consumption vectors $\alpha_c$ in value iteration requires only a minor additional computation that is easy to integrate. For more details we refer to Isom et al. (2008).

The computed policy is characterized by the pairs $(\alpha_r, \alpha_c) \in V$, which means that a constraint can be integrated by removing the vectors that correspond to actions that violate the resource constraint. Exact value iteration executes a pruning algorithm to discard dominated vectors (Walraven & Spaan, 2017), and it turns out that it is relatively easy to integrate the additional constraint in this pruning algorithm. The pruning procedure starts with an empty vector set $U$, and it iteratively checks whether candidate vector pairs $(w_r, w_c)$ should be added to the set $U$. It does this by finding a belief point $b$ in which $(w_r, w_c)$ dominates all other vectors in $U$ while remaining resource feasible. This computation can be performed by solving the following mixed-integer linear program[1]:

$$
\begin{aligned}
&\max d \\
&\text{s.t. } b \cdot w_c \leq L \\
&\quad (w_r - u_r^k) \cdot b + \mathcal{M} q^k \geq d \quad \forall (u_r^k, u_c^k) \in U \\
&\quad u_c^k \cdot b + \mathcal{M}(1 - q^k) \geq L \quad \forall (u_r^k, u_c^k) \in U \\
&\quad \sum_{s \in S} b_s = 1 \\
&\quad b_s \geq 0 \qquad \forall s \\
&\quad q^k \in \{0, 1\} \qquad \forall k,
\end{aligned}
\tag{20}
$$

in which $L$ denotes the resource limit and $\mathcal{M}$ is a sufficiently large constant. The second and third constraint together define that $(w_r - u_r^k) \cdot b \geq d$ should hold if $u_c^k \cdot b \leq L$. If the optimization problem is feasible and if $d > 0$ then candidate vector should be added to $U$. The mixed-integer linear program replaces the original linear program that is used by exact value iteration. Unfortunately, from a practical point of view the resulting algorithm is not useful due to the limited scalability of exact POMDP algorithms and due to the large number of mixed-integer linear programs that need to be solved while computing a solution. In addition to scalability issues, the policies computed with exact value iteration are deterministic, which may be suboptimal in constrained problems. Furthermore, the pruning operator ensures that every intermediate policy satisfies the constraint during the execution of value iteration, while it is only required that the final policy satisfies the constraint. This additional restriction may lead to policies that are unnecessarily conservative.

The limitations of exact value iteration for constrained problems have been addressed by Kim et al. (2011). They propose an approximate point-based value iteration algorithm which takes resource consumption into account. It does so by integrating the concept of admissible resource consumption, which enables the algorithm to keep track of resource consumption while executing point-based backups. The admissible resource consumption $d_t$ defines the resource quantity that can be consumed during policy execution starting from

1. Compared to the formulation by Isom et al. (2008) we modified the constraints such that the formulation is consistent with the problem setting considered in this survey.

**Algorithm 2** Constrained point-based backup stage (Kim et al., 2011)

$V_{n+1} \leftarrow \emptyset$
$\Gamma^{a,o} \leftarrow$ backprojections obtained for $\alpha^k \in V_n$ with $g^k_{a,o}(s) = \sum_{s'} P(o|s',a)P(s'|s,a)\alpha^k(s')$
**for** $(b,d) \in B$ **do**
  **for** $a \in A$ **do**
    **for** $o \in O$ **do**
      $d_o \leftarrow \frac{1}{\gamma}(d - C(b,a))P(o \mid b,a)$
      solve LP (22) with pairs $(\alpha^k_r, \alpha^k_c) \in \Gamma^{a,o}$ and $(b, d_o)$ to get probabilities $\tilde{w}_k$
      $\tilde{\alpha}^{a,o}_r \leftarrow \sum_k \tilde{w}_k \alpha^k_r$
      $\tilde{\alpha}^{a,o}_c \leftarrow \sum_k \tilde{w}_k \alpha^k_c$
    **end for**
    $\alpha^{(b,d),a}_r \leftarrow \alpha^a_r + \gamma \sum_{o \in O} \tilde{\alpha}^{a,o}_r$ in which $\alpha^a_r$ is the immediate reward vector for $a$
    $\alpha^{(b,d),a}_c \leftarrow \alpha^a_c + \gamma \sum_{o \in O} \tilde{\alpha}^{a,o}_c$ in which $\alpha^a_c$ is the resource consumption vector for $a$
  **end for**
  $\Gamma^{(b,d)} \leftarrow \bigcup_{a \in A} \{(\alpha^{(b,d),a}_r, \alpha^{(b,d),a}_c)\}$
  solve LP (22) with $\Gamma^{(b,d)}$ and $(b,d)$ to get probabilities $w_k$
  $V_{n+1} \leftarrow V_{n+1} \cup \{(\alpha^k_r, \alpha^k_c) \in \Gamma^{b,d} \mid w_k > 0\}$
**end for**
**return** $V_{n+1}$

time $t$:

$$d_t = \begin{cases} L & t = 0 \\ \frac{1}{\gamma}(d_{t-1} - C(b_{t-1}, a_{t-1})) & \text{otherwise,} \end{cases} \tag{21}$$

in which $b_{t-1}$ denotes the belief at time $t-1$ and $a_{t-1}$ is the action that has been executed in this belief. The term $C(b_{t-1}, a_{t-1})$ denotes the expected resource consumption for this action execution. Intuitively, the admissible resource consumption $d_t$ indicates how many additional units of a resource can be consumed in the remaining time steps, and it is updated after every action execution.

For a given belief point $b$ and admissible resource consumption $d$, the action to be executed can be determined by solving the following linear program based on the vector pairs $(\alpha^k_r, \alpha^k_c) \in V$:

$$\begin{aligned} \max_{w_k} \; & b \cdot \sum_k w_k \alpha^k_r \\ \text{s.t. } & b \cdot \sum_k w_k \alpha^k_c \leq d \\ & \sum_k w_k = 1 \\ & w_k \geq 0 \qquad\qquad \forall k, \end{aligned} \tag{22}$$

which yields a probability $w_k$ for each vector pair in $V$. The action to be executed is determined by sampling a vector pair based on the distribution. The resulting vector pair defines the action.

The constrained point-based value iteration algorithm combines the concept of admissible resource consumption and the linear program to compute a vector-based value function $V$

in such a way that the resource constraint is respected. The algorithm starts by sampling a set $B$ containing belief-admissible consumption pairs $(b, d)$. After that, the algorithm computes a series of value functions $V_1, V_2, \ldots$ by executing backup stages. Within a backup stage a new value function $V_{n+1}$ is computed given the value function $V_n$. The full procedure is described in Algorithm 2. For each $(b, d) \in B$ and $a \in A$ it constructs a vector $\alpha_r^{(b,d),a}$ using a linear combination that considers the admissible resource consumption and the observations that can be made in the next timestep. When constructing the final value function $V_{n+1}$ the linear program is used again to discard the vectors that cannot be chosen for the given admissible resource consumption $d$. Point-based value iteration keeps executing backup stages until the value function converges.

Given a value function $V$ that has been computed using constrained point-based value iteration, policy execution can be performed using the linear program in (22) and some additional bookkeeping to keep track of the admissible resource consumption. Each action selection requires solving a linear program, but it can be expected that this operation is relatively cheap to execute and it does not represent a major bottleneck.

The constrained variant of point-based value iteration scales better than the exact algorithm for POMDPs with constraints because it limits policy computation to a finite number of reachable beliefs. This restriction provides significant computational benefits, but unfortunately it also creates undesirable effects in problems with constraints. If the set $B$ does not contain all belief points that are reachable during policy execution, then it is not guaranteed that the computed policy satisfies the resource constraint in expectation. This means that constraint satisfaction highly depends on the beliefs that have been sampled prior to executing point-based value iteration.

Both aforementioned algorithms extend existing unconstrained POMDP algorithms with additional constraints. Poupart et al. (2015) follow an alternative strategy and extend an existing constrained MDP algorithm with partial observability. They observe that a POMDP with constraints can be seen as a constrained belief-state MDP, and this makes it possible to apply the linear programming formulation for constrained MDPs, similar to the algorithm that we have discussed for MDPs in Section 4.1.1. Assuming that the set $B$ contains all reachable beliefs, the optimization problem can be written as follows:

$$\begin{aligned}
\max_{y_{b,a}} \; & \sum_{b \in B} \sum_{a \in A} \bar{R}(b,a) y_{b,a} \\
\text{s.t.} \; & \sum_{a' \in A} y_{b',a'} = \delta(b', b_1) + \gamma \sum_{b \in B} \sum_{a \in A} P(b' \mid b, a) && \forall b' \\
& \sum_{b \in B} \sum_{a \in A} \bar{C}(b,a) y_{b,a} \leq L \\
& y_{b,a} \geq 0 && \forall b \in B, a \in A,
\end{aligned} \tag{23}$$

in which $\bar{R}(b,a) = \sum_s R(s,a)b(s)$ and $\bar{C}(b,a) = \sum_s C(s,a)b(s)$ denote the reward and cost function defined over beliefs rather than states. Similarly, the function $P(b' \mid b, a) = \sum_{\{o \mid b_a^o = b'\}} P(o \mid b, a)$ denotes the transition function over beliefs. The function $\delta(b', b_1)$ is equal to 1 if $b' = b_1$ and 0 otherwise. The variables $y_{b,a}$ define a stochastic policy $\pi(b,a) = y_{b,a} / \sum_{a' \in A} y_{b,a'}$, such that $\pi(b,a)$ denotes the probability to execute $a$ in belief $b$.

The formulation above can only be applied after enumerating all reachable beliefs, which is typically not feasible. Instead, a subset $\hat{B}$ can be used to obtain a solution based on a finite

number of beliefs, but in that case the transition function $P(b' \mid b, a)$ is no longer well-defined if some reachable beliefs are not present in $\hat{B}$. Fortunately, it is possible to replace a belief $b \notin \hat{B}$ with a convex combination of beliefs that are present in $\hat{B} = \{b_k\}_{k=1,\ldots,|\hat{B}|}$. This convex combination can be obtained using the following linear program:

$$\begin{aligned} \min \; & \sum_k w_k \left\| b - b_k \right\|_2^2 \\ \text{s.t.} \; & \sum_k w_k b_k(s) = b(s) \quad \forall s \\ & \sum_k w_k = 1 \\ & w_k \geq 0, \end{aligned} \tag{24}$$

such that $w(b_k, b) = w_k$ is the probability assigned to $b_k \in \hat{B}$ when interpolating $b$. The convex combination can be obtained to construct an approximate transition function:

$$\tilde{P}(b' \mid b, a) = \sum_o P(o \mid b, a) w(b', b_a^o). \tag{25}$$

This approximation becomes more accurate once more reachable beliefs are present in the set $\hat{B}$. Given the subset $\hat{B}$ and the transition function approximation it becomes feasible to solve the constrained belief-state MDP in Eq. 24 without full belief enumeration.

The solution provided by the linear program based on $\hat{B}$ can be executed by constructing a stochastic finite state controller. For each belief $b \in \hat{B}$ a node $n_b$ is created, in which action $a$ is chosen with probability $P(a \mid n_b) = y_{b,a} \;/\; \sum_{a' \in A} y_{b,a'}$. After executing action $a$ and observing $o$ the finite state controller transitions to node $n_{b'}$ with probability $P(n_{b'} \mid n_b, a, o) = w(b', b_a^o)$. For additional details regarding exact policy evaluation and the expansion of the set $\hat{B}$ we refer to Poupart et al. (2015).

The final stochastic preallocation method for POMDPs that we consider is based on column generation, similar to the column generation algorithm for MDPs discussed in Section 4.1.1. The original column generation algorithm by Yost and Washburn (2000) applies directly to POMDPs, but it requires solving a series of unconstrained POMDPs using an exact solution algorithm (i.e., the exact PLAN step in Algorithm 1). The scalability of exact POMDP algorithms is typically limited, which renders the column generation approach intractable in the POMDP setting, and it prevents application of this approach to more realistic application domains. Walraven and Spaan (2018) changed the original column generation algorithm into an approximate algorithm that is called Column Generation algorithm for Constrained POMDPs (CGCP), which offers much better scalability for problems with partial observability.

The scalability problems of the PLAN step are addressed by integrating point-based value iteration algorithms, which dramatically decrease the running time that is required. However, after integrating point-based algorithms there are additional scalability problems that remain. The new policies $\pi_{i,\text{new}}$ need to be evaluated in order to obtain the expected reward and expected resource consumption that is used in the column in the linear program in Eq. 16. Performing exact policy evaluation is computationally demanding, which means that the approach still does not scale well. Walraven and Spaan (2018) propose to convert

the policy $\pi_{i,\text{new}}$ into a finite state controller that is approximately equivalent, which can be evaluated efficiently by solving a linear constraint system. As a result, the PLAN step becomes feasible to execute in the POMDP setting and it makes it possible to use the column generation algorithm for larger problems.

Compared to other stochastic preallocations for POMDPs, the type of stochastic solution computed by CGCP can be seen as an important disadvantage; rather than randomizing the decisions at every time step, it only randomizes the policy choice of the agents prior to execution, which potentially increases the variance of the cost incurred during execution. Furthermore, there are no guarantees about the number of iterations that column generation requires to converge. In the POMDP setting this may be problematic, because one iteration of column generation requires solving a POMDP, which is more computationally demanding than solving subproblems in the MDP setting.

However, compared to constrained value iteration discussed above, the column generation algorithm also provides several advantages. First, it solves the constrained POMDP problem as a sequence of unconstrained POMDPs, which means that state-of-the-art unconstrained point-based methods are easy to integrate. Second, the method naturally supports problems with multiple independent agents, which is a problem setting that has not been considered by the other methods. Third, the column generation approach makes it possible to parallelize the computations, since subproblems belonging to the individual agents can be solved independently in parallel. Finally, the approximate method based on column generation guarantees that the resource constraint is satisfied in expectation, which is not the case in constrained point-based value iteration. In addition, CGCP has been shown to outperform the LP in Eq. 23 empirically in several domains.

#### 4.1.3 Bounded Risk of Constraint Violations

A special type of constraint occurs when we want to bound the probability of constraint violations by some tolerance $\alpha$. This problem occurs when domains exhibit constraints which are in between the typical soft and hard categories, and has the added benefit of allowing the system operator to influence the risk/reward balance of computed policies. For the single-agent case, Dolgov and Durfee (2003) propose to approximate the constraint using Markov's inequality, which gives a rough but effective initial bound. In the multiagent case, De Nijs et al. (2017) apply Hoeffding's inequality on the sum of independent random variables to improve the bound. Nevertheless, this bound also proves to be loose empirically, prompting the authors to explore a gradient-descent style optimization to obtain relaxed bounds that closely match the tolerance level. Unfortunately, the relaxation scheme relies on empirical estimates of the violation probability, and therefore it cannot be used to provide a strict guarantee on the probability of constraint violations. However, if the domain does not require such a strict guarantee then it can be an attractive scheme to obtain policies.

These strategies bound the risk of constraint violation by imposing tighter constraints on the space of admissible policies, leaving the variance in consumption free. Alternative formulations and frameworks have been proposed which maximize the ratio between expected reward and its standard deviation (Sobel, 1985; Huang & Kallenberg, 1994). These methods potentially could be applied to (also) reduce the variation in expected resource consumption, but further work is needed to establish if this can be scaled up to multiagent systems.

### 4.2 Deterministic Preallocations

The Constrained MDP formalism is a powerful stochastic preallocation method for resource-constrained multiagent MDPs, but its fundamental drawback is that it meets the constraints only in expectation. Wu and Durfee (2010) address this by proposing several Mixed-Integer Linear Programs (MILP) that can be used to obtain deterministic preallocations that guarantee strict adherence to the constraints. When we have a model with binary resources, we can add the following binary variables and constraints to the LP (12):

$$\begin{aligned} &\sum_{s\in S_i}\sum_{a\in A_i} x_{i,t,s,a}\cdot c_{i,j}(t,s,a) \le \Delta_{i,j,t} && \forall i,j,t \\ &\sum_{i=1}^{n} \Delta_{i,j,t} \le L_{j,t} && \forall j,t \\ &\Delta_{i,j,t} \in \{0,1\} && \forall i,j,t. \end{aligned} \tag{26}$$

When the resource consumption function $c_{i,j}(t,s,a)$ is binary-valued, the product with the probability $x_{i,t,s,a}$ is guaranteed to lie in $[0,1]$, with values greater than 0 when a resource-consuming action is chosen with any probability. Thus, the first constraint ensures that for any agent policy that has any chance of using resources, resources must be allocated. The second constraint ensures that no more resources are allocated than are available in total. The resulting MILP thus computes policies optimizing the expected value while ensuring all constraints are strictly satisfied.

While the MILP (26) only applies to problems with binary resource consumption, Dolgov and Durfee (2006) show that the approach can be modified to apply to models with arbitrary resource consumption functions. To do so, we start by precomputing a test-function $\mathbb{I}$ over the (ordered) set $C_t$ of *potential* consumption vectors $c_j$:

$$\begin{aligned} C_t &= \{c_j \mid \forall s,a\colon c_j(t,s,a) > 0\}, \\ \mathbb{I}(j,c_k) &= \begin{cases} 1 & \text{if } c_k = C_{t,j}, \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{27}$$

Function $\mathbb{I}(j,c_k)$ checks if resource consumption vector $c_k$ appears at position $j$ in the set $C_t$. This allows us to map the expected consumption of an action to an indicator variable $\Delta_{i,c_k,t}$ per 'consumption level' $c_k$, in the same way as the $\Delta_{i,j,t}$ indicator variables in Eq. 26. Then, we know that $\Delta_{i,c_k,t}\cdot c_k$ resources are required if there is any chance that an action requiring $c_k$ is chosen. We add additional linear variables $u_{i,j,t}$ for the required consumption of agent $i$, resulting in the following addition to the LP (12):

$$\begin{aligned} &\sum_{s\in S_i}\sum_{a\in A_i} x_{i,t,s,a}\cdot \mathbb{I}(k,c(t,s,a)) \le \Delta_{i,c_k,t} && \forall i,c_k,t, \\ &\Delta_{i,c_k,t}\cdot c_{k,j} \ge u_{i,j,t} && \forall i,j,c_k,t, \\ &\sum_{i=1}^{n} u_{i,j,t} \le L_{j,t} && \forall j,t, \\ &\Delta_{i,c_k,t} \in \{0,1\} && \forall i,c_k,t. \end{aligned} \tag{28}$$

With the modification proposed in Eq. 28, strict constraint adherence can be guaranteed for all types of models. However the cost of optimizing the MILP is worst-case exponential in the number of binary variables $\Delta_{i,c_k,t}$, and therefore in the number of agents, the length of the planning horizon, and the number of resource levels. Nevertheless, note that both the CMDP and MILP algorithms are based on the same LP. Therefore we are allowed to mix their constraint models, applying the MILP technique only to those constraints which are indeed hard. As a consequence, we expect this algorithm to be especially well suited to problems which have a small number of hard constraints.

In the stochastic preallocation method based on column generation, the single-agent problems are decomposed by the introduction of $\lambda_{j,t}$ dual costs per resource. For deterministic preallocations, Agrawal et al. (2016) make use of the same principle applied to the (binary-consumption) resource preallocation MILP model. This results in a single-agent MILP where the global constraint has been replaced with a $\lambda_{j,t}$ cost per requested allocation in the objective function:

$$\begin{aligned}
\max_{x_{t,s,a}} & \sum_{t=1}^{h} \sum_{s \in S_i} \sum_{a \in A_i} x_{t,s,a} R_i(s,a) - \sum_{t=1}^{h} \sum_{j} \lambda_{j,t} \Delta^i_{j,t} \\
\text{s.t.} & \sum_{a' \in A_i} x_{t+1,s',a'} = \sum_{s \in S_i} \sum_{a \in A_i} \mathrm{P}_i(s' \mid s,a) x_{t,s,a} && \forall t < h, s' \\
& \sum_{a' \in A_i} x_{1,s',a'} = \mathrm{P}_i(1,s') && \forall s' \\
& \sum_{s \in S_i} \sum_{a \in A_i} x_{t,s,a} \cdot c_j(t,s,a) \leq \Delta^i_{j,t} && \forall j,t \\
& \Delta^i_{j,t} \in \{0,1\} && \forall j,t.
\end{aligned} \tag{29}$$

For a given cost vector $\lambda$ each agent can individually compute its optimal allocation and policy by optimizing Eq. 29. However, depending on the value of $\lambda$, these policies may not satisfy the constraints when combined. Agrawal et al. (2016) propose a greedy rounding scheme to arrive at a joint policy which respects the constraints. The rounding scheme iteratively selects the agent whose policy has the highest expected value that can still be allocated, and allocates the resources to allow the agent to use this policy, until no more policies can be feasibly allocated. In case too many resources would be used by the joint policy computed for $\lambda$, some agents will end up without a policy; these agents instead use their resource-free policy, i.e., the best possible policy subject to constraint $\Delta^i_{j,t} = 0 \ \forall j,t$.

To arrive at the optimal $\lambda$ vector, the authors propose to use a projected subgradient ascent with learning rate $\gamma$,

$$\begin{aligned}
q_{j,t} &= \left( \sum_{i=1}^{n} \Delta^i_{j,t} \right) - L_{j,t} \\
\lambda'_{j,t} &= \lambda_{j,t} + \gamma q_{j,t}.
\end{aligned} \tag{30}$$

This process gradually increases $\lambda_{j,t}$ when the (uncoordinated) consumption due to the current $\lambda_{j,t}$ exceeds the capacity $L_{j,t}$, and vice versa. The learning rate is derived from the estimated distance to optimality, which is given by the distance between the relaxed upper

**Algorithm 3** LDD+GAPS for CMDP $M$ (Agrawal et al., 2016)

$\lambda = \infty$, $\lambda' = 0$
$\forall i\colon \pi_i^0 \leftarrow \text{SOLVEMILP}(M_i, \lambda)$ ▷ Resource-free policies
$\pi^* \leftarrow \pi^0$ ▷ Best joint policy
**while** $\lambda \neq \lambda'$ **do**
  $\lambda \leftarrow \lambda'$
  $\forall i\colon \langle \pi_i, \Delta^i \rangle \leftarrow \text{SOLVEMILP}(M_i, \lambda)$ ▷ Eq. 29
  $\pi' \leftarrow \text{GAPS}(\pi^0, \pi, \Delta)$ ▷ Greedily select feasible policy $\pi'$
  $V^{\text{UP}} \leftarrow \sum_{j,t}(\lambda_{j,t} \cdot L_{j,t}) + \sum_i \big(V_{\pi_i} - \sum_{j,t}(\lambda_{j,t} \cdot \Delta^i_{j,t})\big)$
  $V^{\text{LOW}} \leftarrow V_{\pi'}$
  $\forall j,t\colon q_{j,t} \leftarrow \big(\sum_{i=1}^n \Delta^i_{j,t}\big) - L_{j,t}$
  $\gamma \leftarrow \frac{V^{\text{UP}} - V^{\text{LOW}}}{\|\nabla \mathbf{q}\|^2}$
  $\lambda' \leftarrow \lambda + \gamma \cdot \nabla \mathbf{q}$
  **if** $V_{\pi'} > V_{\pi^*}$ **then** $\pi^* \leftarrow \pi'$ ▷ Keep best feasible policy
**end while**
**return** $\pi^*$

bound given by the values from optimizing individual agents' policies using Eq. 29, and the achieved lower bound given by the value of the joint policy selected by the greedy routine. For $\|\nabla \mathbf{q}\|$ the Euclidean norm over the resource-difference gradient, the learning rate is updated as:

$$\gamma = \frac{V^{\text{UP}} - V^{\text{LOW}}}{\|\nabla \mathbf{q}\|^2}. \tag{31}$$

Algorithm 3 puts all the steps together, presenting the complete approach.

Unfortunately, the single-agent optimization problem of Eq. 29 obtained by decomposing the centralized MILP cannot be solved using dynamic programming: the resource costs incurred when selecting an allocation $\Delta^i_{j,t}$ cannot be attributed to any individual state, but must instead be evaluated with respect to the entire trajectory from the starting state. Therefore the single-agent problem remains of exponential complexity. In addition, the greedy rounding of allocations is only likely to be accurate when the number of agents is relatively large. Nevertheless, because the algorithm splits up planning the single-agent problems from optimizing the global constraint, it obtains similar benefits to Column Generation: not only can the single-agent models be optimized in parallel, but the individual MILP models also have a factor $n$ fewer binary variables, greatly improving its practical complexity.

An alternative linear programming approach for computing a deterministic preallocation has been proposed by Ross and Varadarajan (1989). The main idea is to impose constraints on so-called sample paths, such that all feasible policies violate a resource constraint with probability 0. The method can only be used for instantaneous constraints, and budget constraints cannot be integrated.

### 4.3 Online and Hybrid Methods

Online and hybrid solutions perform planning and resource allocation while decisions are made, rather than computing a solution prior to execution. This can be advantageous

because information that is revealed online can be taken into account in the decisions that are subsequently made. However, this comes at a cost since additional planning and coordination is required during execution.

#### 4.3.1 Online and Hybrid Methods for MDPs

Meuleau et al. (1998) present an online coordination scheme for multiagent planning problems with limited resources. The problem consists of $n$ separate tasks (i.e., agents) which require resources to be completed. The tasks are completely independent, and only the shared resources couple the planning problems corresponding to the tasks. A Markov Task Decomposition (MTD) is proposed which decouples the planning problems using an offline phase and online phase. In the offline phase a value function $V_i(s, t, m)$ is computed for task $i$, which defines the expected reward that is received when starting from state $s$ at time $t$ while $m$ resources are available. This value function can be computed using a simple dynamic programming procedure:

$$V_i(s,t,m) = \max_{a \leq m} \sum_{s' \in S_i} T_i(s,a,s') \left[ R_i(s,a) + V_i(s', t+1, m-a) \right] - c \cdot a, \tag{32}$$

in which the actions $a$ correspond to the resource quantity used and the constant $c$ corresponds to the cost for using one unit of the resource.

The online phase starts with assigning resources $m_i$ to tasks $i$ such that $\sum_i m_i \leq m_g$, in which $m_g$ denotes the total available units of a global resource. Next, the value functions $V_i$ are used to decide which actions can be executed, given the $m_i$ resource units available for the tasks. Finally, the total amount $m_g$ is updated based on the actions that have been executed. The main difficulty of the online procedure is the heuristic allocation of resource units to the tasks. A simple greedy heuristic can be used, based on the increase in expected value when assigning more resources:

$$\Delta V_i(s,t,m) = V_i(s,t,m+1) - V_i(s,t,m), \tag{33}$$

which represents the expected value increase when assigning one additional resource unit to task $i$. Unfortunately this heuristic does not provide any guarantees, but in a simple domain it has shown to be effective.

The coordination scheme based on precomputed value functions and online resource allocation is simple to execute, but it has several practical limitations. For example, actions need to correspond to a resource quantity, and in the online phase only budget constraints can be considered. Instantaneous constraints may be integrated by restricting the resource quantities that are assigned online, but this leads to an overestimation of the expected value because the value functions $V_i$ are computed without this additional restriction.

A more sophisticated budget allocation and reallocation scheme has been proposed by Boutilier and Lu (2016), which follows roughly the same ideas as the concepts used in the method by Meuleau et al. (1998). The planning problem consists of $n$ agents that are allowed to spend total budget $B$. Independent planning problems of the agents are considered separately, and after solving these problems an additional online procedure is used to derive a joint action to execute, given the remaining budget and the joint state of the agents. The resource limit $B$ should be satisfied only in expectation. This is a minor

difference compared to the method described above, which treats the resource constraint as a hard constraint that should not be violated. Another difference is that discounting is used for both the reward in the objective function and the cost in the budget constraint.

A value function $V_t^i(s, b)$ is used to represent the expected discounted reward that an individual agent $i$ receives when starting from state $s$ while the remaining budget is $b$. This value function takes the following form:

$$\begin{aligned} V_t^i(s,b) = \max_{a \in A} & R_i(s,a) + \gamma \sum_{s' \in S} T_i(s,a,s') V_{t+1}^i(s', b_{s'}) \\ \text{s.t. } & c_i(s,a) + \gamma \sum_{s' \in S} T_i(s,a,s') b_{s'} \leq b, \end{aligned} \tag{34}$$

in which the objective function represents the expected discounted reward, and the constraint defines that the expected discounted cost is upper bounded by $b$. The term $b_{s'}$ denotes the remaining budget after executing action $a$ in state $s$ with budget $b$. It can be shown that the value function $V_t$ has a finite number of *useful* budget levels to consider. A budget $b$ is called useful if and only if $V_t(s, b) > V_t(s, b - \varepsilon)$ for all $\varepsilon > 0$. Another convenient property is that the value function is piecewise linear and monotone non-decreasing in budget $b$. Both properties turn out to be crucial in the construction of the budget allocation algorithm.

The complexity of computing an optimal value function depends on the number of useful budget levels. By exploiting the piecewise linearity and the finite number of useful budget levels the value function $V_t(s, b)$ can be approximated using a value function $\tilde{V}_t(s, b)$. This approximation is derived using a procedure that prunes budget levels, which we do not further discuss here. Besides computing the expected value, it is also possible to keep track of the resource usage variance of the corresponding policy. This variance can be useful to know since the method treats the resource constraint as a soft constraint.

Given the value functions for time step $t$, resource allocation is done using a budget-constrained optimization problem defined over the finite number of useful budget levels, indexed by $k$:

$$\begin{aligned} \max_{x_{i,k}} & \sum_i \sum_k V_t^i(s_i, \beta_{i,k}) \cdot x_{i,k} \\ \text{s.t. } & \sum_i \sum_k \beta_{i,k} \cdot x_{i,k} \leq B \\ & \sum_k x_{i,k} = 1 \qquad \forall i \\ & x_{i,k} \in \{0,1\} \qquad \forall i,k. \end{aligned} \tag{35}$$

In this formulation the binary variable $x_{i,k}$ indicates that agent $i$ gets budget $\beta_{i,k}$ assigned at time step $t$. Given the structure of the mixed-integer linear program, it turns out that an optimal solution can be obtained in polynomial time using a simple greedy algorithm. The budget allocation is performed in each time step, which means that budget is reallocated online depending on the outcome of the uncertainty. This requires online communication for solving the budget allocation problem shown above, to obtain the joint state $s = \langle s_1, \ldots, s_n \rangle$ of the agents. In some sense the online budget allocation phase is similar to the procedure proposed by Meuleau et al. (1998). A key difference, however, is that the budget is allocated by solving an optimization problem, rather than assigning the resources using a heuristic.

Both aforementioned online approaches have in common that value functions are precomputed, and these value functions are utilized in the online phase to choose resource-feasible actions. De Nijs et al. (2015) follow a different approach and present an online planning algorithm in which both value function computation and coordination take place in an online fashion. It is assumed that the resource-constrained planning problem of the agents can be decoupled at agent level, identical to the problem setting with independent agents that we sketched in Section 3. Hard resource constraints are enforced by a so-called arbiter which may override the intended actions of the agents in case execution of the intended joint action leads to overconsumption of resources. Agents iteratively compute new policies that are a best response to the decisions made by the arbiter and, implicitly, a best response to the intended actions of the other agents. Below we describe the algorithm in more detail.

The arbiter performs arbitrage in case the policies $\pi_1, \ldots, \pi_n$ of the agents prescribe actions which lead to a violation of at least one resource constraint. Intuitively, the arbiter aims to maximize the future expected reward of the agents while making sure that the current action execution is feasible. This arbitrage step can be formalized using the following mixed-integer linear program:

$$\begin{aligned} \text{ARBITRAGE}(\pi, t, s) = \underset{x}{\text{argmax}} & \sum_{i=1}^{n} \sum_{k=1}^{|A_i|} x_{i,k} \cdot Q_i(t, s_i, a_{i,j}) \\ \text{s.t.} & \sum_{i=1}^{n} \sum_{k=1}^{|A_i|} x_{i,k} \cdot c_{i,j}(t, s_i, a_{i,k}) \leq L_{j,t} \\ & \sum_{k=1}^{|A_i|} x_{i,k} = 1 \qquad \forall i \\ & x_{i,k} \in \{0, 1\} \qquad \forall i, k, \end{aligned} \tag{36}$$

in which the binary variable $x_{i,k}$ indicates whether the $k$'th action of agent $i$ has been selected, and $Q_i$ denotes the $Q$-function corresponding to policy $\pi_i$ of agent $i$. The arbitrage step assumes that resource conflicts can always be resolved using arbitrage. This is natural in several domains, such as temporarily disabling electric heating in a house at the cost of minor discomfort. Since arbitrage takes place online, Eq. 36 must always be solved in the span of a single time step. Because this involves solving a mixed-integer linear program, this may be too computationally demanding. However, it is important to note that the structure allows for an $O(n)$ solution if there is only one resource. Furthermore, suboptimal arbitrage decisions can be made by considering a rounding scheme based on the linear relaxation of the problem.

Agents use the arbitrage scheme to compute a best-response policy. The policies of the agents are simulated multiple times using Monte Carlo trials, during which the agents estimate the probability that an action $a$ becomes $a'$ due to arbitrage. More formally, the agents compute probabilities $P^{\text{ARBI}}(a' \mid \pi_i, s_i, a)$, which represents the probability that the arbiter decides that action $a'$ should be executed while the current policy $\pi_i$ prescribes that action $a$ should be taken in the current state $s_i$. The probability estimates can be used to

compute a best response as follows:

$$\begin{aligned} V_{\pi_i}^{\text{ARBI}}(h+1,\cdot) &= 0, \\ Q_{\pi_i}^{\text{ARBI}}(t,s_i,a) &= R_i(t,s_i,a) + \sum_{s_i' \in S_i} P(s_i' \mid s_i,a) V_{\pi_i}^{\text{ARBI}}(t+1,s_i'), \\ V_{\pi_i}^{\text{ARBI}}(t,s_i) &= \max_{a \in A_i} \sum_{a' \in A_i} P^{\text{ARBI}}(a' \mid \pi, s_i, a) Q_{\pi_i}^{\text{ARBI}}(t,s_i,a'). \end{aligned} \tag{37}$$

This variant of value iteration explicitly considers the probability estimates regarding actions that changed due to arbitrage. The computation of best-response policies based on arbitrage is repeated multiple times until convergence. When this procedure converges, the agents cannot improve their own expected utility by changing their policy individually. No formal guarantees have been provided regarding the best-response strategy and the equilibrium it reaches. This means that the decisions may converge to a local optimum that is much worse compared to a potentially optimal set of decisions. However, the algorithm has shown to be effective in a power-constrained heating problem with multiple heating devices which are tasked to maintain the temperature in a room.

#### 4.3.2 Online and Hybrid Methods for POMDPs

Undurti and How (2010) have presented an online search algorithm which can be used to choose actions in such a way that reward is maximized while avoiding forbidden states. The planning problem can be modeled using a constraint penalty function $C(s)$ which defines the penalties associated with forbidden states:

$$C(s) = \begin{cases} 1 & \text{state } s \text{ is forbidden,} \\ 0 & \text{otherwise.} \end{cases} \tag{38}$$

Based on this function, the constrained planning problem is defined using the following constraint:

$$E\left[\sum_{t=1}^{h} C(s_t)\right] \leq 0. \tag{39}$$

In case forbidden states cannot be left, the expectation in Eq. 39 defines the probability of reaching a forbidden state, which should be upper bounded by 0. The proposed search algorithm relies on two stages. The online stage finds a reward-maximizing action for a given time horizon while avoiding forbidden states. In order to guarantee feasibility of the constraint beyond the time horizon, the algorithm uses a conservative baseline policy which has been precomputed in an offline stage. Below we discuss both stages in more detail.

In the offline stage point-based value iteration is used to obtain a policy which minimizes the sum of constraint penalties:

$$\min_{\pi} E\left[\sum_{t=1}^{h} C(s_t) \;\middle|\; \pi\right], \tag{40}$$

in which $s_t$ denotes the state at time $t$. Since the policy minimizes the constraint penalties incurred, the policy can be used to avoid forbidden states as best as possible. However,

reward is not considered so executing this policy is unlikely to lead to high reward. Given a current belief $b$, an online tree search algorithm is used to find a reward-maximizing action up to a given search depth $d$. Beyond this depth the minimum penalty defined by the offline policy is used to check constraint feasibility.

The proposed search algorithm is conceptually simple but it has several practical and fundamental limitations. Although the current algorithm only considers problems with forbidden states, it can be expected that the approach can be easily generalized to budget constraints. However, instantaneous constraints appear to be more challenging due to the behavior of this constraint beyond the time horizon considered by the search tree. Another limitation is that the algorithm only supports one cost function, and the search algorithm does not easily generalize to settings with multiple cost functions. Finally, in domains where minor violations are allowed the offline policy may be too conservative since it minimizes the cost incurred beyond the planning horizon.

The tree search algorithm proposed by Lee et al. (2018) exploits two further insights that have been introduced earlier in this survey. First, it is observed that solving a constrained planning problems boils down to finding a $\lambda$ such that a policy computed with a modified reward function $R - \lambda \cdot C$ satisfies the constraint. We have seen this insight before in Eq. 15 and in the description of the column generation method in Algorithm 1. Second, it is observed that a POMDP can be seen as a belief-state MDP, similar to the linear programming formulation introduced in Eq. 23. Both insights together lead to an iterative procedure to optimize for $\lambda$ while considering a constraint. The procedure solves a belief-state MDP based on the modified reward function $R - \lambda \cdot C$ and a given $\lambda$. After that, it executes policy evaluation to obtain the expected cost $V_C^{\pi^*_\lambda}$ of the policy that was found using the belief-state MDP. Finally, it updates the $\lambda$ parameter based on the difference between the resource limit $L$ and the current expected cost $V_C^{\pi^*_\lambda}$. If both the solving and evaluation step are performed optimally and if an appropriate step size $\alpha$ is set, then this procedure is guaranteed to converge to an optimal solution.

Unfortunately it is intractable to solve the belief-state MDP optimally. Therefore, Lee et al. (2018) propose an approach based on the Monte Carlo tree search algorithm POMCP (Silver & Veness, 2010). The POMCP algorithm has shown to be very effective for large POMDPs, but the original version does not take additional constraints into account. Lee et al. (2018) show how POMCP can be adapted such that constraints are considered, and they show how POMCP can be interleaved with the $\lambda$ update scheme that we described above. For more details regarding the POMCP algorithm, the required adaptations and the parameter $\tau$ we refer to the respective papers.

Compared to the tree search algorithm by Undurti and How (2010) it can be expected that POMCP for Constrained POMDPs provides better performance because of the more advanced action selection heuristics that are used during the search. Additionally, the algorithm includes a more advanced belief estimation data structure based on state particles, which is expected to improve solution quality as well. A disadvantage of POMCP for Constrained POMDPs is that expected resource consumption is estimated during the tree search, which means that it cannot be strictly guaranteed that the soft constraint is respected while executing the actions. In practice, however, it can be expected that the realization of resource consumption is close to the expectation.

| § | Reference | Conn. | Strictness | PO | Timespan | | | Impl. |
|---|---|---|---|---|---|---|---|---|
| | | | | | Budget | Inst. | Multi | |
| 4.1.1 | Altman (1999) | | soft | | ✓ | ✓ | $m$ | ✓ |
| 4.1.1 | Yost and Washburn (2000) | | soft | ✓ | ✓ | ✓ | $m$ | ✓ |
| 4.1.2 | Isom et al. (2008) | | soft | ✓ | ✓ | | 1 | |
| 4.1.2 | Kim et al. (2011) | | soft | ✓ | ✓ | | $m$ | |
| 4.1.2 | Poupart et al. (2015) | | soft | ✓ | ✓ | | $m$ | ✓ |
| 4.1.2 | Walraven and Spaan (2018) | | soft | ✓ | ✓ | ✓ | $m$ | ✓ |
| 4.1.3 | Dolgov and Durfee (2003) | | soft w. bound | | ✓ | | $m$ | |
| 4.1.3 | De Nijs et al. (2017) | | soft w. bound | | ✓ | ✓ | $m$ | ✓ |
| 4.2 | Wu and Durfee (2010) | | hard | | ✓ | ✓ | $m$ | ✓ |
| 4.2 | Agrawal et al. (2016) | | hard | | ✓ | | $m$ | |
| 4.3.1 | Meuleau et al. (1998) | ✓ | hard | | ✓ | ✓ | $m$ | |
| 4.3.1 | Boutilier and Lu (2016) | ✓ | soft | | ✓ | | 1 | |
| 4.3.1 | De Nijs et al. (2015) | ✓ | hard | | | ✓ | $h$ | |
| 4.3.2 | Undurti and How (2010) | ✓ | hard | ✓ | ✓ | | 1 | |
| 4.3.2 | Lee et al. (2018) | ✓ | soft | ✓ | ✓ | | $m$ | ✓ |

Table 3: Overview of algorithms for constrained multiagent Markov decision problems. Columns indicate whether the algorithm requires continuous communication (Conn.), whether it is able to plan for models with partially observable state dynamics (PO), whether it deals with (a) hard or soft constraint(s), whether constraints are active across the entire horizon (budget), individual time steps (Inst.), and/or whether multiple constraints are allowed (Multi). The final column indicates whether an implementation is available in our open-source Constrained Planning Toolbox.

### 4.4 Reflections on the State of the Art

We are now ready to reflect on the state of the art. In Table 3 we provide an overview of the algorithms described in this section, organized along the taxonomy introduced earlier. Furthermore, to aid in the reproducibility of past results and algorithms, we implemented a significant portion of them in an open-source Constrained Planning Toolbox, which may be found on-line.[2] The final column of Table 3 indicates whether the algorithm is present in the toolbox at the time of writing the article.

We observe that soft, in-expectation constraints have received the majority of attention. The expectation is a natural objective for decision making under uncertainty, because it aggregates over outcomes, making it easy to compute and analyse. Soft-constrained algorithms typically also use the state-occupancy-based linear programs as a base. A major advantage of linear programming formulations is that they allow for an essentially unbounded amount of constraints to be considered. However, these methods also come with significant downsides: firstly, by casting the problem as an LP, we ignore the computational benefits that the recurrent structure of the Bellman equation can give us, in the form of the dynamic

2. The toolbox and documentation can be found in the following repository: `https://github.com/AlgTUDelft/ConstrainedPlanningToolbox`.

programming paradigm. Secondly, by representing the resources outside of the MDP state space, the agents cannot condition their decisions on past usage in the case of budget constraints. Finally, soft constraints on their own offer no guarantees on the magnitude or frequency with which a constraint might be exceeded.

Each of these downsides is individually addressed by some of the surveyed works. The first issue can be avoided by decoupling the constraints from the planning problems through methods such as Lagrangian relaxation (Yost & Washburn, 2000). Boutilier and Lu (2016) show that recourse can be implemented by integrating an online recourse step with a method that computes a policy for every possible budget level. And finally, a few works have applied bounds on the tail probability of resource consumption (Dolgov & Durfee, 2003; De Nijs et al., 2017), to increase the safety of the solution.

Whereas soft constraints can be satisfied in polynomial time, hard constraints do not allow for efficient solutions. This is reflected by the surveyed works that cover hard constraints, which either approximate the solution, or try to cover special cases (or both). As such, these methods are more likely to impose limitations on the number or type of constraints. An interesting observation is that not many hard-constrained works consider instantaneous constraints; this is surprising because instantaneous constraints are in some sense the simplest type of situation to study multiagent coordination, because there is no history to consider.

Hard constraints prove especially challenging for algorithms that also handle partial observability. We speculate that this is an especially difficult problem to solve: if the resource consumption is state-dependent, a constraint can only be guaranteed to be met if there is absolute certainty (in the belief probability sense) that an agent's state is not one where (too many) resources would be used. Only the work by Undurti and How (2010) considers constraints that can be seen as hard constraints, but the application of their current algorithm is limited to domains with forbidden states. More general hard budget constraints and hard instantaneous constraints have not been considered. Further, regarding partial observability, it is important to mention that most existing Constrained POMDP algorithms include discounting in the constraints, which is an additional assumption that is not always appropriate in realistic domains.

A final reflection on the discussed contributions is that these (obviously) concentrate on the coordination of the use of shared resources. Some of the online/hybrid methods are completely centralized algorithms for both planning and coordination of the resources. These leave only execution to the agents. Other potential dependencies between the agents would then need to be included in this centralized algorithm, which may swiftly lead to fully-dependent multiagent MDPs. The methods that use pre-allocation however, return autonomy to the agents to plan and execute their actions as long as they stick to this agreed allocation. An important advantage of these approaches is that they could rather straightforwardly be combined with other methods for coordinating agent interactions. This in fact also applies to decentralized online methods where agents plan independently, and then are forced to (learn to) coordinate when accessing the shared resources (e.g., by a so-called arbiter, De Nijs et al., 2015). Combining algorithms for CMMDPs with other methods for agent coordination seems to be an otherwise completely unexplored territory.

## 5. Extensions and Variants

The planning problems and algorithms discussed in this survey involve multiple independent agents which operate in an uncertain environment with resource constraints. This is a rich problem class with several applications, and as a result several closely related variants have been proposed. In this section we discuss the following variants: 1) instead of sharing resources, agents need to allocate tasks, 2) agents may have to deal with constraints on the probability of violating a limit (chance constraints), 3) agents are still learning about the environment while meeting constraints (safe reinforcement learning), or 4) the limit may be uncertain itself. Such variants are not supported by the algorithms that we described in the previous section. In addition to these, we provide below a brief overview of alternative planning problem models that can be (ab)used to deal with resource constraints, such as 1) multi-objective problems, where instead of modeling a resource limitation as a constraint, the respective resource use is seen as an objective to minimize, 2) congested multiagent systems where resource costs increase with resource use, and 3) optimal decentralized policy computation, which allows expression of other interactions between agents than just resource use.

**Task Allocation Problems** In task allocation problems, agents are required to complete a set of tasks, usually with the objective to achieve all tasks as quickly as possible (a task could for example be 'pick up a package and deliver it'). Tasks are similar to resources, in that we want mutual exclusivity to hold, meaning that each task is completed exactly once. Basic exclusivity between agents can be implemented through budget constraints: for every task, add one budget constraint with one unit of availability. Any agent that starts on a given task consumes the resource. However, sometimes task models may specify additional features such as the possibility of task failure, or the existence of precedence constraints requiring (one or more) tasks to be completed before the current task can be started (Beynier & Mouaddib, 2006, 2011). Such cases may be covered by careful use of resource production (for example on all transitions where a task fails, allowing it to be re-attempted). Nevertheless, it is often easier to model tasks into the coordination problem directly, as shown by Agrawal et al. (2016), who propose an efficient Lagrangian relaxation of a multiagent problem that includes both tasks and resources including precedence constraints.

**Chance Constraints** Chance constraints arise in domains in which a notion of risk should be bounded. In such domains it is important to bound or minimize the probability that an undesirable event occurs, such as reaching a dangerous state or exceeding the capacity of a resource. Problem domains where such constraints occur include the dynamic allocation of radio spectrum while minimizing the risk of collisions (Tehrani et al., 2012), as well as planning autonomous planetary exploration (Santana et al., 2016b). In the conceptualization considered in this survey we can model a chance constraint for a single agent by defining a resource consumption function that equals 1 when the undesirable event occurs, and equals 0 otherwise. The expected resource consumption becomes equal to the probability that the event occurs, and the resource limit represents an upper bound on this probability. The mapping from chance constraints to a resource constraint in our conceptualization is straightforward, but there are two major limitations associated with this approach. First, the constraint only represents a valid chance constraint if policy execution terminates when the

undesirable event occurs. This is a major limitation, because problem domains in which such events do not cause termination cannot be considered. Second, chance constraints are not additive (Ono et al., 2015), which means that the existence of a chance constraint couples the planning problems of multiple agents. Tailored algorithms for dealing with chance constraints have been developed, such as heuristic search for chance-constrained POMDPs (Santana et al., 2016a). Alternatively, in case the chance constraints apply to a finite-horizon POMDP, they may be converted into regular constrained POMDP models through a polynomial-time reduction (Khonji et al., 2019), after which the algorithms surveyed here may be applied.

**Safe Reinforcement Learning** The methods described in this work all require that the models of the agents are fully specified (i.e., the transition and reward functions are given as input). However, it may not always be possible to (fully) specify the models in advance, which requires algorithms that learn the dynamics of a system through interactions. This is called reinforcement learning. In standard reinforcement learning an agent is able to explore the environment freely without any additional constraints. However, in domains with limited resources such freedom may lead to violations of constraints during exploration, highlighting the need for *safe* exploration in reinforcement learning (García & Fernández, 2015; Ray et al., 2019). Achiam, Held, Tamar, and Abbeel (2017) made the connection between safe reinforcement learning and CMDPs by observing that notions of safety can be encoded in terms of the constraints, leading to a line of work on constrained policy optimization (Tessler et al., 2019; Yang et al., 2020). In general, constrained reinforcement learning has received a large amount of attention in recent years, for instance focusing on safety during training (Chow et al., 2018; Simão et al., 2021), online actor-critic approaches (Yu et al., 2019; Yang et al., 2021), or learning with safety functions that are unknown a priori (Wachi & Sui, 2020). Safe reinforcement learning in multiagent systems remains under-explored, however.

In Bayesian reinforcement learning, uncertainty over model parameters is explicitly tracked, making this framework well-suited for designing safe learning algorithms; see the survey by Ghavamzadeh, Mannor, Pineau, and Tamar (2015, Section 6) for an overview. Bayesian reinforcement learning also provides the most straightforward connection between reinforcement learning and planning, because the problem of optimizing a learning policy in this setting is equivalent to planning in a POMDP model (Duff, 2002). Several works have looked at solving constrained reinforcement learning problems through the Constrained POMDP lens: Kim, Kim, and Poupart (2012), and later Lee, Jang, Poupart, and Kim (2017) look at scaling up the single-agent case by using approximate back-ups and belief-state merging, respectively. In the multiagent case, Castañon (1997) studies a sensor management problem where each agent represents an object to be classified. De Nijs, Theocharous, Vlassis, De Weerdt, and Spaan (2018b) scale up the multiagent constrained learning setting to a large tourist recommendation problem, by truncating the planning horizon when the remaining risk falls below the approximation threshold.

**Uncertainty in Constraints** In constrained settings we may also encounter domains where the constraint models are not fully specified. This may happen when actions take an uncertain amount of time (Bresina et al., 2002), or because the constraint itself depends on a quantity which cannot be predicted with perfect accuracy, such as wind speeds affecting available power from wind farms. Compared to the safe reinforcement learning setting, relatively few works have tackled uncertainty in constraints. Stochastic resource consumption

has been studied by Mausam, Benazera, Brafman, Meuleau, and Hansen (2005). In this line of work the resource quantity available is assumed to be known, but there is uncertainty regarding the consumption of a resource when executing an action, in the context of a planning for a planetary rover with budget constraints. De Nijs, Spaan, and De Weerdt (2018a) proposed two methods to integrate stochastic resource constraints in algorithms for computing deterministic and stochastic preallocations.

Conversely to the above problem variants, which incited adaptations of the presented algorithms for CMMDP, algorithms for the following variants may inspire novel ways to solve CMMDPs.

**Multi-objective Problems** Multi-objective planning under uncertainty (Roijers et al., 2013, 2018) deals with the computation of decision-making policies while considering multiple objective functions, rather than just an individual reward signal. It is clear that there is a strong connection between constrained planning and planning for multiple objectives. The resource usage corresponding to each resource constraint can be seen as a separate objective that needs to be optimized. In the context of constrained planning we have made a similar observation when we introduced column generation for computing stochastic preallocations (Section 4.1.1). In this algorithm the modified reward function takes the form $R-\lambda \cdot C$, which is a linear combination of the reward objective $R$ and the resource consumption objective $C$, and $\lambda$ is a pricing parameter iteratively determined by the algorithm. The challenge in applying insights from multi-objective planning methods to constrained planning lies exactly in choosing the values for such a $\lambda$ parameter (i.e., weight vector) in order to satisfy the resource constraint(s), since these methods either assume fixed objective weights, or they compute solutions for all possible assignments of weights to objectives.

**Congested Systems** There is a close link between constrained multiagent systems and congested multiagent systems. While a constraint specifies some threshold that should be satisfied at all times, it allows the system to operate freely within the constrained space. Congested systems, on the other hand, do not impose specific limits, but instead assign higher costs to states with higher resource usage. While this survey focuses on constrained systems, we mention two state-of-the-art approaches for congested systems as a source of inspiration. Kumar, Varakantham, and Kumar (2017) study models of congested systems where the reward of the system is submodular (decreasing reward growth with increased resource use) in the number of agents participating in an action. They explore greedy and lazy greedy approaches, showing that practical performance far exceeds the theoretical guarantee of 50 percent of optimality. He, Wallace, Gange, Liebman, and Wilson (2018) instead model congestion by a quadratic optimization problem, simultaneously varying the resource price and control decisions. They apply the Frank-Wolfe convex optimization algorithm to find a probabilistic schedule optimizing congestion prices in constant time. One effective way to model resource constraints by congestion is a queue where agents are waiting until a resource becomes available. A practical illustration here is the coordination of the use of charging stations for electric vehicles (De Weerdt et al., 2016).

**Optimal Decentralized Policies** The multiagent planning model considered in this survey assumes that independent agents are coupled by one or more resource constraints. Preallocation methods assign resources to agents in such a way that agents can execute their

policies in a decentralized way without any communication during execution. Decentralized MDPs (Bernstein et al., 2002) and Decentralized POMDPs (Oliehoek & Amato, 2016) are based on a similar notion of decentralization, in which multiple agents collaborate in a decentralized way in order to reach a common goal. The key difference, however, is that the common goal is typically defined using a joint reward function that is conditioned on joint states. This means that specific combinations of agent states can be rewarded, which cannot be expressed in the model considered in this survey. The increased expressiveness of the model comes at a cost, because solving Decentralized POMDPs is computationally very demanding and it is limited to small instances. An independence assumption and additive rewards restrict the application of the model, but result in more efficient algorithms (Scharpff et al., 2016).

Decentralized POMDPs have been extended with resource constraints in work by Wu, Jennings, and Chen (2012). They propose an exact dynamic programming algorithm and a policy iteration approach that considers the resource constraints during planning. In its current form the algorithm only supports constraints on budget. As discussed above, Beynier and Mouaddib (2006, 2011) propose a task-oriented Dec-MDP variation which models limited resources that are available to agents, but these resources are not shared across agents. Finally, to overcome such limitations, an under-explored direction is to model the usage of shared resources as influence sources (Oliehoek et al., 2012).

## 6. Research Directions and Algorithmic Challenges

Considering the discussed contributions on constrained multiagent Markov decision problems and related problems and approaches, we next provide an overview of avenues of work that can be investigated in the future.

**Creating Challenging Benchmark Problems** Constrained multiagent Markov decision problems share a common underlying modeling paradigm, described in Section 3. Because of this, most algorithms in this survey have been compared against one another in some form. Nevertheless, we expect that future research in this area would strongly benefit from a common set of challenging and relevant benchmark problems.

Openly accessible benchmark problems would provide the typical benefit of making different ideas more directly comparable. Additionally, having access to realistic or ideally real-world problems adds further credibility to the field. However, artificial benchmark problem instances could also be of use, for example to identify specifically challenging or successful conditions. Creating problems with meaningful uncertainty is generally difficult, which is demonstrated by the first probabilistic International Planning Competition: a deterministic replanner still achieved the best performance on most problem instances with noise added to originally deterministic action effects (Younes et al., 2005). At the same time, Pisinger (2005) showed that knapsack problems require specific correlations in weight and value to be hard to solve. Because constrained multiagent decision-making problems have both uncertainty and knapsack-like constraints, producing instances that are hard in both dimensions is likely a challenging research topic in its own right. Furthermore, a careful design of benchmark domains will help to ensure wider applicability of algorithm implementations, by including domains with currently understudied cases, such as continuous-valued resource consumptions, actions that produce resources, and inverted ($\geq L$) constraints.

**Handling Partial Observability** Existing work on constrained planning mostly considers models with full observability of environment states. There are a few algorithms that explicitly take partial observability into account, as described in Sections 4.1.2 and 4.3.2, but there are a few aspects which currently limit the applicability of most of these algorithms. Most existing algorithms for Constrained POMDPs only support budget constraints which include a discount factor in the definition of the constraint. This means that many types of constraints that arise in domains cannot be modeled. Furthermore, the majority of the existing algorithms consider soft constraints that need to be satisfied in expectation. Approaches to enforce hard constraints and approaches for providing guarantees in the MDP setting may be applied in the POMDP setting as well, but typically this leads to scalability problems. Integrating multiple types of constraints and hard resource limits requires additional work.

**Using Non-Robust Communication Channels** In our taxonomy mapping problem domain properties to solution approaches (Table 2), we identified 'disconnected' domains: domains where agents cannot expect to be able to communicate during policy execution. This restriction limits us to the preallocation algorithms described in Sections 4.1.1 and 4.2. These algorithms compute fixed resource allocations, and therefore come at the price of not being able to adapt to realizations of uncertain state trajectories.

However, in many domains, non-robust communication options may be available. For example, in the flexible load control setting of Section 2.1.1, it is undesirable for solutions to depend on communication between loads, because power supply issues could take down communication exactly at the time coordination is most critical. Currently only preallocation algorithms are safe for this setting, but there is room for new algorithms that exploit communication when it is available to better handle uncertainty. In future work it can be investigated whether communication aspects can be explicitly considered in the planning algorithms. In the unconstrained setting, Spaan, Oliehoek, and Vlassis (2008) look at integrating communication delays in Decentralized POMDPs, but this line of work has not yet been extended to general MDPs or POMDPs coupled through resource constraints.

**Balancing Offline and Online Computation** In Section 4 we described both online and offline coordination algorithms that can be used in domains with resource constraints. These algorithms have mostly been studied in isolation, making them either fully online or fully offline. An important question is whether both types of approaches can strengthen each other. In some domains, the online decision window may be too short to allow sufficient deliberation time, whereas offline policy computation can take significantly longer. In such a setting, offline preallocation-based policies may serve as heuristic in online coordination, such that the online decisions can improve upon the decisions that would be made in the offline case. A few online algorithms include some form of local policy computation prior to online coordination, but typically these policies disregard the decisions of other agents. In future work it can be investigated whether hybrid approaches can be developed, combining the strengths of both online and offline methods.

**Decoupling Level** The multiagent planning model considered in this survey assumes that the planning problem is decoupled at the level of individual agents. This means that the model does not allow for modeling of dependencies between the planning problems of the agents, other than global resource constraints. In practice the assumption of agent-level

decoupling may be too extreme and it may restrict the potential for application (Oliehoek et al., 2015). As an alternative to decouplings at the level of agents one can think about clustering based on groups of agents, such that planning problems for groups of agents can be defined rather than individual agents. Choosing the right clusters creates a new optimization problem that decides at what level the agents are clustered into groups. This direction of research may be expanded based on influences between agents (Witwicki & Durfee, 2010; Scharpff et al., 2016).

When state transitions are fully coupled, we may have no alternative other than optimizing the agents jointly. However, even if we can compute an optimal stochastic policy, the resulting joint actions cannot be executed without communication: if we give these joint actions to individual agents to execute, they may individually randomize their actions to arrive at a joint action that was not intended. This can be avoided by letting the agents communicate during action selection. Paruchuri, Tambe, Ordonez, and Kraus (2004) show that it is possible to obtain policies that coordinate on soft resource constraints by reasoning explicitly about inter-agent communication. They propose splitting the joint action selection in a coupled CMDP into an ordered sequence of individual actions with communication after-states. This way, an agent can decide as part of the planning for which action selections it needs to communicate.

## 7. Conclusions

Multiagent planning problems in domains such as smart energy management, vehicle routing, and conditional maintenance are effectively modeled and solved as constrained multiagent Markov decision processes. Because of their wide applicability, many different solution approaches for these problems exist. In this survey we first identify which domain properties dictate whether or not a solution approach applies to it, namely: whether constraints are soft or hard, whether agents are continuously connected, whether the domain is fully observable, and whether a constraint is on a capacity (instantaneous) or on a budget, on a single resource or on multiple. We then use these domain properties to define a taxonomy of the solution approaches. By positioning existing algorithms in this taxonomy, a number of open problems, understudied research areas and promising directions that can be followed to extend the research field in the future are identified.

Firstly, while there are many algorithms that efficiently cope with soft constraints, including a few that even provide guarantees on the number or probability of constraint violations, there are fewer methods for dealing with hard constraints. Especially regarding hard instantaneous constraints this is surprising, because this is a relevant variant and arguably the most simple one. Almost no work has been done on methods for dealing with hard constraints when the state of the domain is only partially observable. This, however, is less surprising, because to be successful in such domains online and continuous communication seems essential, which significantly complicates the required algorithms. A second challenge regarding mainly the algorithms for partially observable domains is to drop the discounting of future rewards from the model as this is only realistic in few domains. Thirdly, we have identified that the pre-allocation methods and a few of the (non-centralized) online methods are suited for extending with other inter-agent coordination issues, but that this is rather unexplored territory.

More generally, a number of the insights from algorithms for constrained multiagent Markov decision processes (CMMDPs) can inspire algorithms for domains involving multiagent task allocation, chance constraints, safe reinforcement learning, and, an especially understudied variant, domains where there is uncertainty in constraint limits. Conversely, novel CMMDP approaches could be derived from algorithms that deal with multiple objectives, congested multiagent systems, or decentralized (PO)MDPs.

As a final set of directions for future work we identified the benefit of good benchmark problems, the understudied challenge of dealing with non-robust communication, balancing online and offline computations, and novel ways of decoupling or clustering agent computations.

## Acknowledgments

Authors Frits de Nijs and Erwin Walraven contributed equally to this article. This work is funded by distribution system operator Alliander, and by the Netherlands Organisation for Scientific Research (NWO), as part of the Uncertainty Reduction in Smart Energy Systems program.

## References

Achiam, J., Held, D., Tamar, A., & Abbeel, P. (2017). Constrained Policy Optimization. In *Proceedings of the International Conference on Machine Learning*.

Adelman, D., & Mersereau, A. J. (2008). Relaxations of weakly coupled stochastic dynamic programs. *Operations Research*, *56*(3), 712–727.

Agrawal, P., Varakantham, P., & Yeoh, W. (2016). Scalable greedy algorithms for task/resource constrained multi-agent stochastic planning. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Altman, E. (1999). *Constrained Markov Decision Processes*. CRC Press.

Becker, R., Zilberstein, S., Lesser, V., & Goldman, C. V. (2004). Solving transition independent decentralized Markov decision processes. *Journal of Artificial Intelligence Research*, *22*, 423–455.

Bellman, R. E. (1957). A Markovian decision process. *Journal of Mathematics and Mechanics*, *6*(5), 679–684.

Bernstein, D. S., Givan, R., Immerman, N., & Zilberstein, S. (2002). The complexity of decentralized control of Markov decision processes. *Mathematics of operations research*, *27*(4), 819–840.

Beutler, F. J., & Ross, K. W. (1985). Optimal policies for controlled Markov chains with a constraint. *Journal of Mathematical Analysis and Applications*, *112*(1), 236–252.

Beynier, A., & Mouaddib, A.-I. (2006). An iterative algorithm for solving constrained decentralized Markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Beynier, A., & Mouaddib, A.-I. (2011). Solving efficiently decentralized MDPs with temporal and resource constraints. *Autonomous Agents and Multi-Agent Systems*, *23*, 486–539.

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

Boutilier, C., & Lu, T. (2016). Budget Allocation using Weakly Coupled, Constrained Markov Decision Processes. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.

Bresina, J., Dearden, R., Meuleau, N., Ramakrishnan, S., Smith, D., & Washington, R. (2002). Planning under continuous time and resource uncertainty: A challenge for AI. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.

Byon, E., & Ding, Y. (2010). Season-dependent condition-based maintenance for a wind turbine using a partially observed Markov decision process. *IEEE Transactions on Power Systems*, *25*(4), 1823–1834.

Cassandra, A., Littman, M. L., & Zhang, N. L. (1997). Incremental Pruning: A Simple, Fast, Exact Method for Partially Observable Markov Decision Processes. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.

Castañon, D. A. (1997). Approximate dynamic programming for sensor management. In *Proceedings of the IEEE Conference on Decision and Control*, Vol. 5.

Chow, Y., Nachum, O., Duéñez-Guzmán, E. A., & Ghavamzadeh, M. (2018). A Lyapunov-based approach to safe reinforcement learning. In *Advances in Neural Information Processing Systems*.

Claes, D., Oliehoek, F., Baier, H., & Tuyls, K. (2017). Decentralised online planning for multi-robot warehouse commissioning. In *Proceedings of the Conference on Autonomous Agents and Multiagent Systems*.

De Nijs, F., Spaan, M. T. J., & De Weerdt, M. M. (2015). Best-Response Planning of Thermostatically Controlled Loads under Power Constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

De Nijs, F., Spaan, M. T. J., & De Weerdt, M. M. (2018a). Preallocation and Planning under Stochastic Resource Constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

De Nijs, F., Theocharous, G., Vlassis, N., De Weerdt, M. M., & Spaan, M. T. J. (2018b). Capacity-aware sequential recommendations. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*.

De Nijs, F., Walraven, E., De Weerdt, M. M., & Spaan, M. T. J. (2017). Bounding the Probability of Resource Constraint Violations in Multi-Agent MDPs. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

De Weerdt, M. M., Stein, S., Gerding, E. H., Robu, V., & Jennings, N. R. (2016). Intention-aware routing of electric vehicles. *IEEE Transactions on Intelligent Transportation Systems*, *17*(5), 1472–1482.

De Weerdt, M. M., Albert, M., Conitzer, V., & Van der Linden, K. (2018). Complexity of scheduling charging in the smart grid. In *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems*.

Dolgov, D. A., & Durfee, E. H. (2003). Approximating Optimal Policies for Agents with Limited Execution Resources. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Dolgov, D. A., & Durfee, E. H. (2006). Resource allocation among agents with MDP-induced preferences. *Journal of Artificial Intelligence Research*, *27*, 505–549.

Duff, M. O. (2002). *Optimal learning: Computational procedures for Bayes-adaptive Markov decision processes*. Ph.D. thesis, University of Massachusetts Amherst.

García, J., & Fernández, F. (2015). A Comprehensive Survey on Safe Reinforcement Learning. *Journal of Machine Learning Research*, *16*(1), 1437–1480.

Ghavamzadeh, M., Mannor, S., Pineau, J., & Tamar, A. (2015). Bayesian reinforcement learning: a survey. *Foundations and Trends in Machine Learning*, *8*(5–6), 359–483.

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

Harman, R. M. (2002). Wireless solutions for aircraft condition based maintenance systems. In *Proceedings of the IEEE Aerospace Conference*.

He, S., Wallace, M., Gange, G., Liebman, A., & Wilson, C. (2018). A fast and scalable algorithm for scheduling large numbers of devices under real-time pricing. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming*. Springer.

Hoy, M. B. (2018). Alexa, Siri, Cortana, and more: An introduction to voice assistants. *Medical Reference Services Quarterly*, *37*(1), 81–88.

Huang, Y., & Kallenberg, L. C. (1994). On finding optimal policies for Markov decision chains: a unifying framework for mean-variance-tradeoffs. *Mathematics of Operations Research*, *19*(2), 434–448.

Isom, J. D., Meyn, S. P., & Braatz, R. D. (2008). Piecewise Linear Dynamic Programming for Constrained POMDPs. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Jardine, A. K., Lin, D., & Banjevic, D. (2006). A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing*, *20*(7), 1483–1510.

Kaelbling, L. P., Littman, M. L., & Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, *101*(1), 99–134.

Kallenberg, L. C. M. (1983). *Linear programming and finite Markovian control problems*. No. 148 in Mathematical Centre Tracts. Mathematisch Centrum, Amsterdam.

Khonji, M., Jasour, A., & Williams, B. (2019). Approximability of constant-horizon constrained POMDP. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Kim, D., Kim, K., & Poupart, P. (2012). Cost-sensitive exploration in Bayesian reinforcement learning. In Pereira, F., Burges, C. J. C., Bottou, L., & Weinberger, K. Q. (Eds.), *Advances in Neural Information Processing Systems 25*.

Kim, D., Lee, J., Kim, K., & Poupart, P. (2011). Point-Based Value Iteration for Constrained POMDPs. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Kumar, R. R., Varakantham, P., & Kumar, A. (2017). Decentralized planning in stochastic environments with submodular rewards. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Kurniawati, H., Hsu, D., & Lee, W. S. (2008). SARSOP : Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces. In *Robotics: Science and Systems IV*.

Lee, J., Jang, Y., Poupart, P., & Kim, K. (2017). Constrained Bayesian reinforcement learning via approximate linear programming. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Lee, J., Kim, G., Poupart, P., & Kim, K. (2018). Monte-Carlo Tree Search for Constrained POMDPs. In *Advances in Neural Information Processing Systems*.

Littman, M. L., Dean, T. L., & Kaelbling, L. P. (1995). On the complexity of solving Markov decision problems. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.

Mausam, Benazera, E., Brafman, R., Meuleau, N., & Hansen, E. A. (2005). Planning with Continuous Resources in Stochastic Domains. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Meuleau, N., Hauskrecht, M., Kim, K.-E., Peshkin, L., Kaelbling, L. P., Dean, T. L., & Boutilier, C. (1998). Solving very large weakly coupled Markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Möller, J., Trilling, D., Helberger, N., & van Es, B. (2018). Do not blame it on the algorithm: an empirical assessment of multiple recommender systems and their impact on content diversity. *Information, Communication & Society*, *21*(7), 959–977.

Neves, L. C., & Frangopol, D. M. (2005). Condition, safety and cost profiles for deteriorating structures with emphasis on bridges. *Reliability Engineering & System Safety*, *89*(2), 185 – 198.

Oliehoek, F. A., & Amato, C. (2016). *A Concise Introduction to Decentralized POMDPs*. Springer.

Oliehoek, F. A., Witwicki, S. J., & Kaelbling, L. P. (2012). Influence-based abstraction for multiagent systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Oliehoek, F. A., Spaan, M. T. J., & Witwicki, S. J. (2015). Factored upper bounds for multiagent planning problems under uncertainty with non-factored value functions. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Ono, M., Pavone, M., Kuwata, Y., & Balaram, J. (2015). Chance-constrained dynamic programming with application to risk-aware robotic space exploration. *Autonomous Robots*, *39*(4), 555–571.

Papadimitriou, C. H., & Tsitsiklis, J. N. (1987). The complexity of Markov decision processes. *Mathematics of Operations Research*, *12*(3), 441–450.

Paruchuri, P., Tambe, M., Ordonez, F., & Kraus, S. (2004). Towards a formalization of teamwork with resource constraints. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems*.

Pineau, J., Gordon, G., & Thrun, S. (2003). Point-based value iteration: An anytime algorithm for POMDPs. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Pisinger, D. (2005). Where are the hard knapsack problems?. *Computers & Operations Research*, *32*(9), 2271–2284.

Pita, J., Jain, M., Marecki, J., Ordóñez, F., Portway, C., Tambe, M., Western, C., Paruchuri, P., & Kraus, S. (2008). Deployed armor protection: the application of a game theoretic model for security at the los angeles international airport. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems*.

Poupart, P., Kim, K., & Kim, D. (2011). Closing the Gap: Improved Bounds on Optimal POMDP Solutions. In *Proceedings of the International Conference on Automated Planning and Scheduling*.

Poupart, P., Malhotra, A., Pei, P., Kim, K.-E., Goh, B., & Bowling, M. (2015). Approximate Linear Programming for Constrained Partially Observable Markov Decision Processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc.

Ray, A., Achiam, J., & Amodei, D. (2019). Benchmarking safe exploration in deep reinforcement learning. *Preprint—under review*.

Robbel, P., Oliehoek, F. A., & Kochenderfer, M. J. (2016). Exploiting anonymity in approximate linear programming: Scaling to large multiagent mdps. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Roijers, D. M., Vamplew, P., Whiteson, S., & Dazeley, R. (2013). A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, *48*, 67–113.

Roijers, D. M., Walraven, E., & Spaan, M. T. J. (2018). Bootstrapping LPs in Value Iteration for Multi-Objective and Partially Observable MDPs. In *Proceedings of the International Conference on Automated Planning and Scheduling*.

Ross, K. W., & Varadarajan, R. (1989). Markov decision processes with sample path constraints: the communicating case. *Operations Research*, *37*(5), 780–790.

Rossman, L. A. (1977). Reliability-constrained dynamic programing and randomized release rules in reservoir management. *Water Resources Research*, *13*(2), 247–255.

Santana, P., Thiébaux, S., & Williams, B. (2016a). RAO*: an Algorithm for Chance-Constrained POMDP's. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Santana, P., Vaquero, T., McGhan, C. L., Toledo, C., Timmons, E., Williams, B., & Murray, R. (2016b). Risk-aware planning in hybrid domains: An application to autonomous planetary rovers. In *Proceedings of the AIAA Space and Astronautics Forum and Exposition*.

Scharpff, J., Roijers, D. M., Oliehoek, F. A., Spaan, M. T. J., & de Weerdt, M. M. (2016). Solving transition-independent multi-agent mdps with sparse interactions. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., & Hassabis, D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, *362*(6419), 1140–1144.

Silver, D., & Veness, J. (2010). Monte-Carlo Planning in Large POMDPs. In *Advances in Neural Information Processing Systems*.

Simão, T. D., Jansen, N., & Spaan, M. T. J. (2021). AlwaysSafe: Reinforcement learning without safety constraint violations during training. In *Proc. of Int. Conference on Autonomous Agents and Multi Agent Systems*.

Smith, T., & Simmons, R. (2005). Point-Based POMDP Algorithms: Improved Analysis and Implementation. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.

Sobel, M. J. (1985). Maximal mean/standard deviation ratio in an undiscounted MDP. *Operations Research Letters*, *4*(4), 157–159.

Sondik, E. J. (1978). The Optimal Control of Partially Observable Markov Processes over the Infinite Horizon: Discounted Cost. *Operations Research*, *26*(2), 241–276.

Spaan, M. T. J. (2012). Partially observable Markov decision processes. In Wiering, M., & van Otterlo, M. (Eds.), *Reinforcement Learning*, Vol. 12 of *Adaptation, Learning, and Optimization*, pp. 387–414. Springer.

Spaan, M. T. J., Oliehoek, F. A., & Vlassis, N. (2008). Multiagent planning under uncertainty with stochastic communication delays. In *Proceedings of the International Conference on Automated Planning and Scheduling*.

Spaan, M. T. J., & Vlassis, N. (2005). Perseus: Randomized Point-based Value Iteration for POMDPs. *Journal of Artificial Intelligence Research*, *24*, 195–220.

Tehrani, P., Liu, K., & Zhao, Q. (2012). Opportunistic spectrum access in unslotted primary systems. *Journal of the Franklin Institute*, *349*(3), 985–1010.

Tessler, C., Mankowitz, D. J., & Mannor, S. (2019). Reward constrained policy optimization. In *7th International Conference on Learning Representations*.

Undurti, A., & How, J. P. (2010). An Online Algorithm for Constrained POMDPs. In *Proceedings of the International Conference on Robotics and Automation*.

Van den Boomen, M., Spaan, M. T. J., Shang, Y., & Wolfert, A. R. M. (2020). Infrastructure maintenance and replacement optimization under multiple uncertainties and managerial flexibility. *Construction Management and Economics*, *38*(1), 91–107.

Wachi, A., & Sui, Y. (2020). Safe reinforcement learning in constrained Markov decision processes. In *Proceedings of the International Conference on Machine Learning*.

Walraven, E., & Spaan, M. T. J. (2017). Accelerated Vector Pruning for Optimal POMDP Solvers. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Walraven, E., & Spaan, M. T. J. (2018). Column Generation Algorithms for Constrained POMDPs. *Journal of Artificial Intelligence Research*, *62*, 489–533.

Witwicki, S. J., & Durfee, E. H. (2010). Influence-Based Policy Abstraction for Weakly-Coupled Dec-POMDPs. In *Proceedings of the International Conference on Automated Planning and Scheduling*.

Wu, F., Jennings, N., & Chen, X. (2012). Sample-based policy iteration for constrained DEC-POMDPs. In *Proceedings of the European Conference on Artificial Intelligence*.

Wu, J., & Durfee, E. H. (2010). Resource-driven mission-phasing techniques for constrained agents in stochastic environments. *Journal of Artificial Intelligence Research*, *38*, 415–473.

Yang, Q., Simão, T. D., Tindemans, S. H., & Spaan, M. T. J. (2021). WCSAC: Worst-case soft actor critic for safety-constrained reinforcement learning. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*.

Yang, T., Rosca, J., Narasimhan, K., & Ramadge, P. J. (2020). Projection-based constrained policy optimization. In *8th International Conference on Learning Representations*.

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

Younes, H. k. L. S., Littman, M. L., Weissman, D., & Asmuth, J. (2005). The first probabilistic track of the international planning competition. *Journal of Artificial Intelligence Research*, *24*, 851–887.

Yu, M., Yang, Z., Kolar, M., & Wang, Z. (2019). Convergent policy optimization for safe reinforcement learning. In *Advances in Neural Information Processing Systems*.