Resource-constrained Multi-agent Markov Decision Processes

de Nijs, Frits

DOI
10.4233/uuid:89c0f1a2-d19f-4466-9cc5-52aeb3950e53
Publication date
2019
Document Version
Final published version

Resource-constrained Multi-agent Markov Decision Processes

Resource-constrained Multi-agent Markov Decision Processes

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. dr. ir. T. H. J. J. van der Hagen;
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op

donderdag, 4 april 2019, om 12:30 uur

door

Frits DE NIJS

ingenieur in de computerwetenschappen, Technische Universiteit Delft, Nederland
geboren te 's-Gravenhage.

SIKS Dissertation Series No. 2019-08
The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

This research was made possible through the financial support of Alliander.

ISBN: 978-94-6375-357-9
Printing: Ridderprint BV, www.ridderprint.nl
Keywords: Decision making under uncertainty, multi-agent systems, optimization, constraint decoupling, reinforcement learning.

Contents

Summary

Intelligent autonomous agents, designed to automate and simplify many aspects of our society, will increasingly be required to also interact with other agents autonomously. Where agents interact, they are likely to encounter resource constraints. For example, agents managing household appliances to optimize electricity usage might need to share the limited capacity of the distribution grid.

This thesis describes research into new algorithms for optimizing the behavior of agents operating in constrained environments, when these agents have significant uncertainty about the effects of their actions on their state. Such systems are effectively modeled in a framework of constrained multi-agent Markov decision processes (MDPs). A single-agent MDP model captures the uncertainty in the outcome of the actions chosen by a specific agent. It does so by providing a probabilistic model of state transitions, describing the likelihood of arriving in a future state, conditional on the current state and action. Agents collect different rewards or penalties depending on the current state and chosen action, informing their objective of maximizing their expected reward. To include constraints, resource consumption functions are added to the actions, and the agents' (shared) objective is modified with a condition restricting their (cumulative) resource consumption.

We begin by analyzing approaches proposed in the literature to solve constrained, multi-agent MDPs, categorizing works according to the scope of the constraints, and whether constraints are enforced statically in advance or (also) dynamically during execution. Static solutions preallocate the available resources by committing to either the worst case, or the expected resource demand of an agent. Unfortunately, both approaches have their drawbacks: no effective polynomial-time algorithms have been proposed for computing worst-case allocations, and expected-case allocations cannot provide any guarantees to meeting the constraints. Dynamic approaches, which allocate resources according to realizations of uncertain state, have surprisingly not been studied extensively. Finally, we observe that all existing works assume the agent models are fully specified and known in advance, a significant obstacle in practice. In this thesis we address each of these challenges in turn.

Based on this analysis of the literature, we identify and describe four promising algorithms to compute optimal static resource preallocations. Two of them, a linear programming model and a Lagrangian decomposition using column generation, both compute expected-demand allocations. Accordingly, the other two algorithms compute worst-case allocations; the first approach is a mixed-integer linear programming model, while the second approach again applies a Lagrangian decomposition to the coupling constraint. We expect the two decomposition algorithms to solve multi-agent problems more efficiently, because they break down a large problem into smaller individual agent subproblems which can be solved in parallel.

For our first contribution, we study how to develop more effective static resource preallocation algorithms. We present constrained factored policy iteration, an iterative resource allocation algorithm for computing (sub-optimal) worst-case resource preallocations in polynomial time. At the same time, we demonstrate how a reduced resource capacity limit, derived from Hoeffding's inequality, can give a probabilistic bound on the probability of resource constraint violations of expected consumption preallocations. We show furthermore that, by iteratively loosening the reduced limits, we can move the actual probability of constraint violations arbitrarily close to a given risk bound.

An additional challenge occurs when the resource capacity is itself subject to uncertainty, such as in the case of a renewable power generation forecast from weather predictions. We show that preallocations can also be used effectively in this situation, by merging the forecast into the state transition uncertainty of each single-agent MDP. This significantly extends the time that agents can operate without communicating.

Next, we investigate how to use dynamic resource allocation to overcome inefficiencies resulting from uncertainty. We first study the potential to deploy the previously proposed static preallocation algorithms in a rolling horizon fashion. However, we prove that this can lead to arbitrarily poor solution quality, as a result of such policies not taking into account the consequence of being (or failing to be) awarded resources. We show that these challenges can be overcome by employing a resource arbiter, an on-line module which prevents constraint violations by modifying the actions selected by agents. In order to compute policies which are aware of the arbiter's effects, we propose algorithms which simulate agents' joint behavior (including that of the arbiter) and compute individual best responses to the expected outcome.

Subsequently, we relax the assumption that we know the agent models in advance. Instead, we take an optimal reinforcement learning perspective, by computing policies which optimally trade off exploration for new knowledge about agent models with exploitation of current model knowledge. We propose bounded-regret belief space planning, a new approximate algorithm for the learning problem. We demonstrate that this algorithm can be integrated into existing static preallocation algorithms, thereby allowing us to compute resource constraint aware learning policies.

In conclusion, this thesis proposes novel algorithms to advance the state of the art in three challenging settings: computing static preallocations, dynamic allocations, and constrained model learning policies. Taken together, these algorithms show how agents can coordinate their actions under uncertainty and shared resource constraints in a broad range of conditions. Furthermore, the proposed solutions are complementary: static preallocations can be used as back-up strategy for when a communication disruption prevents the use of dynamic allocations.

To advance this line of research beyond the scope of this thesis, we make three recommendations for future research. In the first place, we see the need to develop planning techniques which can cover multiple timescales, in order to efficiently co-optimize investment decisions with operational requirements. Secondly, we expect a practical deployment will need to solve the problem of users misrepresenting their preferences in order to obtain better resource allocations. And finally, we expect that the interaction between autonomous agent and human user can be made more effective, by making the agent actively work to maintain trust, through explaining its decisions and keeping track of the user's emotional state.

Samenvatting

Intelligente en autonome agenten, ontwikkeld om meerdere aspecten van onze samenleving te automatiseren en makkelijker te maken, zullen steeds vaker autonoom met andere agenten moeten interacteren. Deze interacties tussen agenten zullen vaak over begrenzingen op hulpbronnen gaan. Agenten die huishoudelijke apparaten aansturen om zo gunstig mogelijk met elektriciteit om te gaan, zullen bijvoorbeeld rekening moeten houden met de maximale doorvoercapaciteit van het elektriciteitsdistributienetwerk.

Dit proefschrift beschrijft onderzoek naar nieuwe algoritmen om het gedrag van agenten in begrensde omgevingen te optimaliseren, rekening houdend met significante onzekerheid over het effect van hun acties op hun toestand. Deze systemen zijn effectief te modelleren in het raamwerk van Markoviaanse beslissingsprocessen (MBs) met meerdere agenten. Het MB van een individuele agent beschrijft de onzekerheid over het gevolg van een gekozen actie, door middel van een kansmodel op de toestandtransitie. Dit kansmodel beschrijft de waarschijnlijkheid van een volgende toestand, conditioneel op de huidige toestand en de gekozen actie. Agenten worden verder beloond of gestraft al naar gelang hun huidige toestand en actie, wat ze er toe drijft om hun verwachting over de beloning te maximaliseren. Om begrenzingen in dit model mee te nemen voegt men verbruiksfuncties toe aan de acties, en neemt men begrenzingen op (de som van) het hulpbronverbruik mee in het doel van de agenten.

We beginnen door de literatuur te analyseren naar bestaande methoden voor begrensde MBs met meerdere agenten, welke we indelen op basis van de reikwijdte van de begrenzingen, en de manier waarop hulpbronnen verdeeld worden: alleen statisch van tevoren, of (ook) dynamisch tijdens actieselectie. Statische oplossingen wijzen hulpbronnen toe aan agenten op basis van ofwel hun verbruik in het slechtste geval, ofwel hun gemiddelde verbruik. Helaas hebben bestaande methoden voor ieder type hun nadelen: er bestaat nog geen effectief algoritme om in polynomiale tijd een toewijzing voor het slechtste geval te berekenen, en toewijzingen voor het gemiddeld verbruik bieden geen garanties dat het daadwerkelijk verbruik aan de limiet voldoet. Dynamische oplossingen wijzen hulpbronnen pas toe op het moment dat de toestand bekend is, echter, deze zijn verrassend genoeg nog niet uitgebreid bestudeerd. Als laatste merken we op dat alle bestaande literatuur er vanuit gaat dat de modellen van agenten volledig gespecificeerd en bekend zijn, wat in de praktijk een significant obstakel is. In dit proefschrift behandelen we achtereenvolgens elk van deze uitdagingen.

Uit deze literatuurstudie identificeren en beschrijven we vier veelbelovende algoritmen om optimale statische toewijzingen te berekenen. Twee daarvan, een lineair programma en een Lagrangiaanse decompositie op basis van kolomgeneratie, berekenen allebei toewijzingen voor het gemiddeld verbruik. Derhalve berekenen de overige twee methoden toewijzingen voor het slechtste geval; de eerste methode is een lineair programma met gehele getallen, terwijl de tweede ook hiervoor een Lagrangiaanse decompositie op de gedeelde begrenzing voorstelt. Wij verwachten dat de twee decompositiealgoritmen in het algemeen efficiënter zijn in het oplossen van problemen met meerdere agenten, omdat deze één groot probleem opdelen in kleinere subproblemen per agent, welke in parallel opgelost kunnen worden.

Onze eerste bijdrage bestaat uit een studie naar effectievere algoritmen om statische toewijzingen te berekenen. We presenteren 'constrained factored policy iteration', een (suboptimaal) iteratief algoritme om in polynomiale tijd een toekenning van hulpbronnen voor het slechtste geval te berekenen. Tegelijkertijd laten we zien dat we, door een strakkere begrenzing te bepalen via de Hoeffding-ongelijkheid, een bovengrens kunnen stellen aan de kans op het overschrijden van de capaciteit van de hulpbronnen door toewijzingen voor gemiddeld verbruik. Vervolgens tonen we aan dat we de daadwerkelijke kans op overschrijdingen willekeurig dicht tegen een gegeven bovengrens kunnen brengen, door de strakkere begrenzing stapsgewijs losser te maken.

Onzekerheid omtrent de daadwerkelijke hoeveelheid hulpbronnen, zoals in het geval van een voorspelling over de energieproductie uit hernieuwbare bronnen afgeleid uit de weersvoorspelling, vormt een extra uitdaging voor statische toewijzingen. We laten zien dat ook in deze context statische toewijzingen gebruikt kunnen worden, wanneer we de voorspelling opnemen in de transitieonzekerheid van het MB van iedere agent. Hierdoor kunnen agenten significant langer opereren zonder te hoeven communiceren.

Vervolgens onderzoeken we hoe we hulpbronnen dynamisch kunnen toewijzen, om zo inefficiënties ten gevolge van onzekerheid te verminderen. Als eerste bekijken we of we de hiervoor genoemde statische toewijzingen met een rollende horizon kunnen toepassen. Echter bewijzen we dat dit tot willekeurig slechte oplossingen kan leiden, omdat deze oplossingen geen rekening houden met de consequenties van het (niet) toegewezen krijgen van hulpbronnen. We laten zien dat deze uitdaging opgelost kan worden door gebruik te maken van een hulpbronrechter, welke op het moment van uitvoeren overschrijdingen voorkomt door de acties van agenten aan te passen. Om agenten acties te laten kiezen die rekening houden met het effect van deze rechter, stellen we algoritmen voor die het gedrag van alle agenten (inclusief de rechter) simuleren om zo individuele beste tegenstrategiën te berekenen op het verwachtte gedrag.

Daaropvolgend richten we ons op de aanname dat we de modellen van de agenten van tevoren weten. Daarvoor nemen we het perspectief van optimaal bekrachtigend leren, wat inhoud dat we een gedrag berekenen dat een optimale afweging maakt tussen verkennen, om nieuwe kennis over het model op te doen, en het benutten van de huidige

modelkennis. We stellen ‘bounded-regret belief space planning’ voor, een nieuw benaderingsalgoritme voor het leerprobleem. We tonen aan dat dit algoritme geïntegreerd kan worden in bestaande algoritmen voor de statische toewijzing van hulpbronnen. Daardoor kunnen we lerend gedrag berekenen dat rekening houd met de begrenzingen.

Concluderend stellen we in dit proefschrift nieuwe algoritmen voor die de huidige stand van zaken vooruit brengen, door drie uitdagingen aan te vallen in het berekenen van: statische toewijzingen, dynamische toewijzingen, en modeldynamiek lerend gedrag. Samengenomen laten deze algoritmen voor een brede groep condities zien hoe agenten hun gedrag kunnen coördineren onder onzekerheid en gedeelde beperkingen op hulpbronnen. Bovendien zijn de voorgestelde oplossingen complementair: statische toewijzingen kunnen ingezet worden als aanvullende strategie, voor het geval de communicatie wegvalt en dynamische toewijzing niet gebruikt kan worden.

Om deze onderzoeksrichting na dit proefschrift verder door te trekken, doen we drie aanbevelingen voor vervolgonderzoek. In de eerste plaats zien we de noodzaak om planningstechnieken te ontwikkelen die met meerdere tijdsschalen om kunnen gaan, zodat het mogelijk wordt om op een efficiënte manier investeringsbeslissingen samen met operationele beperkingen te optimaliseren. Ten tweede verwachten we dat een praktische implementatie van dit werk een oplossing zal moeten vinden voor het probleem van gebruikers die hun voorkeuren onjuist rapporteren, om zo een betere toewijzing van hulpbronnen te ontvangen. En als laatste verwachten we dat de interactie tussen de autonome agent en menselijke gebruiker effectiever gemaakt kan worden, door de agent actief te laten werken aan het vertrouwen van de gebruiker in het systeem, door gekozen beslissingen uit te leggen en de emotionele toestand van de gebruiker in acht te nemen.

Chapter 1

Introduction

Artificial Intelligence (AI) solutions are rapidly becoming integrated into society. Current well-known examples of AI in daily life are digital personal assistants (Hoy, 2018), (social) media recommender systems (Möller et al., 2018), and game playing AI capable of surpassing human experts (Silver et al., 2017). These examples have in common that they primarily interact one-on-one with the user. However, the introduction of intelligent agents to automate or simplify more aspects of society will increasingly see agents interact with each other autonomously, with the aim of creating intelligent *multi-agent* systems. These systems provide maximum benefit to society when agents are cooperating to achieve their common goals, which requires them to understand and reason about the impact they have on each other.

Where agents interact, sooner or later they will encounter resource constraints. Resource constraints are everywhere in daily life, even in contexts where we would not think of them as resources. For example, in an office environment the shared printer is a constrained resource, in more than one way. On the one hand, only one user can print at the same time, making the printer itself a resource. On the other hand, printing also consumes ink and paper, and running out of either one prevents subsequent users from printing until they are refilled. Although printing has relatively small-scale interactions which can be handled manually or using a print queue, more intelligent control may nevertheless improve user experience: consider the situation where a user with a large, low-priority print job is queued before a user with a single, high-priority page. In this case, interrupting the low-priority task would greatly improve the users' overall utility.

While such interactions are relatively rare when printing, there are many situations where agents must continuously coordinate. One such example is the control of the future electricity grid: the ongoing electrification of households (e.g., replacing gas with electricity for heating) is expected to cause neighborhood demand to exceed the power limits on the distribution grid increasingly often. Autonomous energy management systems can help alleviate this problem by optimizing when flexible devices activate subject to the power constraints, thereby spreading demand over time (Scott et al., 2013).

These examples illustrate that it is important for agents not only to coordinate, but to anticipate the availability of resources. However, when we *plan* to anticipate the future, we inevitably have to cope with the uncertainty inherent in predicting the consequences of actions. The presence of other agents further complicates this problem, because the future availability of resources also depends on the uncertain futures of *all other* agents. Therefore, this thesis explores the question:

> *How can agents optimize their behavior under uncertainty, to maximize their collective utility while jointly respecting the global resource constraints?*

In answering this question, we find that Markov decision processes are a powerful modeling framework for optimizing decision making under uncertainty. Unfortunately, when evaluating approaches developed for constrained Markov decision processes, we observe that existing algorithms are either intractable to compute for interesting models, or result in solutions which regularly exceed the imposed constraints in practice. Therefore, in this thesis we develop algorithms which are tractable to solve, yet can provide hard guarantees on constraint satisfaction. These algorithms bring practical control of multi-agent systems subject to constraints several steps closer to reality.

In this thesis we tackle the main research question from the following three perspectives, which together cover a broad spectrum of problem domains:

1. how to compute safe resource preallocations for a decentralized setting, where agents cannot communicate during policy execution;
2. how to compute dynamic resource allocations, in case agents are able to communicate during policy execution; and
3. how to compute constrained policies for the setting where agents start off without an accurate model of their dynamics, and therefore must learn safely?

As a result of developing novel efficient algorithms for each of these settings, we significantly advance the state of the art in planning *resource-constrained multi-agent Markov decision processes*.

In the remainder of this chapter, we motivate our work in the context of a running example of a system for balancing the future electricity grid using heat pumps, presented below. In Section 1.1, we first introduce the concept of planning, and how to reason about and incorporate uncertainty. Next, in Section 1.2 we provide an overview of prior work on resource constraints in multi-agent systems. We categorize prior work based on the type of solution it provides, and identify several open challenges in current research, motivating the work in this thesis. Section 1.3 then highlights the contributions we provide to address these open challenges. Finally, Section 1.3.1 gives an overview of the structure of the thesis.

Running example: demand response using heat pumps

The goal of greatly reducing worldwide greenhouse gas emissions in the coming decades imposes an ‘energy transition’: shifting as many systems as possible from fossil fuels to sustainable alternatives. This transition affects both consumer-side systems and the production-side companies. Consumer energy demand, such as from cars and household heating will transition from petrol- and gas-based to electricity-based. On the production side, coal and gas will have to be replaced by renewable sources like wind and solar power, which are only partially under our control. As a consequence, the limits of our future grid infrastructure will be reached much more frequently: new electrical loads such as electric vehicles and heat pumps have both a significantly higher load than traditional household appliances *and* they are much more likely to be ‘on’ at the same time, strongly affecting the peak demand and even threatening to exceed peak capacity of grid elements such as transformers. At the same time, power production becomes more volatile as a result of the effect of weather on renewable generation.

Both challenges could in principle be averted with significant investments, by reinforcing the electricity grid and keeping significant controllable generation on standby. However, to keep our energy system affordable it may be more effective to make demand responsive to grid limitations, shifting some of the demand in time (Palensky and Dietrich, 2011; Scott, 2016). Electric household heating systems like heat pumps have a significant potential to contribute, because they allow us to exploit the available system inertia: for well-insulated buildings, running the heat pump a few hours earlier can obtain the same level of comfort at negligible extra energy loss. However, if all houses individually optimize their behavior, they will likely all respond in the same way, shifting the peak in time without reducing it. Therefore, we should instead jointly optimize an activation schedule for a neighborhood of heat pumps, subject to the available capacity of the network. This is an example of the constrained, multi-agent planning problems that we intend to address in this thesis.

1.1 Planning under uncertainty

When we develop agents for a specific task such as operating a heat pump, we expect them to be intelligent enough to *anticipate* events that are known to happen in the future. As an example, consider the situation in Figure 1.1. Here, a future power restriction constraining the heater to remain off threatens to lower the temperature below the minimum comfort level. Knowledge about this restriction allows us to anticipate, by enabling the heat pump a few minutes earlier than usual. This type of anticipatory control is only possible if agents plan their actions in advance. In particular, agents should optimize a controller capable of producing the sequence of decisions that optimizes the temperature trajectory.

Assuming that the interior temperature behaves perfectly according to a determinis-

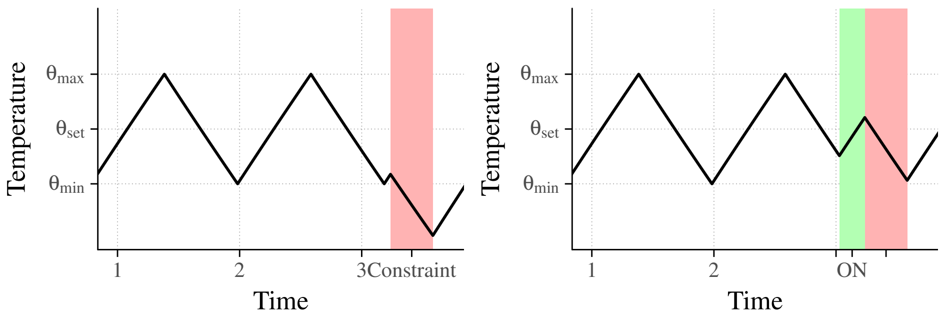


Figure 1.1: Knowledge of a future outage gives us the opportunity to anticipate, by heating at an earlier time than we would normally.

tic temperature equation, it suffices to compute an adequate control for each decision step. However, this ignores the significant uncertainty present in practice; some influences which occur in reality may have only been modeled on average, such as the additional heating effect of solar irradiation, or the influence of outdoor temperature on the efficiency of the heat exchange. Other effects on the indoor temperature transition, such as the future outdoor temperature, and whether the occupant will open a window or start cooking, may be impossible to predict perfectly. Therefore, when an agent develops a plan, it should explicitly reason about *uncertainty* in the consequences of its actions.

Uncertainty implies that taking an action may have multiple potential outcomes as consequence, each with some probability of occurring. For example, when the indoor temperature is currently 20°C and the heat pump is switched off, the temperature may have 70% chance to lie between 18.5 and 19°C in one hour. When a rational decision maker wants to reason over the value of a particular action with uncertain outcomes, it should base its decision on probability theory (Bertsekas and Tsitsiklis, 2008), as this framework provides us with the tools to infer the *expected* quality of different decisions.

Consider a discretized version of the heat pump planning problem, where the temperature 'state' of the system is one of {too high, high, med, low, too low} for the household occupants' comfort. Then, assuming there is some penalty to the controller for letting the temperature reach the 'too high/low' states, we can optimize over uncertain temperature transitions given a model of the uncertainty. Suppose that the temperature behaves as the transition function in Figure 1.2 (left). In this case, there is a 70% chance of reaching 'too low' when the heat pump is switched OFF when the temperature is currently 'low'. Therefore, by reasoning over the value of each potential outcome, we can determine that the expected value of ON exceeds OFF in this state.

To plan an optimal control policy for the heat pump we must determine the best action to take in *each* potential state, at every point in time. To determine the consequences of switching the heat pump ON or OFF at time t, we need to know the expected value of each temperature state at time $t + 1$. Therefore, it requires us to reason 'backwards' in

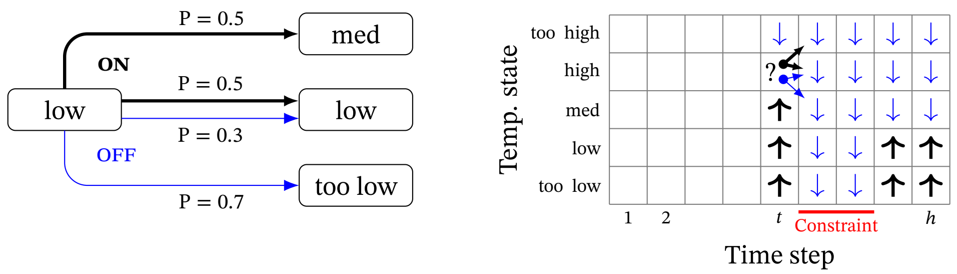


Figure 1.2: Example temperature transition function, giving the probability of reachable outcomes under the chosen action (left), and partially computed heat pump activation policy subject to a power constraint (right).

time, as shown schematically in Figure 1.2 (right). This allows the policy to anticipate a power restriction like in the example of Figure 1.1; at time step *t* just before the constraint, we determine the action to take in the 'high' temperature state as follows. Keeping the heat pump OFF, we reach a 'too low' temperature after the constraint with probability $0.7^3 = 0.343$, which is less than the 0.5 probability of reaching the 'too high' state when switching the heat pump ON. Therefore, the optimal policy at time *t* in temperature state 'high' is to switch the heat pump OFF.

Of course, this is not an idea restricted to our example of controlling heat pumps; the Bellman optimality principle (Bellman, 1957b) formalizes the notion of basing optimal control decisions on the expected consequences, as we did in our example. A Markov decision process (MDP; Bellman, 1957a; Puterman, 1994) describes the formal model of decision making problems under uncertainty. It consists of the considered states of the world (such as the temperature), the potential actions that an agent can take (switching the heat pump on or off), a transition function describing the likelihood of outcomes as a consequence of taking an action (as the example on the left side of Figure 1.2), and a *reward function*, giving the relative utility or how preferred the current state of the world is to the designer of the model. The reward function defines the goals of the controller, and by planning a policy for an MDP we are optimizing for the expected total reward.

The MDP formalism plays a central role both in works on decision-theoretic planning (Boutilier, Dean, and Hanks, 1999), where all components of the model are assumed known and a policy can be *computed* a priori, and in reinforcement learning (Sutton and Barto, 2018), where the control must be learned and refined through interactions with the environment. In this thesis we also base our algorithms on agents modeled as MDPs, because of their broad applicability. However, here we focus our attention on *coordination* in cooperative multi-agent MDPs (Boutilier, 1996). Coordination is especially promising when agents are 'weakly coupled' (Meuleau et al., 1998), meaning that the agents are nearly independent. Returning to the heat pump example, the decision of one agent to switch on its heater is unlikely to influence the temperature of any other house in the neighborhood. Thus, the agents in a neighborhood are transition independent (Becker et al., 2004), *except* for the fact that they are sharing the power production and transportation infrastructure, which places a limit on the maximum power draw, making the agents resource constrained.

1.2 Planning under resource constraints

Almost every system, environment or scenario we can imagine is in some sense constrained, whether due to the cost of designing and implementing an unconstrained system, or due to limitations imposed by the physical world. A resource constraint, in the broadest sense, is the specification that certain actions are only allowed when a resource is available; for example, an advertiser can only buy another advertising slot when it still has the financial budget to do so, and an aggregation of heat pumps cannot let its demand exceed the capacity of the distribution grid. Resource constraints can be imposed through the reward function of a Markov decision process, by imposing a 'reward' of negative infinity for constraint violating actions. However, doing so obscures the structure of the problem; therefore, it is beneficial to consider the constraints explicitly, as separate objectives to be satisfied while optimizing the main reward objective.

1.2.1 Existing work on constrained multi-agent systems

There exists an extensive body of work on decision making under uncertainty, see the books of (Puterman, 1994; Kochenderfer, 2015; Russell and Norvig, 2016) for an introduction to the field; the generalization of this problem to multiple decision makers is also an established field (Boutilier, 1996; Durfee and Zilberstein, 2013). Similarly, resource constrained models of decision making under uncertainty have received significant attention (Altman, 1999). However, in this thesis we are primarily interested in the combination of these three aspects, which only a limited number of previous works have considered. Here we give an overview of the most relevant previous work on planning under uncertainty in resource-constrained multi-agent problems.

Meuleau et al. (1998) look at a multi-agent planning problem subject to several constraints, motivated by a military operation planning domain. They consider optimizing the deployment of weapons by planes to targets. Each individual plane is limited in the amount of weapons it can carry, and the total stockpile shared between them is also finite. In addition, the total number of planes that can be deployed in each time step is also bounded. Their solution method focuses on optimizing the allocation of weapons to targets, by computing the expected value of each number of weapons sent to the target in each time-step through dynamic programming. The allocation of planes is made heuristically, through the use of a greedy (de)allocation of resources in an on-line phase.

The work of Dolgov and Durfee (2006b) proposes a mixed-integer linear programming approach to solve a server maintenance domain where a number of system administrators each try to keep as many of their own servers running as possible. The solution statically allocates resources required to reboot specific servers subject to multiple constraints: each admin has a limited budget with which to buy resources, and the number of each resource (i.e., skilled technicians in the region) to be allocated is also bounded. Wu and Durfee (2010) extend this work to multiple resource-allocation phases, during which agents can swap resources, in the context of multi-rover planetary exploration. Finally, Agrawal, Varakantham, and Yeoh (2016) explore an approximation to the problem, through a Lagrangian relaxation of the coupling constraint on the total number of each resource, using a greedy rounding scheme to obtain a feasible solution.

Gordon et al. (2012) look at a crowd management problem in a theme park setting, where the constraint is on the total number of visitors that can simultaneously participate in an attraction. They also employ a Lagrangian relaxation technique, with the difference that their method employs a probabilistic rounding scheme, which is made possible by the assumption that each individual agent has limited influence on the overall solution.

A similar setting was studied by Varakantham et al. (2012), who investigate a taxi routing domain where the taxis are constrained to pick up no more than the total number of passengers requesting a ride in a given zone of the city. In their algorithm, agents iteratively compute a best-response route to the previous joint solution, using the concept of fictitious play to ensure the distribution stabilizes.

Boutilier and Lu (2016) study algorithms for the coordination of agents subject to a global budget; they investigate an advertising domain, where the agents represent browsing sessions of web visitors. Agents may choose to insert (different levels of) advertisements on the pages of visitors, however, they should do so without exceeding the total budget of the advertiser. Their solution augments the state space of each agent with a factor indicating the amount of budget the agent is allowed to spend, resulting in optimal single-agent policies for all possible budget levels. A greedy on-line component is then used to continually (re)distribute remaining budget based on the realized states of the individual agents, allowing recourse of budget should a visitor close its session.

The work of Chen et al. (2016) looks at search-and-rescue scenarios with autonomous vehicles in a hazardous environment. Each vehicle has an individual health budget, with the vehicle expiring when it receives too much damage. Despite having only local constraints, the agents need to coordinate to keep coverage of the entire environment. However, because agents have relatively sparse interactions, they only have to coordinate with their direct neighbors. The authors propose to compute on-line control actions through a parallelized variant of Monte-Carlo tree search, where each agent keeps track of the health of its direct neighbors.

Finally, there is a close link between constrained multi-agent systems and *congested* multi-agent 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 value to lower resource demand states, *optimizing* consumption through time. While this thesis focuses on constrained systems, we mention two state-of-the-art approaches for congested systems here. 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% of optimality. He et al. (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.

1.2.2 Analysis of existing work

The works mentioned in the previous section all treat problems with multiple agents and resource constraints, but with different constraint models and control assumptions. In this section, we analyze these works along three dimensions: whether the resource represents a budget or an infrastructure capacity, whether the constraint is local to the agent or applies globally, and whether the algorithm computes a static control policy off-line or dynamically adjusts the resource allocation and control decisions on-line.

Budget and infrastructure constraints

The first distinction we make is between 'budget' constraints and 'infrastructure' constraints, which is a difference in how the resource behaves over time. A budget, like the advertising budget considered by Boutilier and Lu (2016), represents a bounded quantity which is (partially) consumed over time until it is depleted, at which point no further consumption is allowed. Budget constraints are used to model 'stockpiles', such as an amount of money, or the remaining energy stored in a battery.

On the other hand, an infrastructure constraint, also called instantaneous constraint by Meuleau et al. (1998), represents a maximum consumption capacity at each decision point, which replenishes after each time step. 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. Infrastructure constraints can be used to model bandwidth in a communication network, instantaneous power production and transmission capacity in an electricity grid, but also shared tools or CPU cycles.

Local and global constraints

In addition to this temporal aspect, there is also the consideration to whom the constraint applies. We observe a distinction between local constraints, which apply to individual agents, and global constraints, which apply to all agents simultaneously. All combinations of budget and infrastructure with local and global constraints can appear:

- Local budget: battery capacity of an autonomous vehicle.
- Local infrastructure: steep slope which can only be traversed downwards.
- Global budget: shared budget which individual agents use to make purchases.
- Global infrastructure: shared electricity transmission grid.

A multi-agent system with only local constraints may still have agents interacting with each other, for example when the agents model robots tasked with autonomous search-and-rescue with limited battery capacity or health, as in (Chen et al., 2016). When one of the robots heads back to charge, others should consider taking over its search area.

Static and dynamic control

Subsequently, we identify two types of approaches to how control is enforced in the solution: *static control* ensures that the resources are distributed in such a way at plan time, that during policy execution the resource usage is guaranteed to adhere to the constraints, and *dynamic control*, where the resource constraints are (additionally) enforced at execution time, allocating resources to agents based on their realized states.

Both approaches come with advantages and drawbacks; static control means that the system cannot respond to the effects of uncertainty, and therefore needs to be conservative with its allocations. On the other hand, under static control the agents are free to execute their policies without needing to communicate, allowing decentralized action selection. Dynamic control does require the agents to remain in contact with each other or with a centralized controller, but in turn this controller is able to assign resources to agents which need it based on their current state, instead of on their expected state(s).

1.2.3 Open challenges in solving resource-constrained problems

In Table 1.1 we classify the existing works on constrained multi-agent systems according to the dimensions described in the previous section. We observe that global infrastructure constraints are the most commonly considered constraint addressed by current studies. However, while these constraints are well-studied, existing solutions nevertheless suffer from drawbacks that prevent application to large multi-agent systems.

Algorithms to handle global infrastructure constraints typically do so by computing static control policies. Dolgov and Durfee (2006b) show that optimal static control can be achieved by computing resource preallocations through a mixed-integer linear program. Unfortunately, this comes at the cost of exponential complexity in the number of agents and horizon length, which has prompted work in relaxed approaches. However, these approaches either require specific assumptions on the agent models, in the case of Gordon et al. (2012), or retain exponential complexity in the horizon length for Agrawal, Varakantham, and Yeoh (2016). Further relaxation is possible by computing policies which satisfy the constraints in expectation. However, this means that constraint violations are likely to occur in practice, raising the question:

	Local		Global		
Reference	Budget	Infra.	Budget	Infra.	Control
Meuleau et al. (1998)	✓		✓	✓	Dynamic
Dolgov and Durfee (2006b)	✓	✓		✓	Static
Wu and Durfee (2010)	✓	✓		✓	Static
Gordon et al. (2012)				✓	Static
Varakantham et al. (2012)				✓	Static
Boutilier and Lu (2016)			✓		Dynamic
Agrawal et al. (2016)	✓	✓		✓	Static
Chen et al. (2016)	✓				Dynamic
Kumar et al. (2017)				✓	Static
He et al. (2018)				✓	Static

Table 1.1: Classification of resource-constrained multi-agent planning problems in literature, according to the identified dimensions.

RQ 1. *How can safe resource preallocations for a decentralized setting be computed efficiently?*

Static control has the advantage that individual agents can execute their policies without having to communicate, which makes them robust to communication failures. Nevertheless, in practice some form of communication is likely available, which opens up the opportunity to dynamically coordinate resource allocations after observing the realizations of uncertain state transitions, making more effective use of resources and potentially resulting in better solutions. Existing algorithms for dynamic control of globally constrained systems use an open-loop structure, where the individual agent policies are not aware of the dynamic controller; as a result, the computed agent policies may seriously overestimate the actual value (Meuleau et al., 1998). How to compute high-quality dynamically controlled policies is therefore still an open challenge, prompting the question:

RQ 2. *How can we compute (near-)optimal policies exploiting communication between agents to perform dynamic resource allocation?*

One of the requirements to deploy planning approaches such as the methods described in Table 1.1 is the availability of an accurate model for each agent in the system. Practical systems may not have access to (fully) accurate models of agent dynamics; for example, in the heat pump planning example, it is unlikely that the system operator would know the exact insulation values of each household. To learn these values requires learning algorithms that are aware of the resource constraints. As prior work has focused on the planning setting, how to learn agent models in a globally constrained multi-agent system is still an open challenge, that we address in this thesis by investigating the following question:

RQ 3. *How can we compute policies to learn a model of agents' dynamics, when they operate in resource-constrained environments?*

1.3 Contributions and roadmap

In this thesis, we address the open challenges described in the previous section, by developing several novel approaches for solving multi-agent resource-constrained problems subject to uncertainty. In particular, we answer each of the raised questions in turn, by developing novel theory and algorithms tailored to each of the specific problem settings.

First we tackle *research question 1*, by showing two alternative approaches to address the challenge of computing decentralized control policies efficiently. One approach, *constrained factored policy iteration*, iteratively allocates resources to agents according to a myopic best value first approach. While this approach forgoes optimality guarantees, it is efficiently computable and proves to be highly effective in empirical evaluations. The second approach, *dynamic bound relaxation* (De Nijs et al., 2017), ensures that the probability of resource constraint violations made by stochastic resource allocation algorithms is upper bounded by a given risk probability. Through extensive experimental evaluation we show that both approaches outperform the state of the art. Additionally, we demonstrate how *preallocation algorithms can be made robust to uncertainty* about the level of the constraint itself, allowing agents to operate for longer without communicating (De Nijs, Spaan, and De Weerdt, 2018).

Next, we answer *research question 2*, where agents do have the opportunity to use continuous communication to dynamically adapt their resource allocation. In that case, we may attempt to embed the algorithms for static resource allocation in a rolling horizon strategy. However, we show that due to state transition uncertainty, *this strategy may perform arbitrarily worse* than the optimal centralized control policy. To overcome this weakness, we propose *two novel algorithms for decoupling agents with a dynamic resource arbiter*, which intervenes when the agents' chosen action would otherwise violate one or more constraints (De Nijs, Spaan, and De Weerdt, 2015; De Nijs, Spaan, and De Weerdt, 2016). We demonstrate empirically that both algorithms find high-quality solutions, while being robust against individual agents' uncertainty.

Finally, we address *research question 3*, the challenge of optimizing resource constrained decisions when the true models of agents are not fully known. We propose to learn agent dynamics interactively, through the use of Bayesian reasoning over a belief prior on potential agent models. By computing an optimal learning policy over all possible beliefs, the previously investigated constraint handling algorithms can be applied to the joint coordination problem. Unfortunately, computing such an optimal learning policy quickly becomes intractable due to the exponential growth of the belief space; we show that this limitation can be overcome by a *novel algorithm which* *bounds the growth of the belief space* whenever an agent can safely assume to know its dynamics (De Nijs et al., 2018). We evaluate this algorithm on a challenging tourist recommendation domain, and show that it is highly scalable while at the same time finding nearly optimal learning policies.

Taken together, these algorithms form a comprehensive suite of state-of-the-art approaches, covering a broad range of constrained multi-agent decision making problems. We next indicate where in this thesis these contributions can be found.

1.3.1 Thesis roadmap

This thesis investigates each of the three challenges identified in turn, through the following chapters:

Background (Chapter 2)

Introduces mathematical definitions and notation used throughout the thesis, including the formal model of multi-agent resource-constrained Markov Decision Processes (MDPs). Presents the state-of-the-art algorithms that form the baseline of our empirical comparisons.

Resource Preallocation Algorithms (Chapter 3)

Explores algorithms for computing static resource preallocations, and proposes two novel algorithms to overcome their drawbacks: a greedy resource allocation scheme, and a technique to bound the probability of constraint violations.

Dynamic Resource Allocation (Chapter 4)

Investigates the potential of communication to improve the quality of the solution. We propose a resource arbiter to assign resources to agents dynamically, and evaluate two techniques to compute arbiter-aware policies: one using the best-response principle, and the other based on fictitious play.

Constrained Multi-agent Learning (Chapter 5)

Studies how the previous results can be extended to the learning problem, where the dynamics of the model are hidden from the agent. We propose an optimal learning approach where the agent optimally trades of value of information with its expected reward, and show how to make this approach scale.

Conclusions (Chapter 6)

Concludes the thesis, with a discussion of the challenges and questions which remain open after the contributions described in the preceding chapters.

Chapter 2

Background

Before we can describe the contributions and algorithms in this thesis in detail, we first need to explain the notation, the models, and the previously developed algorithms to handle such resource-constrained (multi-agent) planning problems. We base our work on problems which can be modeled as Markov Decision Processes (MDPs), a well-known framework for modeling planning problems containing uncertainty. These models are described in detail in the book of Puterman (1994). Here we will restrict ourselves to presenting the basics, including the notation used throughout this thesis, in Section 2.1. Because traditional Markov decision processes are single-actor models, we pay special attention to the modeling of multi-agent Markov decision processes, and the independent agents assumption, in Section 2.1.2.

Then, we describe how we add resource constraints to the models, and investigate the consequences of doing so, in Section 2.2. Recall from the previous chapter (Table 1.1) that the state-of-the-art algorithms for multi-agent constrained planning problems compute static resource preallocations. These algorithms are derived from a linear program model for optimizing MDP policies, which makes it straightforward to add constraints. We present the linear program and resulting constrained formulations in Section 2.3. Because these approaches use a single centralized model, they do not exploit the independence between agents; fortunately, more advanced algorithms exist which can decompose the planning problems of the agents, presented in Section 2.4.

2.1 Models of decision making under uncertainty

In this section we describe the models of decision making which we will use in all subsequent chapters. One of the primary challenges a decision maker faces is *uncertainty*: every model is an abstraction of reality, and as a consequence this, predictions made by the model may differ from outcomes in reality. Combined with our potentially limited understanding of the modeled process, and the randomness inherent to reality, its clear that a decision maker should reason about uncertainty.

2.1.1 Markov Decision Processes

A finite horizon Markov Decision Process (Bellman, 1957a, MDP) is defined by the tuple $\langle S, A, T, R, h \rangle$. It consists of the finite sets of states S and actions A, a transition function T and reward function R defined over these sets, and a finite horizon h. Every time step t, the operator chooses an action $a \in A$, after which the system transitions from state $s \in S$ to subsequent state $s' \in S$. The uncertainty in the transition is captured by the transition function $T : S \times A \times S \to [0, 1]$, giving the probability of advancing to the next state as $T(s, a, s') = \mathrm{P}\left(s' \mid s, a\right)$. The choice of action a in state s is valued through the instantaneous reward function $R : S \times A \to \mathbb{R}$.

A solution to the planning problem takes the form of a policy $\pi : \{1, \dots, h\} \times S \to A$, which prescribes the action to take in each ⟨time, state⟩-pair. The objective of a planner is to compute the policy which obtains the maximum expected value over the entire operating horizon, when starting from a starting state s_1. The expected value of a policy is given by the value function

$$V_\pi(t, s) = R(s, \pi(t, s)) + \sum_{s' \in S} \Big(T(s, \pi(t, s), s') V_\pi(t + 1, s') \Big). \tag{2.1}$$

The optimal policy π^* can be computed efficiently through an application of dynamic programming over the time dimension, computing the value function at time t on the basis of the values at $t + 1$ by selecting the value maximizing action in each state, i.e.

$$\begin{aligned} V_{\pi^*}(h, s) &= \max_{a \in A} R(s, a), \\ V_{\pi^*}(t, s) &= \max_{a \in A} \Big(R(s, a) + \sum_{s' \in S} \Big(T(s, a, s') V_{\pi^*}(t + 1, s') \Big) \Big). \end{aligned} \tag{2.2}$$

2.1.2 Modeling multi-agent systems

There are several approaches to modeling stochastic worlds containing multiple actors, or agents, influencing the world simultaneously. Markov Games (Littman, 1994; Vrancx, Verbeeck, and Nowé, 2008) are a general framework for systems containing agents that may compete with each other to maximize their own individual reward. However, in this thesis we will restrict our attention to the cooperative case, which is simply known as the Multi-agent Markov Decision Process (MMDP; Boutilier, 1996). An MMDP models a system consisting of n agents, each responsible for choosing an action a_i according to its individual policy $\pi_i(t, s)$ defined over the system state s. These actions are combined into a joint action a for the definition of state transition and instantaneous reward functions.

In large-scale multi-agent systems, the requirement that each agent can observe the entire system state s at every instant can be too restrictive (Becker et al., 2004).

When agents condition their policy only on a locally observed factor of the state space s_i, the model is a decentralized MDP (Dec-MDP). General Dec-MDP problems are NEXP-complete (Bernstein et al., 2002), but the problem remains tractable when the transition and reward functions are independent. Independence factors the functions into per-agent components T_i, R_i without dependence on other agents' local state, as

$$T(s,a,s') = \prod_{i=1}^{n} T_i(s_i, a_i, s'_i), \quad R(s,a) = \sum_{i=1}^{n} R_i(s_i, a_i). \tag{2.3}$$

When both conditions are present, we have a model where each agent i is represented by its own MDP $\langle S_i, A_i, T_i, R_i, h\rangle$. Therefore, the optimal policy for each agent can also be computed separately from the other agents. However, this situation changes when the agents must choose actions to jointly satisfy constraints, as this introduces a weak coupling between their models (Meuleau et al., 1998).

2.2 Resource constraints

The basic (multi-agent) MDP 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), with a recent comprehensive overview of the theory and algorithms to solve constrained MDPs given by Altman (1999).

In this thesis, we focus on global resource constraints. The resources in a problem may be grouped into resource types, such as energy, money or engineers. Each type of resource j has an amount of availability in each time step t, giving the resource limit $L_{j,t}$. For each agent i the consumption of resource type j is mapped using function $c_{i,j} : S_i \times A_i \to [0, c_{\max,i,j}]$, where $c_{\max,i,j}$ denotes the maximum potential consumption of resource type j by agent i. Agents are collectively constrained to use at most $L_{j,t}$ of the resource, which means that a constraint violation occurs if the agents collectively use more units of the type. Joint policy π violates the resource constraint for type r in joint state s at time t if

$$\sum_{i=1}^{n} c_{i,r}(s_i, \pi_i(t, s_i)) > L(r). \tag{2.4}$$

2.2.1 Coupling strength of constraints

The presence of constraints causes agents to exert *influence* on the (allowable) decisions of other agents. Where agents influence each other only locally, Witwicki, Oliehoek, and Kaelbling (2012) and Oliehoek, Spaan, and Witwicki (2015) show that such sparse

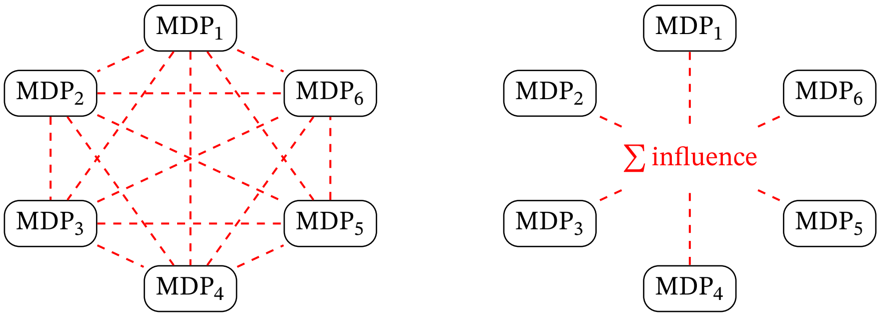


Figure 2.1: Constraints impose an all-to-all coupling between the otherwise independent single-agent MDPs (*left*). The weak coupling imposed by constraints suggests abstracting the influence agents have on each other (*right*).

influences can be approximated with bounded loss of quality, even when the original models are not factored. However, whether agents factor into independent models or not, global constraints always fully couple their interactions. Globally constrained but otherwise independent multi-agent systems have nevertheless previously been characterized as 'weakly coupled' (Meuleau et al., 1998; Adelman and Mersereau, 2008). This claim appeals to the intuitive idea that, from the perspective of one particular agent, the other agents exert *anonymous* influence (Robbel, Oliehoek, and Kochenderfer, 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. Figure 2.1 schematically presents this idea of decoupling the agents by abstracting their influence on the constraint.

2.3 Decoupling constraints: preallocation algorithms

In order to decouple constraints as indicated in the previous section, several existing algorithms compute resource preallocations. A resource preallocation is an unconditional a priori assignment of resources to agents, allowing each agent to reason about the constraints on its policy. Therefore, a resource preallocation decouples the agents completely, because each agent only needs to consider its own state and allocation to determine if it can safely choose a resource-consuming action. Formally, a resource preallocation defines a per-agent resource limit L_i such that the total allocation satisfies the constraints,

$$\begin{aligned} \sum_{i=1}^{n} L_{i,j,t} &\leq L_{j,t} && \forall j,t, \\ c(t,s,\pi_i(t,s)) &\leq L_{i,j,t} && \forall i,s,j,t. \end{aligned} \tag{2.5}$$

Unfortunately, such a deterministic preallocation is itself still hard to compute. To obtain a tractable algorithm, the single-agent constraint must be relaxed to satisfaction in expectation,

$$\mathrm{E}[C_{\pi_i}] \leq L_{i,j,t} \qquad \forall i, j, t. \tag{2.6}$$

As a result of this relaxation, the consumption of an agent may sometimes exceed its preallocated amount. While this can result in occasional violations of the global constraint, this may be acceptable in domains where constraints are non-destructive (e.g. when the limit is the length of a queue, exceeding it delays service).

Several algorithms to compute preallocations have been proposed, which we describe in the following sections. First we explain the occupancy dual LP, which forms the basis of the Constrained MDP LP and the resource preallocation MILP. Then we explain a more scalable column generation approach to computing preallocation policies.

2.3.1 Computing MDP policies through linear programming

Traditional MDP algorithms exploit the fact that the Bellman equation not only describes optimality, but also prescribes the method to get there through the fix-point: starting from random 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, Dean, and Kaelbling, 1995):

$$\begin{aligned}
\min_{v_{1,s}} \; & \sum_{s\in S} \mathrm{P}(1,s) v_{1,s} \\
\text{s.t. } & v_{t,s} \geq R(s,a) + \sum_{s'\in S} \mathrm{P}(s' \mid s,a) v_{t+1,s'} && \forall t < h, s, a \\
& v_{h,s} \geq R(s,a) && \forall s, a
\end{aligned} \tag{2.7}$$

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 consumption of actions. However, by the strong duality theorem, LP (2.7) has an equivalent 'dual' LP, which does have variables for actions (Littman, Dean, and Kaelbling, 1995),

$$\begin{aligned}
\max_{x_{t,s,a}} \; & \sum_{t=1}^{h} \sum_{s\in S} \sum_{a\in A} x_{t,s,a} R(s,a) \\
\text{s.t. } & \sum_{a'\in A} x_{t+1,s',a'} = \sum_{s\in S} \sum_{a\in A} \mathrm{P}(s' \mid s,a) x_{t,s,a} && \forall t < h, s' \\
& \sum_{a'\in A} x_{1,s',a'} = \mathrm{P}(1,s') && \forall s'
\end{aligned} \tag{2.8}$$

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 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

$$\mathrm{P}(a \mid \pi(t,s)) = \frac{x_{t,s,a}}{\sum_{a' \in A} x_{t,s,a'}} \tag{2.9}$$

Thus far, we have presented the LPs in the context of planning for a single MDP. In the case of a multi-agent 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 multi-agent dual LP:

$$\begin{aligned} \max_{x_{i,t,s,a}} \sum_{i=1}^{n} \sum_{t=1}^{h} \sum_{s \in S_i} \sum_{a \in A_i} & x_{i,t,s,a} R_i(s,a) \\ \text{s.t.} \sum_{a' \in A_i} x_{i,t+1,s',a'} &= \sum_{s \in S_i} \sum_{a \in A_i} \mathrm{P}_i(s' \mid s, a) x_{i,t,s,a} \qquad \forall i, t < h, s' \\ \sum_{a' \in A_i} x_{i,1,s',a'} &= \mathrm{P}_i(1, s') \qquad \forall i, s' \end{aligned} \tag{2.10}$$

2.3.2 Constrained MDPs

Constrained MDPs (CMDPs; Altman, 1999) leverage the dual LP in order to handle additional constraints, such as the ones we intend to model. In our case, we can add the resource constraints by adding the following constraint to LP (2.10):

$$\sum_{i=1}^{n} \sum_{s \in S_i} \sum_{a \in A_i} x_{i,t,s,a} \cdot c_j(t,s,a) \le L_{j,t} \qquad \forall j, t \tag{2.11}$$

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.

2.3.3 Resource allocation MILP

The Constrained MDP formalism is a powerful solution for our resource-constrained multi-agent 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 guarantee strict adherence to the constraints. Because in our setting agents are able to 'swap' resources at any time, we can make use of the model with an unbounded number of allocations. Practically, this means that when we have a model with binary resource, we can add the following binary variables and constraints to LP (2.10):

$$\begin{aligned}
&\sum_{s \in S_i} \sum_{a \in A_i} x_{i,t,s,a} \cdot c_j(t,s,a) \leq \Delta_{i,j,t} && \forall i,j,t \\
&\sum_{i=1}^{n} \Delta_{i,j,t} \leq L_{j,t} && \forall j,t \\
&\Delta_{i,j,t} \in \{0,1\} && \forall i,j,t
\end{aligned} \tag{2.12}$$

When the resource consumption function $c_j(t,s,a)$ 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 MILP (2.12) only applies to problems with binary resource consumption, Dolgov and Durfee (2006a) 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 $\mathcal{I}$ over the (ordered) set C_t of *potential* consumption vectors c_j:

$$\begin{aligned}
C_t &= \{c_j \mid \forall s,a:\; c_j(t,s,a) > 0\}, \\
\mathcal{I}(j, c_k) &= \begin{cases} 1 & \text{if } c_k = C_{t,j}, \\ 0 & \text{otherwise.} \end{cases}
\end{aligned} \tag{2.13}$$

Function $\mathcal{I}(j, c_k)$ 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 (2.12). 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 LP (2.10):

$$\begin{aligned}
&\sum_{s\in S_i}\sum_{a\in A_i} x_{i,t,s,a}\cdot \mathcal{I}\big(k, c(t,s,a)\big) \le \Delta_{i,c_k,t} && \forall i, c_k, t\\
&\Delta_{i,c_k,t}\cdot c_{k,j} \ge u_{i,j,t} && \forall i, j, c_k, t\\
&\sum_{i=1}^{n} u_{i,j,t} \le L_{j,t} && \forall j, t\\
&\Delta_{i,c_k,t}\in\{0,1\} && \forall i, c_k, t
\end{aligned}\tag{2.14}$$

With the modification proposed in (2.14), 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 strict. As a consequence, we expect this algorithm to be effective on problems which have a small number of critical constraints.

2.4 Multi-agent preallocation algorithms

The preallocation (MI)LPs discussed in section 2.3 decouple the constraints from the agents' planning problems, but they still require optimizing a single large centralized program. Advanced algorithms have been proposed which allow agents to solve their individual subproblems independently in both the expected value and strict constraint settings: (i) Column Generation applies a Lagrangian decomposition to the CMDP, resulting in an algorithm where the individual agent problems can be solved independently using dynamic programming. (ii) Lagrangian Dual Decomposition and Greedy Agent-based Prioritized Shaping (LDD+GAPS) applies a similar technique to the MILP model, with an additional greedy component to round the relaxed solution to a strict feasible solution.

2.4.1 Column Generation for Constrained MDPs

Column Generation (Gilmore and Gomory, 1961) is an effective technique for decomposing combinatorial optimization problems, provided the problem has some method to generate new potential solutions efficiently. The technique uses the insight that, when a linear program (LP) is used to select solutions from an exhaustive set, the simplex algorithm only considers one variable at a time. If we can generate the optimal element to be selected on the fly, we avoid having to maintain the exhaustive set of candidate variables explicitly. Generating the 'column' of constants that define the candidate variable comes down to optimizing an ancillary problem subject to the dual prices of the current simplex solution, known as the λ costs.

Conceptually, a column generation algorithm thus centers on a LP in which each variable x_i corresponds to one specific solution to the combinatorial optimization problem, such as a specific cut of a given stock, a path in a graph, or the policy of an MDP. Solving this LP for a given set of solutions Z results in the optimal mix of those solutions subject to the constraints, and a λ cost per constraint. This cost can be interpreted as the amount by which the objective value of the optimal mix would increase if one more unit of that resource were available. Of course, if vector λ gives the gains that could be made with the current set of solutions, then any unseen solution which obtains *more* than this value per unit of resource could replace part of the currently selected mix. The solution which would improve the solution the most is the one maximizing the value under costs λ. Finding this best solution is an optimization problem over λ.

Yost and Washburn (2000) identified that we can compute a policy that optimizes for λ 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_π, the expected consumption of a policy, which we will denote C_π, follows the same structure:

$$C_{\pi,j}(t,s) = c_j(t,s,\pi(s)) + \sum_{s' \in S} \big(\mathrm{P}(s' \mid s, \pi(s)) \cdot C_{\pi,j}(t+1,s')\big). \tag{2.15}$$

When we are searching for the maximally improving column, we are searching for the column satisfying (for $\times$ the cross product of the price vector λ with the consumption vector C_π):

$$\max_\pi \big(V_\pi(t,s) - \lambda \times C_\pi(t,s)\big) = \max_\pi V^C_{\pi,\lambda}(t,s). \tag{2.16}$$

Both V_π and C_π are Markovian, and therefore we can write out the optimization problem from the perspective of a single current state:

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

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 multi-agent case a newly computed policy optimized for objective (2.16) is then added to the set of potential policies Z_i for each agent i, which together form the

Algorithm 1 Column generation for CMDP M (Yost and Washburn, 2000).

```
   $\lambda = 0, \lambda' = \infty, Z = \emptyset$
1: while $\lambda \neq \lambda'$ do
2:     $\lambda \leftarrow \lambda'$
3:     $\forall i : \pi_{i,\text{new}} \leftarrow \text{PLAN}(M_i, \lambda)$          ▷ Equation (2.16)
4:     $Z_i \leftarrow Z_i \cup \pi_{i,\text{new}}$
5:     $\langle x, \lambda' \rangle \leftarrow \text{SOLVELP}(Z)$          ▷ Equation (2.17)
6: end while
7: return $\langle x, Z \rangle$
```

search space of the Column Generation 'master LP' selecting the optimal mix of policies subject to constraints:

$$\begin{aligned} \max_{x_{i,k}} \; & \sum_{i=1}^{n} \sum_{\pi_k \in Z_i} x_{i,k} V_{\pi_k}(1, s_1), & \\ \text{s.t.} \; & \sum_{i=1}^{n} \sum_{\pi_k \in Z_i} x_{i,k}\, C_{\pi_k,j,t}(1, s_1) \leq L_{j,t} & \forall j, t, \\ & \sum_{\pi_k \in Z_i} x_{i,k} = 1, & \forall i, \\ & x_{i,k} \geq 0, & \forall i, k. \end{aligned} \tag{2.17}$$

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

$$\mathrm{P}(\pi_i = \pi_{i,k}) = \frac{x_{i,k}}{\sum_{\pi_{k'} \in Z_i} x_{i,k'}} \tag{2.18}$$

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 (2.16) directly exploits the time-recursive structure present in the MDP. Therefore, we expect that in practice Column Generation will be able to significantly outperform CMDP.

2.4.2 Lagrangian dual decomposition for worst-case allocations

In Column Generation, the single-agent problems are decomposed by the introduction of $\lambda_{j,t}$ dual costs per resource. Agrawal, Varakantham, and Yeoh (2016) make use of the same principle applied to the (binary-consumption) resource preallocation MILP model, resulting 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:

$$\begin{aligned}
\max_{x_{t,s,a}} \sum_{t=1}^{h} \sum_{s \in S_i} \sum_{a \in A_i} & x_{t,s,a} R_i(s,a) - \sum_{t=1}^{h} \sum_{j} \lambda_{j,t} \Delta^i_{j,t} \\
\text{s.t.} \sum_{a' \in A_i} x_{t+1,s',a'} &= \sum_{s \in S_i} \sum_{a \in A_i} \mathrm{P}_i(s' \mid s,a) x_{t,s,a} && \forall t < h, s \\
\sum_{a' \in A_i} x_{1,s',a'} &= \mathrm{P}_i(1,s') && \forall s' \\
\sum_{s \in S_i} \sum_{a \in A_i} x_{t,s,a} \cdot c_j(t,s,a) &\leq \Delta^i_{j,t} && \forall j,t \\
\Delta^i_{j,t} &\in \{0,1\} && \forall j,t
\end{aligned} \tag{2.19}$$

For a given cost vector λ each agent can individually compute its optimal allocation and policy by optimizing (2.19). However, depending on the value of λ, these policies may not satisfy the constraints when combined. Agrawal, Varakantham, and Yeoh (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 this agent his policy, until no more policies can be feasibly allocated. In case too many resources would be used by the joint policy computed for λ, some agents will end up without a policy; these agents instead use their resource-free policy (i.e. the optimal policy computed with constraint $\Delta^i_{j,t} = 0 \ \forall j,t$).

To arrive at the optimal λ vector, the authors propose to use a projected subgradient ascent with learning rate γ,

$$\begin{aligned}
q_{j,t} &= \Big(\sum_{i=1}^{n} \Delta^i_{j,t}\Big) - L_{j,t} \\
\lambda'_{j,t} &= \lambda_{j,t} + \gamma q_{j,t}.
\end{aligned} \tag{2.20}$$

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 bound given by the agent policy values from optimizing individual policies using (2.19), 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:

$$\gamma = \frac{V^{\text{UP}} - V^{\text{LOW}}}{||\nabla \mathbf{q}||^2} \tag{2.21}$$

Algorithm 2 puts all the steps together, presenting the complete approach.

Algorithm 2 LDD+GAPS for CMDP M (Agrawal, Varakantham, and Yeoh, 2016).

```
    $\lambda = \infty, \lambda' = 0$
 1: $\forall i : \pi_i^0 \leftarrow$ solveMILP$(M_i, \lambda)$          ▷ Resource-free policies.
 2: $\pi^* \leftarrow \pi^0$                                        ▷ Best joint policy.
 3: while $\lambda \neq \lambda'$ do
 4:     $\lambda \leftarrow \lambda'$
 5:     $\forall i : \langle \pi_i, \Delta^i \rangle \leftarrow$ solveMILP$(M_i, \lambda)$     ▷ Equation (2.19).
 6:     $\pi' \leftarrow$ gaps$(\pi^0, \pi, \Delta)$          ▷ Greedily select feasible policy $\pi'$.
 7:     $V^{\text{UP}} \leftarrow \sum_{j,t}(\lambda_{j,t} \cdot L_{j,t}) + \sum_i \big(V_{\pi_i} - \sum_{j,t}(\lambda_{j,t} \cdot \Delta^i_{j,t})\big)$
 8:     $V^{\text{LOW}} \leftarrow V_{\pi'}$
 9:     $\forall j, t : q_{j,t} \leftarrow \big(\sum_{i=1}^n \Delta^i_{j,t}\big) - L_{j,t}$
10:     $\gamma \leftarrow \frac{V^{\text{UP}} - V^{\text{LOW}}}{||\nabla \mathbf{q}||^2}$
11:     $\lambda' \leftarrow \lambda + \gamma \cdot \nabla \mathbf{q}$
12:     if $V_{\pi'} > V_{\pi^*}$ then $\pi^* \leftarrow \pi'$          ▷ Keep best feasible policy.
13: end while
14: return $\pi^*$
```

Unfortunately, the single-agent optimization problem 2.19 obtained by decomposing the centralized MILP cannot be solved using dynamic programming: the resource costs incurred from 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. Nevertheless, because LDD+GAPS, like Column Generation, splits up planning the single-agent problems from optimizing the global constraint, it obtains similar benefits: 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.

Chapter 3

Resource Preallocation Algorithms

Challenges brought on by the environment may force agents to operate on their own, without access to communication with other agents. The option to autonomous decentralized operation may even be required to guarantee robustness of a system. For example, electricity grid infrastructure may be damaged if demand coordination fails during communication outages. In other domains, continuous all-to-all communication is simply impractical: when each agent runs on a mobile device, such as a smartphone or robot, battery concerns and communication range limits make decentralized operation preferable.

In order to respect resource constraints when agents cannot communicate, it is practical to make agreements on consumption beforehand. Therefore, this chapter studies algorithms which compute resource *preallocations*, with the objective of maximizing expected value during decentralized operation. Such a preallocation can be thought of as a contract which the agents are required to satisfy, specifying how many resources may be used at each point in time during execution. Policies computed for a preallocation are communication-free: because the allocation fully specifies the way the constraint should be shared, an agent never needs to coordinate its consumption with others during execution. Several algorithms to compute decentralized policies for resource-constrained multi-agent problems exist; the previous chapter introduces four such algorithms:

1. a Mixed-Integer Linear Program (MILP),
2. its Lagrangian Dual Decomposition (LDD+GAPS),
3. Constrained Markov Decision Processes (CMDP), and
4. the Column Generation (CG) approach.

Unfortunately, the existing algorithms for computing preallocations suffer from a number of weaknesses. The four preallocation algorithms can be further subdivided into two categories, each with their own strengths and weaknesses:

- MILP and LDD+GAPS compute preallocations which restrict the *worst-case* resource consumption. Restricting the worst-case consumption means that no possible state realization can result in the agents violating the constraints, making their joint behavior *safe*. Unfortunately, both algorithms have exponential worst-case complexity in the number of resources. Additionally, the resulting policies may be significantly conservative, potentially resulting in low expected value.
- CMDP and CG compute preallocations which restrict the *expected* resource consumption, which has polynomial worst-case complexity. However, the resulting policies are stochastic and may *violate* the constraints at execution time.

In this chapter we present techniques to mitigate each of these weaknesses through separate algorithms. Section 3.1 presents a polynomial time heuristic algorithm to compute safe allocations, trading optimality for tractability. Section 3.2 presents an application of the Hoeffding bound to strictly upper bound the probability of resource constraint violations by any maximum violation probability. Because these bounds turn out to be somewhat loose in practice, we additionally propose an iterative bound relaxation algorithm.

An additional challenge occurs when the resource capacity is itself subject to uncertainty. For example, the amount of power produced from renewable sources such as wind turbines is a stochastic quantity (Staid et al., 2017). Similarly, when only a subset of agents participate in a traffic congestion control system, the non-participants contribute to congestion stochastically (De Weerdt et al., 2015). Another source of resource uncertainty may occur when an agent's consumption itself is stochastic (Mausam et al., 2005; Schaffer, Clement, and Chien, 2005). Therefore, in Section 3.3 we investigate how preallocation algorithms can be modified to include shared stochastic resource constraints.

We evaluate the proposed improvements by comparing with the existing allocation algorithms in Section 3.4. These experiments show that our contributions have made allocation algorithms significantly more scalable, and that it is possible to effectively bound the probability of constraint violations by any given maximum probability. Additionally, we show that stochastic resource constraints can be handled effectively by our algorithms, especially when agents are intermittently allowed to communicate. Section 3.5 discusses work which is closely related to the contributions in this chapter. Finally, Section 3.6 concludes the chapter by diving into open challenges to further improve, and apply allocation algorithms in practice.

3.1 Efficient approximation for safe preallocations

Existing algorithms for computing safe preallocations MILP and LDD+GAPS both compute *optimal* allocations by solving mixed-integer linear programming models. These models represent potential resource-to-agent allocations as binary variables, constraining an agent's policy use a resource only if the allocation variable can be set to one. In the worst case, each assignment combination needs to be tested to find the optimal preallocation, giving these algorithms an exponential worst-case complexity in the number of resources. As a result, these methods do not scale beyond relatively small problems, which motivates the need for efficient approximate algorithms.

In this section we propose such an efficient heuristic algorithm for computing a safe preallocation, based on two insights:

1. Under the assumption that agents are weakly coupled by constraints, a resource preallocation *decouples* the agents. Two agents are decoupled if no choice made by the first agent can influence the outcomes of the second. This means that agents can efficiently plan an individual policy which maximizes their own expected utility, subject to adhering to the given allocation.
2. Given a partial resource preallocation, we can efficiently compute the expected value of all *single-step* changes to the allocation. We define a single-step allocation change as any change which enables one more, or one fewer action to be used by the agent.

We use these insights to propose an algorithm which iteratively assigns single-step resource allocations to agents in expected-payoff order.

Before we explain the workings of the algorithm, we must formally define safe preallocations. A preallocation U assigns each of the n agents exclusive access to a subset of the resource constraints $\vec{L}$, giving agent i permission to use *at most* $\vec{U}_i$ resources. A valid (single-agent) policy for allocation $\vec{U}_i$ contains no actions that consume more than $\vec{U}_i$. Recall that $c_i : \{1, \dots, h\} \times S_i \times A_i \to \mathbb{R}^r_+$ is the resource consumption function of agent i, giving the resource demand vector of each action in A_i, dependent on the current time and the state of the agent. If all agent policies adhere to their allocations, and the allocation itself fits in the resource capacity, the result is a safe joint policy on the basis of U. Formally:

$$\begin{gathered}\sum_{i=1}^{n} \vec{U}_i \leq \vec{L}, \text{ and} \\ \forall i.\, \exists \pi_i.\, \forall t, s_i : c_i(t, s_i, \pi_i(t, s_i)) \leq \vec{U}_i.\end{gathered} \tag{3.1}$$

Given an allocation $\vec{U}_i$, agent i is able to compute an optimal policy π_i^* conditional on $\vec{U}_i$, for example through the well-known dynamic programming algorithm with the condition $c_i(t, s_i, a_i) \leq \vec{U}_i$ as filter on the action selection. Intuitively, an optimal policy for $\vec{U}_i$ is the policy which makes the most of the resources the agent is given. Given starting state s_i, this policy obtains expected value $V_{\pi_i^*}[1, s_i]$. Under the model assumption that agents are cooperative and attempting to maximize the linear combination of their rewards, the value of a preallocation U is given by the sum of agents' (conditional) optimal policies rewards,

$$V_U = \sum_{i=1}^{n} V_{\pi_i^*}[1, s_i]. \tag{3.2}$$

An optimal preallocation U^* satisfies $\forall U' : V_{U^*} \geq V_{U'}$.

From the perspective of an agent, changes in its allocation $\vec{U}_i$ are only meaningful if they enable a new action to be taken. Assume that there exists a $\langle t, s, a \rangle$ for which $c_i(t, s, a) > \vec{U}_i$; then, let $\vec{u}$ be the minimal required increase in allocation to allow a, meaning $\vec{u} = c_i(t, s, a) - \vec{U}_i$. If the agent is given allocation $\vec{U}_i + \vec{u}$, it can compute a new conditionally optimal policy π_i^+. Suppose that this increase for agent i was provided for by decreasing the allocation of agent j to $\vec{U}_j - \vec{u}$, then agent j would also need to update its policy to π_j^-. For any optimal allocation U, a necessary condition is that no such allocation swap is profitable, as the following lemma formalizes.

Lemma 1. *Given a resource preallocation U, any positive resource demand vector $\vec{u}$, and any pair of agents i, j, having optimal policies $\pi_i^+, \pi_i^*, \pi_j^*, \pi_j^-$ adhering to*

$$\begin{aligned} c_i\big(t, s_i, \pi_i^*(t, s_i)\big) &\leq \vec{U}_i & \quad c_i\big(t, s_i, \pi_i^+(t, s_i)\big) &\leq \vec{U}_i + \vec{u} & \quad \forall t, \forall s_i \in S_i, \\ c_j\big(t, s_j, \pi_j^*(t, s_j)\big) &\leq \vec{U}_j & \quad c_j\big(t, s_j, \pi_j^-(t, s_j)\big) &\leq \vec{U}_j - \vec{u} & \quad \forall t, \forall s_j \in S_j. \end{aligned}$$

Then, for U to be optimal for initial state prior $\mathrm{P}(s)$, *it must hold that:*

$$\begin{aligned} &\sum_{s_j \in S_j} \Big(\mathrm{P}(s_j) \cdot \big(V_{\pi_j^*}[1, s_j] - V_{\pi_j^-}[1, s_j] \big) \Big) \geq \\ &\quad \sum_{s_i \in S_i} \Big(\mathrm{P}(s_i) \cdot \big(V_{\pi_i^+}[1, s_i] - V_{\pi_i^*}[1, s_i] \big) \Big) \end{aligned} \qquad \forall \vec{u} > 0, \forall i \neq j. \tag{3.3}$$

Proof. Suppose that there exists a pair of agents i and j, and resource demand vector $\vec{u} > 0$ for which inequality (3.3) does not hold. We define modified allocation U', having $\vec{U}_i' = \vec{U}_i + \vec{u}$, $\vec{U}_j' = \vec{U}_j - \vec{u}$ and $\vec{U}_k' = \vec{U}_k$. The expected value of an agent k following policy π_k^*, optimal subject to its preallocation $\vec{U}_k$, is given by

$$\mathrm{E}[V_k] = \sum_{s_k \in S_k} \Big(\mathrm{P}(s_k) V_{\pi_k^*}[1, s_k] \Big).$$

By independence of the agents, the total expected value of preallocation U is

$$\mathrm{E}[V_U] = \mathrm{E}[V_i] + \mathrm{E}[V_j] + \sum_{\substack{k=1,\\ k \neq i,j}}^{n} \mathrm{E}[V_k].$$

Preallocation U' affects (only) the optimal policies of agents i and j, such that

$$\mathrm{E}[V_{i^+}] = \sum_{s_i \in S_i} \Big(\mathrm{P}(s_i) V_{\pi_i^+}[1, s_i] \Big),$$
$$\mathrm{E}[V_{j^-}] = \sum_{s_j \in S_j} \Big(\mathrm{P}(s_j) V_{\pi_j^-}[1, s_j] \Big),$$

resulting in modified expected value

$$\mathrm{E}[V_{U'}] = \mathrm{E}[V_{i^+}] + \mathrm{E}[V_{j^-}] + \sum_{\substack{k=1,\\ k \neq i,j}}^{n} \mathrm{E}[V_k].$$

From optimality of U it follows that $\mathrm{E}[V_U] \geq \mathrm{E}[V_{U'}]$, and thus $\mathrm{E}[V_U] - \mathrm{E}[V_{U'}] \geq 0$,

$$\begin{aligned} \mathrm{E}[V_i] + \mathrm{E}[V_j] + \sum_{\substack{k=1,\\ k \neq i,j}}^{n} \mathrm{E}[V_k] - \Big(\mathrm{E}[V_{i^+}] + \mathrm{E}[V_{j^-}] + \sum_{\substack{k=1,\\ k \neq i,j}}^{n} \mathrm{E}[V_k] \Big) &\geq 0 \\ \Big(\mathrm{E}[V_i] - \mathrm{E}[V_{i^+}] \Big) + \Big(\mathrm{E}[V_j] - \mathrm{E}[V_{j^-}] \Big) &\geq 0 \\ \mathrm{E}[V_j] - \mathrm{E}[V_{j^-}] &\geq \mathrm{E}[V_{i^+}] - \mathrm{E}[V_i] \end{aligned} \tag{3.4}$$

Each side of equation (3.4) contains terms affecting just one agent, which can be taken together to form equation (3.3), because

$$\begin{aligned} \mathrm{E}[V_{i^+}] - \mathrm{E}[V_i] = &\sum_{s_i \in S_i} \Big(\mathrm{P}(s_i) V_{\pi_i^+}[1, s_i] \Big) - \sum_{s_i \in S_i} \Big(\mathrm{P}(s_i) V_{\pi_i^*}[1, s_i] \Big) = \\ &\sum_{s_i \in S_i} \Big(\mathrm{P}(s_i) \Big(V_{\pi_i^+}[1, s_i] - V_{\pi_i^*}[1, s_i] \Big) \Big). \end{aligned}$$

Therefore, if inequality (3.3) does not hold, $\mathrm{E}[V_U] < \mathrm{E}[V_{U'}]$, contradicting the assumption that U is optimal. □

Intuitively, if inequality (3.3) does not hold we can improve the quality of the joint solution by subtracting $\vec{u}$ units of resource from agent j and providing them to agent i, resulting in a higher cumulative expected value over all agents. We propose to use this observation constructively: starting from a given incomplete allocation U, look for the $\langle i, t, s, a \rangle$-combination with the highest increase in $\mathrm{E}[V_U]$, and allocate the required $c_i(t, s, a) - \vec{U}_i$ to agent i. Then, compute π_i^+ and repeat until no more allocations can be made. This algorithm can be seen as an application of the principle of Policy Iteration (Howard, 1960) in a multi-agent, resource constrained setting.

Policy Iteration is one of the famous algorithms for computing optimal policies for general Markov Decision Processes. The algorithm starts from an arbitrary initial policy. It subsequently updates the policy at any point where choosing a different action results in a *myopic* increase in the expected value, i.e. for any state s where the expected value $\arg\max_{a \in A} Q(t,s,a) > Q(t,s,\pi(t,s))$, it sets $\pi(t,s) \leftarrow a$. This process continues until no more such $\langle t,s,a \rangle$ can be found, at which point the optimal policy is found. Of course, in the resource constrained setting, the resource constraints likely stop the process before the optimal (unconstrained) policy is reached. There is no guarantee that the intermediate policy is optimal for the constraints. Nevertheless, we suspect that this process does lead to good preallocations in practice.

Therefore, we propose the following algorithm: we determine for all agents, for every $\langle i,t,s,a \rangle$ that can be allocated within the limit, what the expected benefit of that assignment to the agent's current policy is, starting from the fully unallocated policy. Then we preallocate the most rewarding ⟨agent, resource vector⟩ pair and replan that agent's policy. Subsequently, the algorithm repeats the search for the best pair, until no more resources can be allocated. Finally, if inequality (3.3) is violated by the final allocation, we swap the assignments between the pair of agents and re-plan both. This process terminates in a polynomial amount of time, because the maximum number of improving assignments is $n \cdot h \cdot |S_i| \cdot |A_i|$, in case every possible state-action pair is allocated in turn for all agents. In each iteration, at most 2 agents' policies need to be (re)computed.

The single-agent planning algorithm requires agents to plan $|\vec{u}|$ additional *potential* policies $\pi_{i,\vec{u}}$, one for every potential resource (de)allocation. This requirement can be fulfilled by augmenting standard value iteration with $|\vec{u}|$ additional value tables, one for every resource vector that could be awarded to agent i. This polynomially increases the complexity of the agent planning problem, but it maintains the property that agents can be planned individually.

The pseudo-code for the preallocation algorithm is given by Algorithm 3. The algorithm uses subroutine planAgent to plan an individual agents' (potential) policies, subject to their current allocation. Every iteration, we first check if more resources can be allocated without requiring a swap (lines 4-6). If no more allocations are possible, we check if there are allocations which can be swapped according to inequality (3.3), in which case the best swap is applied (lines 7-10). The algorithm continues to update the preallocation until it stops changing, at which point the policies have converged to a locally optimal preallocation.

3.2 Bounding violation probability of stochastic allocations

The worst-case preallocation algorithms such as discussed in the previous section compute policies which never consume more than the deterministic assignment. This is in contrast to *stochastic* allocation algorithms like Constrained MDPs and Column Generation, which compute stochastic policies that only ensure that their *expected* resource

Algorithm 3 Constrained Factored Policy Iteration for RC-MMDP $\mathcal{M}$

```
 1: $\vec{U}_i \leftarrow 0,$                                                        $\forall i$
 2: $\pi_{i,\vec{u}} \leftarrow$ PLANAGENT$(\mathcal{M}_i, \vec{U}_i)$                $\forall i$
 3: while $\neg$ CONVERGED$(U)$ do
 4:     if $\exists\langle\vec{u}\rangle : \sum_i \vec{U}_i + \vec{u} \leq \vec{L}$ then      ▷ Preallocate more resources.
 5:         $\langle i, \vec{u}\rangle \leftarrow \arg\max\Big(\mathrm{E}[V_{i^+}] - \mathrm{E}[V_i] \Bigm| \sum_j \vec{U}_j + \vec{u} \leq \vec{L}\Big)$
 6:         $\vec{U}_i \leftarrow \vec{U}_i + \vec{u},\ \pi_{i,\vec{u}} \leftarrow$ PLANAGENT$(\mathcal{M}_i, \vec{U}_i)$
 7:     else if $\exists\langle i, j, \vec{u}\rangle : \neg(\text{Eq. (3.3)})$ then      ▷ Swap resource preallocations.
 8:         $\langle i, j, \vec{u}\rangle \leftarrow \arg\max\Big(\big(\mathrm{E}[V_{i^+}] - \mathrm{E}[V_i]\big) - \big(\mathrm{E}[V_j] - \mathrm{E}[V_{j^-}]\big)\Big)$
 9:         $\vec{U}_i \leftarrow \vec{U}_i + \vec{u},\ \pi_{i,\vec{u}} \leftarrow$ PLANAGENT$(\mathcal{M}_i, \vec{U}_i)$
10:         $\vec{U}_j \leftarrow \vec{U}_j - \vec{u},\ \pi_{j,\vec{u}} \leftarrow$ PLANAGENT$(\mathcal{M}_j, \vec{U}_j)$
11:     end if
12: end while
13: return $\langle\pi_1, \ldots, \pi_n\rangle$
```

consumption does not violate the limits. As such, these methods do not provide any guarantees regarding the probability that a resource limit is violated during execution. In this section we introduce methods which improve these algorithms by ensuring that the probability of violating any individual constraint is upper bounded by a given parameter α.

A stochastic policy prescribes for a single agent, for every state, a probability distribution over the actions to be taken. Given the stochastic policy and a probability distribution over states describing the initial state of the model, we can derive a probability distribution over all reachable states and by extension, over actions. Using the resource cost function, this in turn gives us a probability distribution over resource consumption. In a setting with multiple agents, this process produces random variables $X_{i,j,t}$ describing the stochastic resource consumption for each agent i, resource constraint j, and time t.

In situations where resource constraints are *soft*, (occasional) constraint violations are allowed. This means in practice that there are no (reward or transition) consequences for the agents when their cumulative realized resource consumption exceeds the resource limit. Under this assumption, the $X_{i,j,t}$ variables are independent between agents, because each agent can *execute* their policy unconditionally, and without communication. Therefore, by independence we can compute the total resource consumption of resource j at time t as the sum of the agents' consumption,

$$X_{j,t} = \sum_{i=1}^{n} X_{i,j,t}.$$

The stochastic allocation algorithms guarantee that

$$\mathrm{E}\left[X_{j,t}\right] \le L_{j,t}, \quad \forall j, t.$$

In practical applications, even when constraints are soft, exceeding them typically incurs some cost to the system operator, such as traffic jams in the case of a road network constraint, or increased wear from overheating when exceeding capacity of a power grid element. Therefore, even when we are using stochastic allocation algorithms, we would like to restrict the probability that $X_{j,t}$ exceeds the limit, or

$$\mathrm{P}\left[X_{j,t} > L_{j,t}\right] \le \alpha, \quad \forall j, t.$$

To obtain policies which additionally satisfy the constraint on the tail probability, we propose to impose reduced resource constraints $0 \le L^*_{j,t} \le L_{j,t}$ which result in more conservative policies. Setting the reduced limit $L^*_{j,t}$ can be seen as part of the optimization problem of finding the optimal constrained policies. Suppose that we have access to a function that computes this tail probability, B. Then we obtain the problem

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} \mathrm{E}[V_{\pi_i}] & \\ \text{s.t.} \quad & \sum_{i=1}^{n} \mathrm{E}[c(\pi_i)_{j,t}] \le L^*_{j,t} & \forall j, \forall t \\ & B\big(L^*_{j,t}\big) \le \alpha & \forall j, \forall t \\ & 0 \le L^*_{j,t} \le L_{j,t} & \forall j, \forall t \end{aligned} \tag{3.5}$$

We prove that to solve equation (3.5), we must maximize $L^*_{j,t}$

Lemma 2. *Given limits $K' < K$, it holds for policies π and π' optimal subject to*

$$\sum_{i=1}^{n} \mathrm{E}[c(\pi_i)] \le K, \qquad \sum_{i=1}^{n} \mathrm{E}[c(\pi'_i)] \le K'$$

that

$$\sum_{i=1}^{n} \mathrm{E}[V_{\pi'_i}] \le \sum_{i=1}^{n} \mathrm{E}[V_{\pi_i}].$$

Proof. Suppose that $\sum_{i=1}^{n} \mathrm{E}[V_{\pi'_i}] > \sum_{i=1}^{n} \mathrm{E}[V_{\pi_i}]$. We know that policies π and π' are optimal subject to their constraint. However, we also know that

$$\sum_{i=1}^{n} \mathrm{E}[c(\pi'_i)] \le K' < K,$$

and because there are no other constraints on the policies, policy π' is admissible for constraint K. This contradicts that policy π is optimal subject to K. □

Because the action set and the horizon are both finite for the problems we consider, variables $X_{i,j,t}$ are guaranteed to be bounded. Therefore, we can apply either the Chernoff bound (1952) or Hoeffding's inequality (1963), for binary- or real-valued $X_{i,j,t}$ respectively, to compute an upper bound on the tail probability $\mathrm{P}\left[X_{j,t} > L_{j,t}\right]$. Therefore, we use the appropriate bound to find the maximum $L^*_{j,t}$ for which

$$\mathrm{P}\left[X_{j,t} > L_{j,t} \mid \mathrm{E}\left[X_{j,t}\right] \leq L^*_{j,t}\right] \leq \alpha, \quad \forall j, t.$$

As for many problems the assumption of binary consumption is too restrictive, we restrict attention to Hoeffding's inequality, for which Section 3.2.1 presents the application of the bound to determine $L^*_{j,t}$ for any α. Subsequently, in Section 3.2.2 we demonstrate that the bound may significantly overestimate the tail probability, motivating an algorithm to dynamically relax $L^*_{j,t}$ to tightly meet limit α.

3.2.1 Hoeffding bound

When the policies stochastic consumption is real-valued, we can use Hoefding's inequality to bound the probability that the sum of the random variables exceeds $L_{j,t}$ (Hoeffding, 1963).

Theorem 1. *Given a resource type j and a timestep t for which the reduced limit is defined by*

$$L^*_{j,t} = L_{j,t} - \sqrt{\frac{\ln(\alpha) \cdot \left(\sum_{i=1}^{n} \left(c_{\max,i,j}\right)^2\right)}{-2}}, \tag{3.6}$$

it holds that $\mathrm{P}\left(X_{j,t} > L_{j,t} \,\middle|\, \mathrm{E}[X_{j,t}] \leq L^*_{j,t}\right) \leq \alpha$.

Proof. Without loss of generality we assume that $\mathrm{E}[X_{j,t}] = L^*_{j,t} - \theta = L_{j,t} - \hat{L}_{j,t} - \theta$ for $\theta \geq 0$. Hoeffding's inequality provides a bound on the probability that the sum of n independent random variables deviates from its expectation (Hoeffding, 1963). We use the relation $L_{j,t} = \mathrm{E}[X_{j,t}] + \hat{L}_{j,t} + \theta$ and Hoeffding's inequality to derive the following:

$$\begin{aligned}
&\mathrm{P}\left(X_{j,t} > L_{j,t} \,\middle|\, \mathrm{E}[X_{j,t}] = L_{j,t} - \hat{L}_{j,t} - \theta\right) \\
&= \mathrm{P}\left(X_{j,t} > E[X_{j,t}] + \hat{L}_{j,t} + \theta\right) \\
&= \mathrm{P}\left(X_{j,t} - E[X_{j,t}] > \hat{L}_{j,t} + \theta\right) \\
&\leq \exp\left(\frac{-2 \cdot \left(\hat{L}_{j,t} + \theta\right)^2}{\sum_{i=1}^{n} \left(c_{\max,i,j}\right)^2}\right) \leq \exp\left(\frac{-2 \cdot \left(\hat{L}_{j,t}\right)^2}{\sum_{i=1}^{n} \left(c_{\max,i,j}\right)^2}\right).
\end{aligned} \tag{3.7}$$

To ensure that the probability is upper bounded by α, we need to find a reduction $\hat{L}_{j,t}$ for which it holds that $\exp(-2 \cdot (\hat{L}_{j,t})^2 \,/\, \sum_{i=1}^{n} \left(c_{\max,i,j}\right)^2) = \alpha$. Rewriting this equality yields the term $\hat{L}_{j,t}$ that is subtracted in Equation 3.6. □

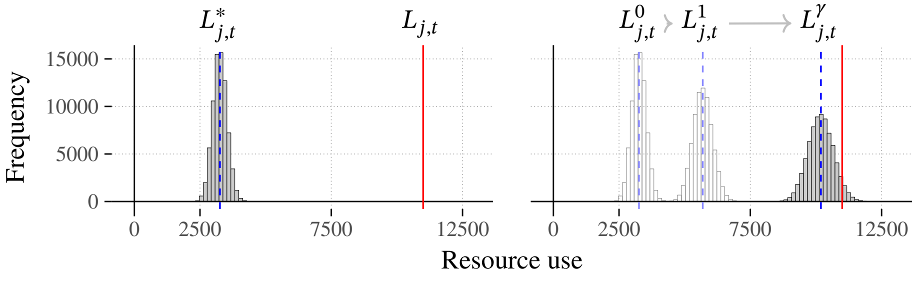


Figure 3.1: Initial Hoeffding bound limit $L^*_{j,t}$ (left) is relaxed over γ iterations (right) to meet the target maximum constraint violation probability $\alpha = 0.05$, on the basis of an empirical estimate of the distribution.

3.2.2 Dynamic bound relaxation

Hoeffding's inequality bounds the constraint violation probability from above, and therefore the resulting resource constraint may be too conservative. This is due to the fact that the inequality defines a general bound on the probability that the sum of n independent random variables deviates from its expectation, regardless of their distribution. In practice this means that the bound can be relatively loose. We propose a dynamic constraint relaxation technique which adjusts the reduced resource limit $L^*_{j,t}$ on the basis of empirical evidence of actual violations during simulation. Our technique adapts the resource limits until the joint policy corresponds perfectly with the desired violation probability.

We start with reduced resource limits $L^*_{j,t}$ obtained using Hoeffding's inequality, as shown in Equation 3.6, and compute a policy for each agent, which can be done using either CMDPs or Column Generation. After obtaining the policies our algorithm runs m Monte Carlo trials to obtain an estimate of the probability distribution of the actual resource consumption. Figure 3.1 (left) illustrates such a sampled distribution (on an instance of the advertising domain presented in Section 3.4.2), which also illustrates that the resource limit reduction obtained using Hoeffding's inequality can be very conservative. After estimating the distribution we determine the limits $L^*_{j,t} \leq \tilde{L}_{j,t} \leq L_{j,t}$ for which αm violations of the true limit $L_{j,t}$ occur, as illustrated in Figure 3.1 (right). In practice the distribution may change significantly by increasing the limit. Hence, we propose to iteratively relax the resource constraints as follows:

$$\begin{aligned} L^0_{j,t} &= L^*_{j,t}, \\ L^{\gamma+1}_{j,t} &= (L^\gamma_{j,t} + \tilde{L}_{j,t})/\beta. \end{aligned} \tag{3.8}$$

where $\tilde{L}_{j,t}$ is determined based on Monte Carlo trials. The auxiliary variable γ represents the iteration index and is used to keep track of the previously chosen resource limit. The parameter β controls the speed of the convergence.

The algorithm starts iteration $\gamma = 0$ with the resource limits $L^0_{j,t}$ obtained using Equation 3.6. Based on these limits it computes a policy for each agent, after which it executes the Monte Carlo trials to obtain the limits $\tilde{L}_{j,t}$. Finally, the relaxed resource limits $L^{\gamma+1}_{j,t}$ are computed. When starting iteration $\gamma + 1$, the algorithm computes policies based on the newly obtained resource limits, after which the entire procedure starts again. This process is repeated until the policies meet the desired violation tolerance α.

Approaching the ideal resource limit from below instead of from above is preferable because in the worst case the resource limit equals the limit obtained using Hoeffding's inequality. Additionally, by relaxing constraints the LP solution obtained in the previous iteration remains feasible, allowing for efficient warm restarts. This is an anytime algorithm because we never tighten constraints, and thus every iteration can only improve the expected value of the policies.

3.3 Preallocating stochastic resource constraints

The models on which the discussed preallocation algorithms apply assume that the planner knows exactly how much resources are available in each time step. In several practical situations, the available resource capacity may not be known exactly beforehand beyond a forecast. Therefore, this section investigates how this assumption can be relaxed to allow models with a *stochastic* resource constraint based on a forecast.

3.3.1 Modeling forecasts as stochastic resource constraints

One prominent domain where resource constraints must be forecast is future energy system control. The integration of renewable energy sources such as wind and solar power generators makes the available power production capacity dependent on the weather, and therefore volatile. As such, the demand side may need to accommodate for supply side fluctuations through the use of buffers like batteries and thermal inertia. Unfortunately, it is not yet possible to predict weather perfectly even on short (day-ahead) time-scales. Therefore controllers of such buffers should take into account multiple statistical forecast scenarios (Pinson et al., 2009; Staid et al., 2017). Such scenarios can be represented by a Markov chain defined over power production outcomes, forming the model of the stochastic resource constraint.

Another motivation for stochastic constraints is to model environments where not all agents participate in the control system. When a system such as a route planning application is introduced, it starts with only a small subset of users. Nevertheless, to avoid congestion and route its users intelligently, it must take into account the intentions of all the agents in the environment (De Weerdt et al., 2015). If we know what policy the other agents will follow, and assuming all agents have similar models, we can aggregate their state-action trajectories into a Markov chain of forecast resource consumption levels by the uncontrolled agents. By subtracting the realization from the total available capacity we obtain a stochastic resource constraint over the remaining capacity.

In both cases, we could collapse the computed Markov chain to its expectation in each time step, to obtain a planning problem with a fixed constraint, which the preallocation algorithms can solve directly. However, doing so would result in policies which make two-sided errors: if the realized constraint is less than the expectation, the policy is likely to cause a violation, while if the realized constraint is more than the expectation, the policy will leave resources unused. In addition, knowledge of the current constraint-state may inform which future scenarios are more likely, allowing the planner to anticipate on future constraint realizations. Therefore, we expect that taking into account the model of stochasticity in the preallocation will result in policies which are both significantly safer and which obtain significantly higher expected value.

In Section 3.3.2 we show how stochastic constraints can be decoupled, allowing them to be tackled by the preallocation approaches studied in this chapter. Section 3.3.3 then explains how the algorithms must be modified to take into account the correlation between the agents induced by the state transition of the global constraint. Finally, Section 3.3.4 shows how preallocations can be adapted to the realized states when agents are able to intermittently coordinate their policies through re-planning. Later, Section 3.4.5 evaluates the benefits to the solution quality from incorporating stochastic constraints, while Section 3.4.6 explores the impact of intermittent re-planning when faced with stochastic constraints.

3.3.2 Decoupling agents subject to stochastic constraints

In the previous section, we motivated the presence of stochastic constraints modeled through a Markov chain of constraint states. Formally, we use S_L to indicate the state space of the resource limit, and let $T_L : S_L \times S_L \to [0,1]$ describe the exogenous transition probabilities over this space. Since all agents must adhere to the same constraint, the transition function of the stochastic constraint threatens to couple the agents together. Fortunately, Becker et al. (2004) show that transition independence is not violated when there exist shared external features, which are a part of the state space that agents cannot affect themselves. As such, the stochastic constraint problem can also be *decomposed* into n single-agent sub-problems, which we propose to do by augmenting the state space of each agent with the current limit (captured in the state feature S_L), so that the sub-problem of agent i becomes a tuple $\langle \bar{S}_i, A_i, \bar{T}_i, \bar{R}_i, h \rangle$ with the following components:

$$\begin{aligned} \bar{S}_i &= S_L \times S_i, \\ \bar{T}_i\big(\langle s_L, s_i \rangle, a_i, \langle s'_L, s'_i \rangle\big) &= T_L(s_L, s'_L) \cdot T_i(s_i, a_i, s'_i), \\ \bar{R}_i\big(\langle s_L, s_i \rangle, a_i\big) &= R_i(s_i, a_i). \end{aligned} \tag{3.9}$$

The global constraint that the agents must adhere to when the resource state is s_L is given by the resource limit function $L(t, s_L)$. Finally, the resource consumption function now also depends on the realized constraint state s_L, resulting in $c_i(t, s_L, s_i, a_i)$ as the consumption function of agent i.

Intuitively, this decomposition states that each agent is able to observe the phenomenon influencing their collective resource constraint, in addition to their own local state. By merging the constraint state into their individual state space, each agent is able to condition their own policy on their shared observations (Becker et al., 2004). In the power grid example, all the agents would receive the weather predictions, and have access to a wind speed sensor. This transformation polynomially increases the size of all MDPs, provided that the number of limit realizations is not itself exponential in the number of agents.

3.3.3 Computing preallocations for models with stochastic constraints

With the sub-problems decoupled, it becomes possible to apply the preallocation algorithms considered in this chapter. For the deterministic preallocation algorithms (MILP, LDD+GAPS and CPI) the working of the algorithms does not need to be changed, as the allocation of resources under one realization of the constraint covers the worst-case consumption of an agent in that realization, independently from the allocation under a different realization. Of course, the presence of stochastic constraints does significantly increase the number of constraints in the system. This is especially significant for the MILP-based approaches, which face exponential complexity in the number of constraints. Therefore, we expect the scalability of the proposed CPI algorithm to be even more important in computing solutions for stochastic constraints.

On the other hand, special care must be taken when constraining the expected consumption, as done by the stochastic preallocation algorithms: the expected value of an agent's resource consumption is distributed over all the realizations of the constraint, but all agents see the same realization in practice. As such, the methods must be adapted to factor in this demand correlation. The challenge in the CMDP LP case is to account for the fact that only one out of $|S_L|$ constraints will be 'active' at any time. By transforming the individual agent problems as defined in equation (3.9), we keep track of the active constraint through the state of the agents. Of course, the sum of all occupancy variables relating to a limit s_L will only sum to the unconditional probability that limit state s_L will be reached at time t. Therefore the consumption limit s_L must be normalized to the probability it will be reached, defined as

$$\begin{aligned} \bar{C}(1, s_L) &= \mathrm{P}_0(s_L), && \forall s_L \in S_L \\ \bar{C}(t+1, s'_L) &= \sum_{s_L \in S_L} \Big(\mathrm{P}(s'_L \mid s_L) \cdot \bar{C}(t, s_L) \Big). && \end{aligned} \tag{3.12}$$

Putting it all together, we obtain the linear program presented in Algorithm 4.

Algorithm 4 Constrained MDP LP for stochastic constraints.

$$\max \sum_{i=1}^{n}\sum_{t=1}^{h}\sum_{\bar{s}\in\bar{S}_{i,t}}\sum_{a\in A_i} x^i_{t,\bar{s},a}\cdot \bar{R}_i(\bar{s},a) \tag{3.10}$$

$$\text{s.t.} \sum_{a'\in A_i} x^i_{t+1,\bar{s}',a'} = \sum_{\bar{s}\in\bar{S}_i}\sum_{a\in A_i} x^i_{t,\bar{s},a}\cdot \bar{T}_i(\bar{s},a,\bar{s}') \qquad \forall i,t,\bar{s}'\in\bar{S}_i$$

$$\sum_{a\in A_i} x^i_{1,\bar{s},a} = T_{1,i}(\bar{s}) \qquad \forall i,\bar{s}\in\bar{S}_i \tag{3.11}$$

$$\sum_{i=1}^{n}\sum_{s_i\in S_i}\sum_{a\in A_i} x^i_{t,\langle s_L,s_i\rangle,a}\cdot c_i(t,s_L,s_i,a)\le \bar{C}(t,s_L)\cdot L(t,s_L) \qquad \forall t,s_L$$

$$0\le x^i_{t,\bar{s},a}\le 1 \qquad \forall i,t,\bar{s},a$$

3.3.4 Replanning to exploit intermittent communication

Preallocation algorithms compute policies under the assumption that agents are unable to communicate during policy execution. This is also an advantage in domains where agents could technically communicate, as it allows the system to operate as normal even when the communication channel is briefly lost. However, we would like to incorporate new information whenever agents do have an opportunity to communicate. For such settings an approach is needed where coordinated policies are computed for a number of sequential decisions that are taken without further communication. Therefore we propose to perform occasional re-planning of the preallocation, using the previously described algorithms as subroutines.

Let $\hat{h}\le h$ be the maximum time that agents may need to operate without communication, and let time t_c be any time step in which communication is possible, and at which point the agents are in state $\bar{s}_\mathrm{c}$. Then, we adapt the algorithms as follows: the algorithm objective functions (e.g. Eq. (3.10)) are changed to range over the time from the communication point until the next sync is guaranteed to happen,

$$\sum_{i=1}^{n}\sum_{t=t_\mathrm{c}}^{\min(t_\mathrm{c}+\hat{h},h)}\sum_{\bar{s}\in\bar{S}_{i,t}}\sum_{a\in A_i} x^i_{t,\bar{s},a}\cdot \bar{R}_i(\bar{s},a),$$

while the initial conditions (e.g. Eq. (3.11)) are set to match the current joint state,

$$\sum_{a\in A_i} x^i_{t_\mathrm{c},\bar{s}_{\mathrm{c},i},a} = 1, \quad \forall i.$$

3.4 Experimental Evaluation

In the previous sections we looked at several adaptations to existing preallocation algorithms. In upcoming sections we test the performance impact of Constrained Policy Iteration and Bounded Violation Probability in turn, against our expectations on their effect on either scalability or resource constraint violation probability. Subsequently, we compare several state-of-the-art algorithms for computing deterministic and stochastic allocations directly, in order to explore the trade-off between optimality, violation probability and scalability. This allows us to make recommendations on which algorithm to use for different types of problems. However, before diving into the experiments themselves, the next sections explain our experimental methodology and the problem domains we test on.

3.4.1 Methodology

To compare the performance of preallocation algorithms, we consider three metrics:

- The expected value of the policy, for which higher values correspond to better policies,
- the probability with which a policy recommends resource constraint violating actions, for which a lower value corresponds to a safer policy, and
- the time required to compute a policy, for which a lower value corresponds to a more efficient algorithm.

All three metrics are affected by the size and characteristics of the problem instance, and therefore we report these values as averages or distributions over several randomly generated instances of a given problem type and dimension.

Unfortunately, the resource constraint violation probability cannot be determined efficiently from a given factored policy, because it is a function of the joint state and action of the agents. As such, computing the true violation probability has exponential complexity in the number of agents. Therefore, we measure the violation frequency by sampling the Markov Chains induced from agents' individual MDP and policy. We report the *maximum* violation frequency over all constraints, because the violation probability depends on the tightness of the constraint: constraints which are slack may never be violated, even by the most aggressive policy.

When measuring the expected value of the policy we are usually not interested in the numeric value itself, but rather in the value relative to optimal, or to another algorithm. Therefore we normalize the expected value whenever possible.

All reported results are obtained on algorithms implemented in Java 8, using Gurobi 7 as library to solve the (MI)LP models, on a laptop machine equipped with a 2.1Ghz quad-core i7 and 16 GB of memory. The authors of LDD+GAPS, Agrawal, Varakantham, and Yeoh, graciously provided their implementation of the algorithm. For the implementation of Column Generation, we used an efficient column handling heuristic to ensure that the master linear programs to solve remain small.

3.4.2 Problem Domains

To test the algorithms under different types of problems, we use three problem domains inspired by real-world problems and a toy problem domain designed to be challenging for preallocation algorithms. Problem domains Mars Rover (Wu and Durfee, 2010) and Advertising (Boutilier and Lu, 2016) are proposed to evaluate the preallocation MILP and budgeted MDPs, respectively. We developed the problem of planning Thermostatically Controlled Loads (De Nijs, Spaan, and De Weerdt, 2015), as well as the toy problem of buying and redeeming tickets in a Lottery (De Nijs et al., 2017). Reference implementations of the real-world domains can be found in our open-source toolbox[1].

Mars Rover (Maze)

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

The grid-world of a vehicle consists of $m \times m$ cells, which represent the states that the vehicle can be in. Only a subset of the cells are traversable, the other cells act as walls. Traversable cells may additionally have a task to be completed, which is known to the agents at plan time. Beyond the locations in the grid, agents also have an 'expired' state, which is reached when the vehicle breaks down.

Agents have access to actions *wait*, *move*, and *do task*. The *move* action is separated into 8 actions, representing the 4 cardinal directions of movement and 2 modes of operation (*safe* and *normal*). Safe movement requires the use of a location-specific resource, but results in a higher probability of reaching the destination and a lower chance of expiring compared to normal movement. The normal movement on the other hand has uncertainty in where the agent will arrive next, and has a much higher risk of breaking the vehicle.

For the resource model, we use the model where the agents are allowed to change their resource consumption in every time step. This means that resources are effectively renewable, and corresponds to the unlimited phase shifting model of Wu and Durfee.

Synthetic Advertising (Ads)

The synthetic advertising domain from (Boutilier and Lu, 2016) models a web-browsing setting where the agents correspond to individual visitors to websites connected to an advertising system that intends to serve ads to users with the highest potential for conversion, subject to the advertising budget of the advertiser.

[1] Found at `https://github.com/AlgTUDelft/ConstrainedPlanningToolbox/`

As such, each agent has 15 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. Global reward is obtained when the agent moves to 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 progressively 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.

Thermostatically Controlled Loads (TCLs)

A thermostatic load is any device that uses (electric) power for the heating or cooling of a body in relation to the outdoor temperature, such as used in refrigerators or central heating systems. The goal of the thermostat is to operate the device such that the temperature of the body remains as close as possible to a given setpoint at all times.

A Markov model of a controlled temperature process is presented in (Mortensen and Haggerty, 1988). In their model, the amount a_i by which the previous temperature is maintained depends on the thermal insulation of the agent i. Given the length of the time-step Δ in hours, the thermal resistance R_i (°C / kW) and capacitance C_i (kWh / °C), the value of a_i is $\exp \frac{-\Delta}{R_i C_i}$. The temperature inputs are given by the outside influences, the current outside temperature θ_t^{out} and a temperature input from the device θ_i^{pwr} that is controlled by the binary control variable $m_{i,t}$. Finally, the temperature transition is affected by a random temperature shift $\theta_{i,t}^{\text{rnd}}$ modeling exogenous actions such as opening a door or window. This results in the model:

$$\theta_{i,t+1} = a_i \theta_{i,t} + (1 - a_i)\left(\theta_t^{\text{out}} + m_{i,t}\theta_i^{\text{pwr}}\right) + \theta_{i,t}^{\text{rnd}}. \tag{3.13}$$

To turn this continuous transition function into a discrete function we discretize the bounded range of $[\min_t (\theta_t^{\text{out}}), \theta_i^{\text{pwr}}]$ into $|S|$ non-overlapping temperature range states.

Lottery Problem

The *Lottery* problem has only 1 resource (the prize), which can be used by an agent in the winning state to redeem a large reward. Which agent reaches the winning state is determined by nature, although in expectation only 1 agent will reach the winning state. Off-line coordination on this type of problem is very difficult because the resource demand depends completely on stochastic transitions. Preallocation strategies can only nominate 1 agent as the potential winner, which means that it can perform arbitrarily worse than on-line coordination.

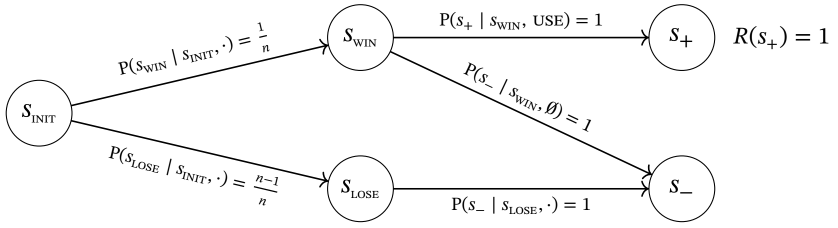

s_{INIT}
$\mathrm{P}(s_{\text{WIN}} \mid s_{\text{INIT}}, \cdot) = \frac{1}{n}$
$\mathrm{P}(s_{\text{LOSE}} \mid s_{\text{INIT}}, \cdot) = \frac{n-1}{n}$
s_{WIN}
s_{LOSE}
$\mathrm{P}(s_+ \mid s_{\text{WIN}}, \text{USE}) = 1$
$\mathrm{P}(s_- \mid s_{\text{WIN}}, \emptyset) = 1$
$\mathrm{P}(s_- \mid s_{\text{LOSE}}, \cdot) = 1$
s_+
s_-
$R(s_+) = 1$

Figure 3.2: Graphical description of the single agent MDP in the *Lottery* problem with n agents.

Figure 3.2 presents the MDP model of an agent in the *Lottery* problem. In the problem, all n agents are identical. The model has 5 states and 2 actions, with the transition probabilities specified in the figure. Unspecified transitions have probability 0 of occurring, and unspecified rewards are set to 0. Because the horizon $h = 2$, states s_+ and s_- are terminal states. The resource limit is such that at most one agent can select action USE.

The goal of the agents is to have one of them use the resource to reach s_+, to distribute the reward across all of them. The problem the agents face is that they cannot know a priori which agent(s) will transition to state s_{WIN} since that is up to chance. All they know is that in expectation, one of them will reach s_{WIN}. An on-line coordination mechanism that assigns the resource after observing the transition result can obtain a maximum value of

$$\begin{aligned} V_{\text{opt}} &= 1 - \left(\frac{n-1}{n}\right)^n, \\ \lim_{n\to\infty} V_{\text{opt}} &= 1 - \frac{1}{e}. \end{aligned} \tag{3.14}$$

In contrast, preallocating the resource to one arbitrary agent obtains a value of

$$\begin{aligned} V_{\text{pre}} &= \frac{1}{n}, \\ \lim_{n\to\infty} V_{\text{pre}} &= 0. \end{aligned} \tag{3.15}$$

Stochastic allocations that require the constraint to be met only in expectation can obtain value

$$V_{\text{sto}} = 1, \tag{3.16}$$

but risk a violation when the random variable X representing the number of agents in

s_{WIN} attains value greater than 1, which has a probability of

$$\begin{aligned} P(X > 1) &= 1 - P(X = 0) - P(X = 1), \\ &= 1 - \left(\frac{n-1}{n}\right)^{n} - \left(\frac{n-1}{n}\right)^{n-1}, \\ \lim_{n\to\infty} P(X > 1) &= \frac{e-2}{e}. \end{aligned} \tag{3.17}$$

Coordinated Search and Rescue Missions

For evaluating the stochastic constraint we consider a disaster response cooperative game as an illustrative domain. Consider a group of countries that collectively commits response teams in order to perform expensive search-and-rescue (SAR) operations that would be too costly to perform individually. Due to the urgent nature of crises, each country must individually decide its response level without time-consuming coordination. They do so in accordance with a single time-step policy that they agreed on beforehand (e.g., at the previous summit).

The size of an operation that a country commits determines the cost to that country, while the sum of all committed operations influences the probability of successful rescues. The cost of an operation of size j is simply j, where $j \leq 4$, which we assume to be the politically acceptable maximum spending on rescue missions. The probability of retrieving a survivor using an operation of size 1 is given by p, which we assume to be 0.2. More generally, the number of survivors i rescued, as a function of the sum of operation sizes j is given by random variable W having probability distribution

$$\mathrm{P}\left(W = i \mid j\right) = \binom{j}{i} p^{i}(1-p)^{(j-i)}.$$

The reward for rescuing a survivor is 100.

In practice, the number of survivors that can be rescued is bounded by the number of people affected, which informs the stochastic constraint in this problem. Due to the high value of rescuing survivors, countries are incentivized to deploy all their resources in the first crisis in an uncoordinated setting. To retain some resources for future calamities, countries constrain their response to be sufficient for the size of the disaster. Because the size of an unexpected disaster can only be estimated when the disaster occurs, the number of potential survivors x is learned only at the time mission size must be determined. We assume that the probability density function on the number of potential survivors of any potential disaster is given by

$$\mathrm{P}(X = x) = \{0 : 0.05,\ 1 : 0.4,\ 2 : 0.3,\ 3 : 0.2,\ 4 : 0.05\}.$$

A centralized joint task force (without maximum operation size) would thus aim to optimize the following function $f_x(j)$ for each disaster size x.

$$f_x(j) = \sum_{i=1}^{j}\Big(100 \cdot \mathrm{P}(W = i \mid j) \cdot \min(x, i)\Big) - j.$$

Since this set of functions attains maximum value at $j = \{0, 14, 22, 30, 37\}$, for $x = \{0, 1, 2, 3, 4\}$ respectively, the joint task force should assign operation sizes to countries such that their sum operation size matches these values. However, when the countries do not have time to communicate their commitment, they must select their responses such that the *expected* sum is equal to the optimal. We compare the proposed coordination planning algorithms with versions that condition their response on the mean disaster size:
1) Deterministic preallocation MILP, E(x): mean disaster survival rate is $\approx$ 1.8 survivors; thus a mission size is selected such that the maximum number of survivors is at most 1.
2) Conditional preallocation MILP, P(x): depending on the potential number of survivors x, the mission response size is selected such that exactly x are rescued. 3) Deterministic preallocation CMDP, E(x): a mission size is selected such that in expectation 1.8 survivors are rescued. 4) Deterministic preallocation CMDP, P(x): a mission size is selected such that in expectation, x survivors are rescued.

3.4.3 Experiments to evaluate scalability improvements

We first evaluate the performance obtained by our proposed Constrained Factored Policy Iteration (CPI) algorithm relative to the baseline MILP, in order to evaluate the scalability of the proposed algorithm, as well as any loss in solution quality. For the comparison we use Maze and TCL instances, as results are identical on Lottery, and MILP cannot solve Advertising due to its non-binary consumption function.

Figure 3.3 presents the expected value of the policies computed by CPI and MILP, as well as the wall-clock time required to compute the policies. We observe that as the size increases and more resources are added to the problem, the MILP formulation quickly becomes intractable, as some TCL horizon 16 and Maze $n = 10$ instances could not be solved within 20 minutes. CPI on the other hand is able to find policies with nearly the same quality as MILP in a fraction of the time. This suggests that CPI is a very effective heuristic algorithm in practice.

3.4.4 Experiments to compare preallocation algorithms

This section aims to explore the differences between the two categories of algorithms, as well as to evaluate the impact of stochastic policies planned for limited maximum violation probability. To this end we report the performance of the algorithms on all domains, presenting not only the expected value and wall-clock solve time, but also the observed worst-case violation frequency, which is the frequency with which the

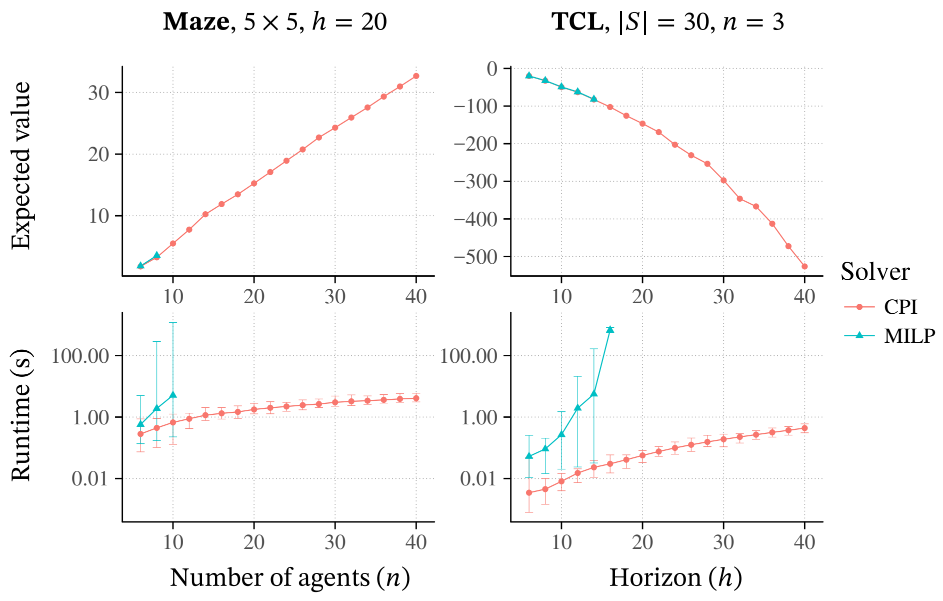


Figure 3.3: Comparison between Constrained Factored Policy Iteration (CPI, Algorithm 3) and MILP (Section 2.3.3), presenting the solution quality in terms of policy expected value, and the wall-clock time required to compute the policy. Error bars indicate minimum and maximum runtimes over 100 different instances, under a 20 minute time-out.

most often violated constraint is exceeded over 500,000 simulations. In the following experiment we use CMDP as our stochastic allocation solver, in order to isolate the relative runtime impact of bounding constraint violations. To explore the scalability of the algorithms as problems become larger, we increase the number of agents while keeping the other dimensions constant.

The leftmost column of Figure 3.4 presents the results of all algorithms on *Lottery*. Expected values are normalized to the CMDP policy, which can return the highest expected value of any allocation algorithm. We observe that the lottery problem is computationally easy to solve even for over 500 agents. As expected, the obtained value decreases as $\frac{1}{n}$ for all deterministic preallocation strategies (MILP, CPI and LDD+GAPS) in an equal manner. We also observe that the Hoeffding bound is very conservative on this problem, resulting in a low value and significantly less violations than the tolerance α. Dynamic constraint relaxation on the other hand is able to obtain constant value with a stable bounded constraint violation probability, both only surpassed by the unbounded CMDP, illustrating the expected trade-off between expected value and number of violations.

The middle and right columns of Figure 3.4 compare the algorithms on the TCL and

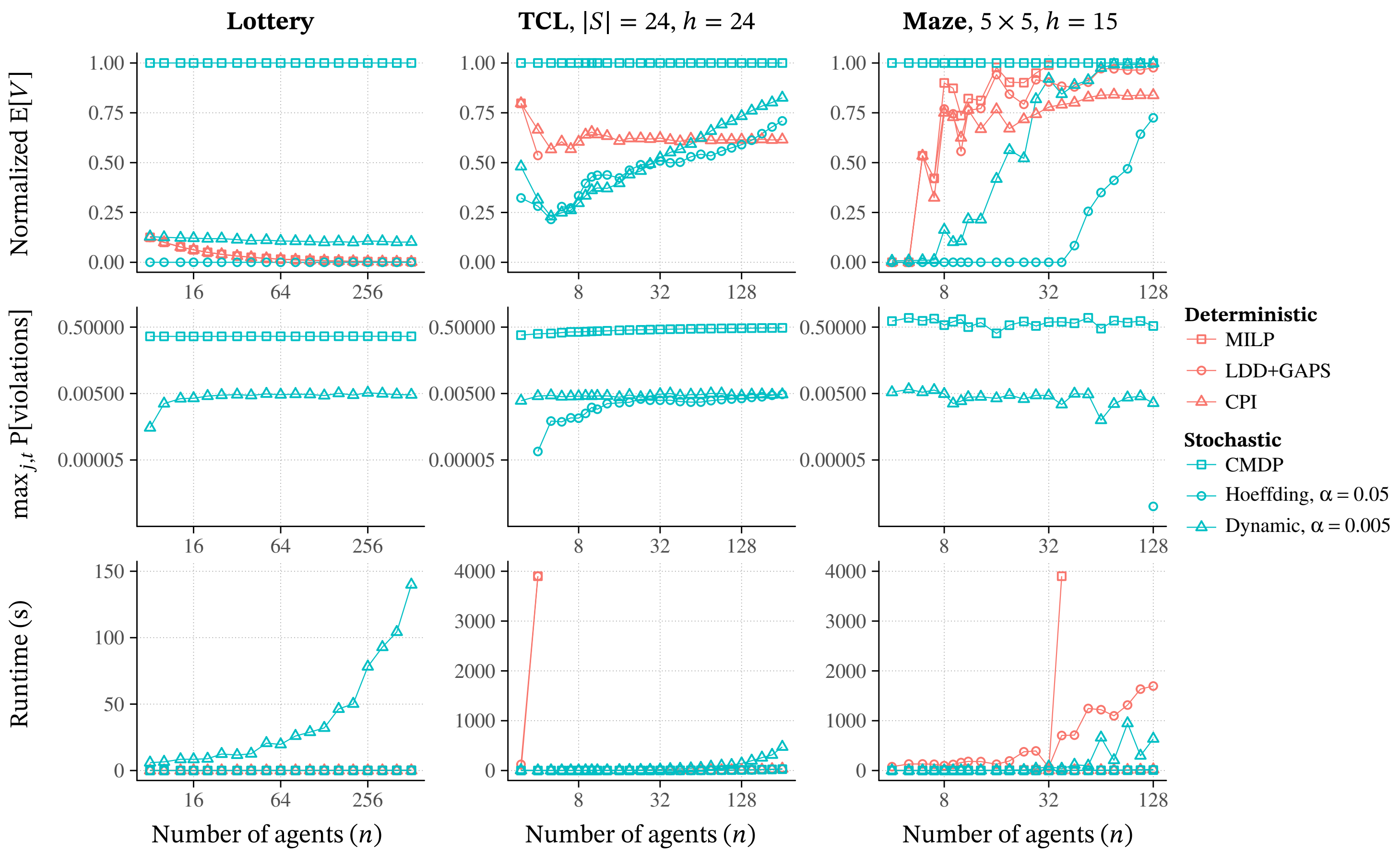


Figure 3.4: Comparison of preallocation algorithm performance on three problem domains. Policy value normalized to CMDP solution. All graphs log x-axis, violations also log y-axis.

Maze problems, respectively. Each TCL agent has 24 states and 2 actions, horizon 24 and 1 resource type (24 resources in total). Our Maze problems have 26 states and 10 actions per agent, horizon 15, and 3 resource types (resulting in 45 resource constraints). We observe that the TCL problem is computationally hard for the preallocation strategies. Both MILP and LDD+GAPS exceed the 1 hour timeout for 4 agents. By contrast, the CPI algorithm is able to find solutions in a few seconds even for the largest instances. The Hoeffding bound is less conservative here, but it still comes an order of magnitude short of the target tolerance. Dynamic constraint relaxation approaches the (stricter) tolerance and additionally obtains a higher value.

In the TCL domain agents are penalized for consuming too many resources, because their temperature would exceed their setpoint. In contrast, for Maze domain agents using more resources is strictly better than using less. This translates into a higher violation probability for the CMDP approach, and an easier problem in general as observed by the better scalability of the preallocation approaches. Nevertheless, for a sufficient number of agents our dynamic constraint relaxation obtains an expected value not significantly lower than the CMDP approach with two orders of magnitude fewer violations.

In conclusion, when comparing stochastic with deterministic preallocations we observe that for large numbers of agents, the values of the bounded approaches tend towards the CMDP value, exceeding the value of the deterministic allocations. When more agents are available to spread the load of the reduced resource limit, their individual rewards are compromised less. Conversely, small problems do not benefit from 'allowing' violations, unless we are willing to accept very high risks.

Large-scale Advertising Domain

Using column generation, we are able to tackle large-scale planning problems, such as the synthetic advertising domain presented by Boutilier and Lu (2016). Their domain consists of assigning advertisement budget to potential customers to maximize the amount of sales, where each individual customer is modeled as an MDP. In this domain the resource constraint is not time-dependent, but applies to all time steps. Because of the scale of the model (1000 agents, each with 15 states and 5 actions), direct application of the CMDP algorithm is intractable. Additionally, the preallocation strategies cannot be applied to this problem because of non-binary resource consumption. However, as Figure 3.5 shows, using Column Generation with pruning it is possible to solve this problem in less than a second (both with and without Hoeffding). We see that the version without constraint reductions violates the selected budget level half of the time. However, the Hoeffding bound is again very conservative in this setting, because of the large potential maximum consumption relative to the expectation. This is addressed by the dynamic constraint relaxation method, which significantly reduces violations compared to the version without constraint reductions. We also observe that there is only a very small reduction in expected value compared to the optimal relaxed solution.

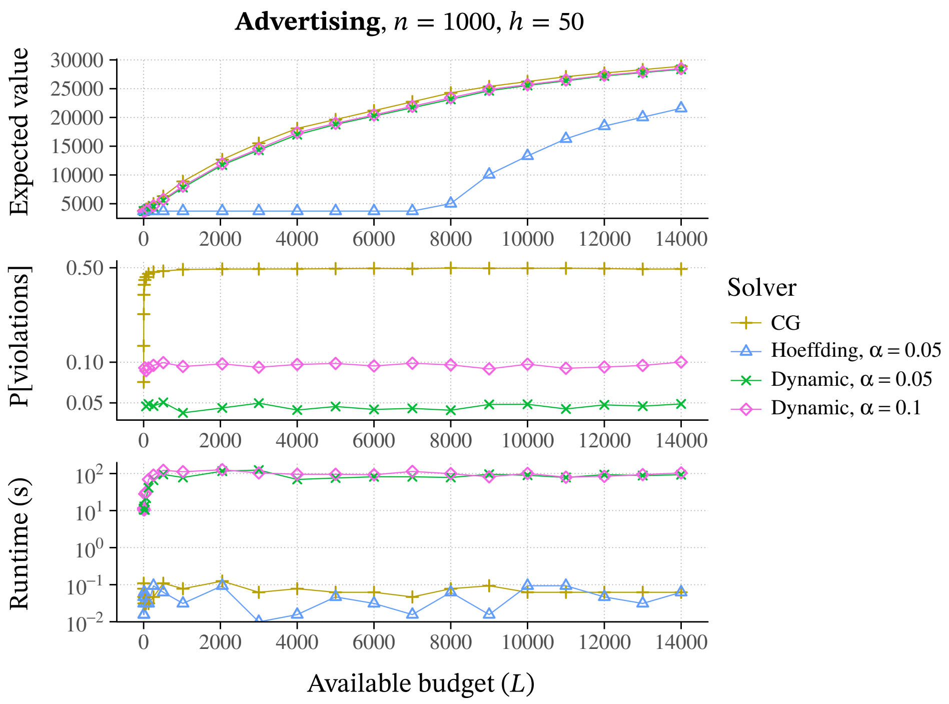


Figure 3.5: Scalability of Column Generation with dynamic constraint relaxation on a large-scale advertising domain.

3.4.5 Experiments on stochastic constraints

Next, we investigate the value of planning for a stochastic resource constraint, starting with an experiment on the search-and-rescue domain. Figure 3.6 presents the results, showing means and standard errors obtained, when each computed policy is sampled 100,000 times. We compute 100 policies per data point to obtain significance with respect to the runtime. The value reported is the *observed* value, given by the number of actual rescues minus the operational costs. As expected, the value obtained when planning for just the mean (results with $\mathrm{E}(X)$) is significantly less than the value obtained through taking into account the uncertainty in X for both algorithms (results denoted by $\mathrm{P}(X = x)$). Additionally, the frequency of deploying more successful operations than there are potential survivors (i.e., overcapacity) is also significantly smaller when planning for $\mathrm{P}(X = x)$ than for $\mathrm{E}(X)$. Planning for the stochastic limit increases the required time to plan policies significantly, but this does not change the scalability characteristics: the trends in the run time depending on the number of agents are the same. Comparing the behavior of the two different algorithms themselves, we observe that the MILP trades off overcapacity probability (i.e., almost none) for slightly reduced

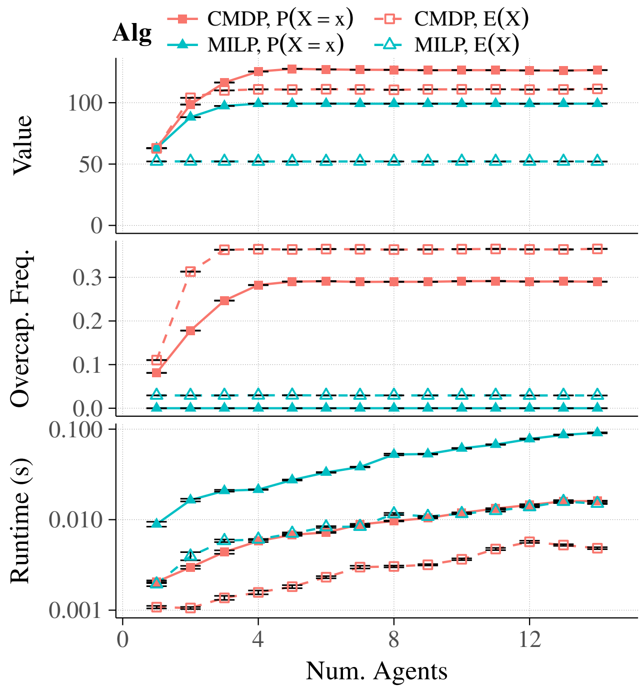


Figure 3.6: Realized mean and standard errors of the value, frequency of overcapacity, and runtime, comparing mean (E(X)) with stochastic limit (P(X)) on SAR problem.

value and more significant runtime costs compared to the CMDP approach.

Next we compare the same methods for planning how thermostatically controlled loads (TCLs) use a shared resource for a longer horizon. Because supply from renewable sources is typically not only fluctuating but also *uncertain*, this domain naturally exhibits stochastic constraints. In these experiments we consider TCL problems with temperature ranges discretized into 25 states, and with agents having 4 actions, corresponding to switching a heater on for $\{0, 5, 10, 15\}$ out of 15 minutes per time step. The thermal parameters are based on reference insulation levels of houses equipped with heat pumps. To model consumer behavior and build quality variation, we add small Gaussian noise to the parameters, resulting in a heterogeneous population of TCLs.

To obtain challenging instances of the TCL problem, we generate resource limit scenarios such that each scenario is *in expectation* sufficient to keep the temperature at the setpoint, but has realizations that are far from the mean. We randomly generate 10 such (deterministic) resource limit scenarios and merge them together in a Markov chain by allowing for a small probability of cross-over between scenarios.

For evaluating the quality of the proposed algorithms, we define an error measure by the distance of the results from a theoretical upper bound, which we obtain by computing

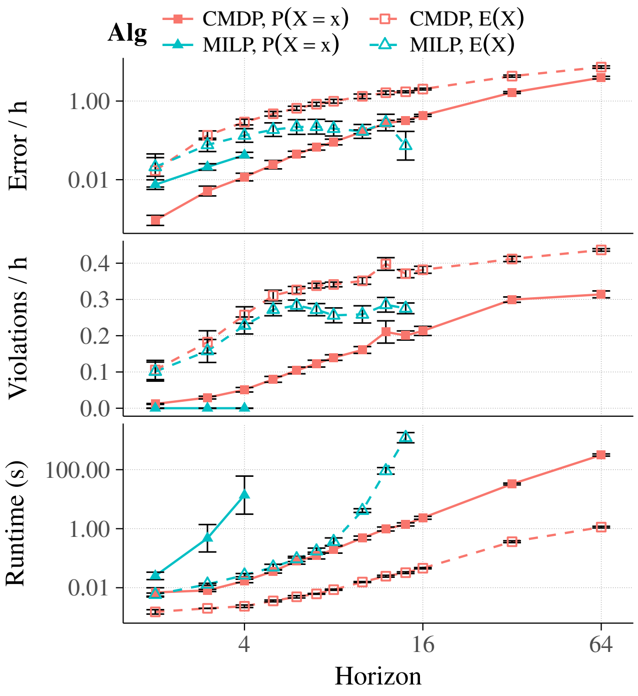


Figure 3.7: Realized mean and standard errors of the absolute error, violation frequency, and runtime, comparing mean (E(X)) with stochastic limit (P(X)) on TCL instances.

joint (centralized) policies with Value Iteration (Puterman, 1994). Because this algorithm has exponential complexity in the number of agents, we perform experiments for 3 agents and an increasing length of the horizon. Figure 3.7 presents the results, normalized by the horizon as each time step has potential to incur error, and each constitutes a new resource constraint that can be violated. The results show a similar trend as in the search-and-rescue instances. Planning for the stochastic resource constraint $\mathrm{P}[X = x]$ increases the runtime of the algorithms as a result of the increase in number of states and constraints. This has the largest effect on MILP, which also has exponential worst-case complexity when planning for the mean constraint. For both algorithms we observe that planning for the stochastic constraint results in a significant increase in the quality of the solution, resulting both in lower error and in lower violation frequency.

Regarding scalability, we observe that the run-time measurements of CMDP form almost a straight line in the log-log plots in both experiments. We therefore conclude that this run time scales polynomially with the number of agents in the SAR domain (Figure 3.6) as well as with the length of the horizon in the TCL domain (Figure 3.7).

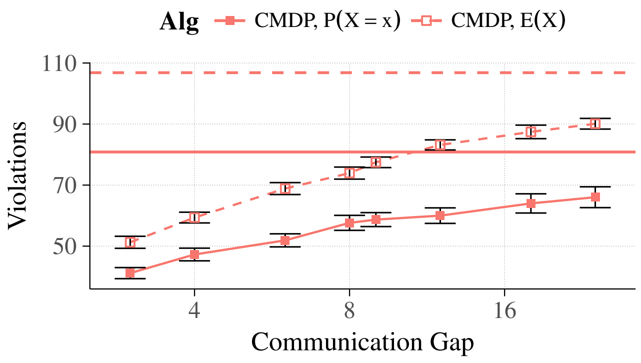


Figure 3.8: Effect of increasing the time between communication in the re-planning algorithm.

3.4.6 Experiments on intermittent re-planning

In order to assess the effect of periodic coordination, we apply the re-planning algorithm to a TCL instance with stochastic constraints, a horizon of $h = 216$ (9 days in hours) and re-planning horizon $\hat{h} = 24$. We let the agents re-plan at a regular interval (the communication gap), and measure the number of violations as a function of the length of the interval. Figure 3.8 shows the results, with the horizontal lines representing the baseline case of coordinating only at the start. We observe that re-planning can greatly reduce the number of violations. However, more importantly, we also observe that planning for stochastic constraints is effective at reducing constraint violations even when agents only need to bridge gaps of 3 steps, demonstrating the practical value of our algorithms.

3.5 Related work

General problems of optimizing a set of decision variables over random variables, subject to constraints which need to be satisfied with a minimum probability are known as chance-constrained programming problems (Charnes and Cooper, 1959). Haskell and Jain (2015) applied chance constraints to perform risk management over the expected value of Markov decision processes.

Luedtke and Ahmed (2008) propose an approximation scheme based on sampling to satisfy chance constraints in general optimization contexts.

Handling stochastic resource constraints has to our knowledge thus far been limited to scheduling under uncertainty, in which case there is only a single agent and a pre-defined set of activities (Fink et al., 2006). Even though stochastic resource constraints are not widely studied, there are several other works that attempt to address deterministic resource constraints through other means than decoupling. Meuleau et al. (1998)

consider large-scale planning problems with instantaneous constraints; however, they only enforce them at execution time, choosing to ignore them in the planning phase. Such an approach would not only overestimate the single agent expected value, it would also be impossible to apply in a decentralized setting, because the on-line enforcement imposes a centralized controller.

The literature on Decentralized (PO)MDPs provides algorithms that exploit the limited influence that agents might exert on each other (Oliehoek, Witwicki, and Kaelbling, 2012). However, our global resource constraints prevent that agents can be easily decoupled using such models. Related to our re-planning algorithm are approaches that consider intermittent communication (Nair et al., 2004) or delayed communication (Spaan, Oliehoek, and Vlassis, 2008; Oliehoek and Spaan, 2012). These methods rely on a solution of the underlying Multi-agent POMDP which is exponentially-sized in the number of agents. Hence, scalability is poor and they are typically only demonstrated for two agents.

3.6 Conclusions and Discussion

We study multiple agents in stochastic domains that need to coordinate their actions off-line due to limited availability of resources and lack of communication. CMDPs and Column Generation algorithms compute policies which satisfy resource constraints in expectation, but these policies provide no guarantees on the probability that constraint violations occur. We therefore propose a new method to bound constraint violation probabilities. Our method is based on Hoeffding's inequality and uses a dynamic constraint relaxation technique to ensure that constraint violation probabilities are tightly bounded by a given tolerance. Policies computed by our method ensure bounded constraint violation probabilities, even if these policies are executed independently without communicating. Since Column Generation has several attractive properties when combining it with our method, we introduced a column pruning technique to accelerate the algorithm. Experiments on hard instances and more realistic problems have shown that our method outperforms two existing state-of-the-art methods for computing deterministic resource allocations.

Studying how constraints on violation probabilities can be encoded directly in a linear program is an interesting future work, because it yields non-convex problem formulations. Secondly, column generation techniques have also been used to solve large security games (Jain et al., 2010), and it remains to be studied if column pruning also further accelerates these and other algorithms based on column generation.

Stochastic resource constraints have not been widely studied in multi-agent planning under uncertainty, although they occur naturally in domains where the resource constraint is a natural process or results from unmodeled external influences. Multi-agent systems are additionally typically expected to operate decentrally for periods at a time, either because replanning time exceeds decision time, or because of communication restrictions. In this chapter we show how stochastic resource constraints can be factored such that policies can still be effectively decoupled. To demonstrate this we extend the preallocation algorithms to additionally handle stochastic constraints. In our experimental evaluation we observe that using our extensions to plan for stochastic constraints results in significantly better solutions than using the original algorithms to plan for the expectation of the limit. We show that these results continue to hold when combined with an intermittent replanning scheme, which allows the system to operate with reduced violations over a longer horizon.

3.6.1 Discussion

Because the preallocation algorithms studied and proposed in this chapter have shown good performance in empirical evaluation, it is worthwhile to consider them as starting point for future extensions and improvements. In this section we provide an overview of future work ideas that make the algorithms more broadly applicable.

Computing fair resource allocations

We assumed in this chapter that the agents are coordinating on extracting the maximum cumulative utility out of the available resources, essentially stating that it does not matter which agent receives the utility. This allows for solutions where all resources and rewards are assigned to only one specific agent, which may be an unacceptable outcome when the agents are acting on behalf of a human operator. Therefore, developing algorithms which can guarantee a 'fair' distribution of utility under constraints is an important challenge for future work.

The first consideration for achieving fair solutions is how to measure the fairness of a solution; Zhang and Shah (2014) suggest a prioritized objective, where in the first place the minimum agent value is maximized, and secondly the cumulative value is maximized using the remaining capacity. This metric can be straightforwardly expressed in a Linear Programming (LP) model such as the Constrained MDP and MILP models presented in Section 2.3, by introducing an additional variable z constrained to not exceed the value of the worst-off agent,

$$\begin{aligned} \max_{x_{i,t,s,a}} \quad & z + \frac{\epsilon}{n} \sum_{i=1}^{n} \sum_{t=1}^{h} \sum_{s \in S_i} \sum_{a \in A_i} x_{i,t,s,a} R_i(s,a) \\ \text{s.t.} \quad & z \leq \sum_{t=1}^{h} \sum_{s \in S_i} \sum_{a \in A_i} x_{i,t,s,a} R_i(s,a) \qquad \forall i \\ & \big(+ \textit{transition and resource constraints} \big) \end{aligned} \tag{3.18}$$

Unfortunately, this technique does not extend directly to the more efficient algorithms developed in this chapter, thus motivating the need for future research.

While this LP-based fairness criterion cannot be applied immediately, we see several opportunities to extend the algorithms developed in this chapter. The CPI algorithm, found in Algorithm 3, assigns resources to agents based on the utility gained relative to the current allocation. This condition could be modified to assign resources first to the least-rewarded agent, and modifying the swap condition such that the resources are only swapped if it does not lower the value of the least-rewarded agent. For the other algorithms, the use of *multi-objective* planning may be a promising route. Multi-objective planners compute multiple policies which together cover any possible weighting of the objectives (Roijers et al., 2013). By treating the resource usage on each constraint as a separate objective, we can obtain the set of all useful policies per agent. Then, we can perform policy selection in similar manner as the Column Generation master LP (2.17), but now using the maximin objective of LP (3.18), to arrive at a fair stochastic mix of policies for each agent.

Optimizing for the consequences of constraint violations

CMDP and CG compute policies which are allowed to violate the resource constraints. When such a violation occurs, the resource-using agents are nevertheless able to continue as if they all received the resource. In domains where constraint violations are not destructive (e.g. higher than intended traffic on a road), the violations do have some consequences (e.g. traffic slowdown) for the agents. Not associating loss incurred from constraint violations actually introduces a paradox where the value of information (e.g. knowing state-trajectory realizations) is negative (Blau, 1974). Unfortunately, these effects of violations cannot be captured in the planning stage, because violations can only be derived from the joint state space. Nevertheless, some existing work provides avenues to incorporate joint effects; agent anonymity investigated by Robbel, Oliehoek, and Kochenderfer (2016) conceptually applies here too, as it is immaterial which other agents are involved in violating the constraint, only the *number* of agents matters. Alternatively, it may be possible to define reward terms over violation *events* (Gupta, Kumar, and Paruchuri, 2018).

Handling equality constraints

Preallocations specify an *upper bound* on consumption. Sometimes, we have objects which must be allocated exactly (e.g. when scheduling tasks, we require all of them to be completed), which requires imposing an *equality* constraint. It is not immediately clear if current algorithms can be adapted to represent such constraints. Guaranteeing to compute a single-agent policy which *always* consumes its allocation requires us to do model-checking as planning (Cimatti, Giunchiglia, and Traverso, 1997). Because an allocation may be unsatisfiable by any policy, the controlling algorithm in the case of CPI or Column Generation, would need to be able to cope with failed allocations.

Alternatives to dynamic bound reduction

In order to satisfy the resource constraints with bounded probability, our dynamic bound relaxation algorithm proposed in section 3.2.2 lowers the total resource availability for the agents. The effect of this choice is to reduce the mean of the resource consumption random variable. Although attractive in its straightforwardness, it neglects to consider options to control higher moments of the variable. If we can reduce the variance of the resource consumption, we could set a higher mean resource availability for the same violation probability. Reducing the probability of low-value outcomes has been studied in the context of risk-averse optimization, where conditional value-at-risk (CVaR) has emerged as suitable measure. Song et al. (2018) show that CVaR can reduce variance in probability of failure (equivalent to a constraint violation) in scheduling for robust makespan.

Additionally, because the algorithm adjusts the constraint based on a posteriori sampled estimates of resource constraint violations, the number of samples required to obtain confidence on the probability of a violation grows rapidly as the probability of the event decreases. This means that, for low α levels, the sample complexity is prohibitive. If the domain requires such strict adherence to the constraints, different estimation techniques need to be used. It may be possible to bias sampling towards the states in which an agent policy is likely to consume resources.

Chapter 4

Dynamic Resource Allocation

The preallocation algorithms studied in the previous chapter have several advantages: the resulting policies are decentralized and can thus be executed without communication, and several preallocation algorithms can guarantee optimality of both the allocation and the policy (or policies) computed for the allocation. Nevertheless, the resulting algorithms also face one major drawback, which is that they are unable to adapt to the realizations of state transitions. Taking the example of planning TCLs, an agent which is allowed to use power at a certain time may end up due to transition uncertainty in a state where its temperature is sufficient to stay off, at which point the resource sits unused.

When agents are able to communicate during policy execution, they are able to inform each other of their expected resource consumption, allowing them to adapt to the effect of stochastic model transitions. In fact, we have shown in Section 3.4.6 re-planning during communication opportunities allow agents to reduce the risk of constraint violations. As such, we consider as a first step to perform rolling horizon re-planning in order to increase the resource efficiency of the solution. Unfortunately, we show in this chapter that the use of re-planning can nevertheless result in arbitrarily poor solutions. Therefore, a different solution paradigm is required to let agents coordinate effectively.

In this chapter we propose two new algorithms that obtain better resource utilization than (re-planning) preallocation algorithms, by making use of the game-theoretic concepts of best-response and fictitious play. Both algorithms make use of a resource *arbiter*, which is a special on-line coordinator to validate the resource consumption of the agents' joint actions. In case of a pending constraint violation, the arbiter modifies the selected action to fit inside the available capacity while minimizing the loss in expected utility. The use of a resource arbiter *decouples* the agents, because from the perspective of an agent, whether or not it succeeds in performing its chosen action only depends on the arbiter's decision.

Because the presence of the arbiter has the potential to change the actions chosen by an agent, the agents must take this into account during planning. We propose two alternative strategies to model the influence of the arbiter:

- The best-response algorithm, which models the influence of the arbiter by the probability that it changes an action a_i into alternative action a_j under the currently computed joint policy. Given these probabilities, each agent can compute its best response by factoring these probabilities into the transition function.
- The fictitious play algorithm, which uses the same principle as the Column Generation algorithm of determining a 'utility cost' of requesting a resource. These costs are computed for a history of previously computed joint policies (the prior plays), which allows the algorithm to reason about potentially reachable joint states.

The organization of this chapter is as follows. First, section 4.1 investigates the worst-case behavior of the preallocation algorithms under a re-planning scheme. Then, section 4.2 presents the concept of a centralized resource constraint arbiter, responsible for optimizing the selected joint action based on the current state. Section 4.3 subsequently shows how agents can compute a best response to the existence of such an on-line arbitrage system. Next, section 4.4 presents our second approach to model the influence of a resource arbiter on the basis of fictitious play. The following section 4.5 then empirically evaluates both algorithms on our testing domains to assess their practical value. Section 4.6 places our proposed algorithms in context with existing and related work. Finally, we conclude the chapter in section 4.7 with a discussion on the practical merits of the proposed algorithms.

4.1 Theoretical analysis of re-planning preallocations

When constraints are preallocated to agents with stochastic state transitions, the resulting uncertainty may leave resources unused, or overused in the case of preallocations which satisfy the constraints in expectation. When agents can coordinate after each state transition, we may be able to update the resource allocation based on agents' current needs. The best we can do is to obtain a new optimal preallocation for the realized state by re-planning. We have shown that re-planning is effective in reducing uncertainty under a stochastic constraint in Section 3.4.6, resulting in lower probability of constraint violations when re-planning more frequently. In fact, re-planning every time step may prevent constraint violations entirely: because the current state is deterministic, the selected actions' consumption is also deterministic. Randomization would only be applied in the first step if the (remaining) capacity is less than the consumption of an agent's best action.

Unfortunately, re-planning can make the policy solution quality arbitrarily worse than the optimal solution in the case of stochastic resource preallocations. In the case of deterministic preallocations, the use of re-planning may fail to improve the usage of

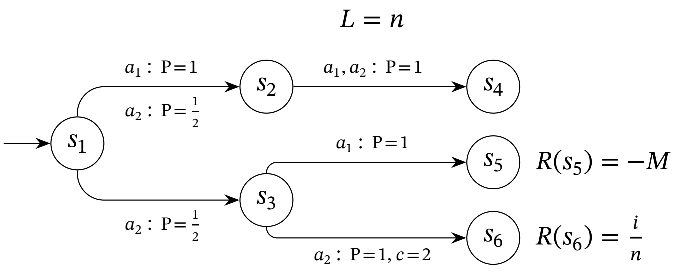


Figure 4.1: Single-agent MDP model with horizon 2, 6 states, 2 actions and transition, reward, consumption and limit functions as indicated by the figure, annotated on the transition. Omitted functions have value 0. Variable i stands for the agent id, n is the number of agents, and M is a large positive constant.

the allocation beyond the original solution, which can also be arbitrarily worse than the optimal solution. We prove both claims through analysis of hand-crafted examples on which re-planning performs worst.

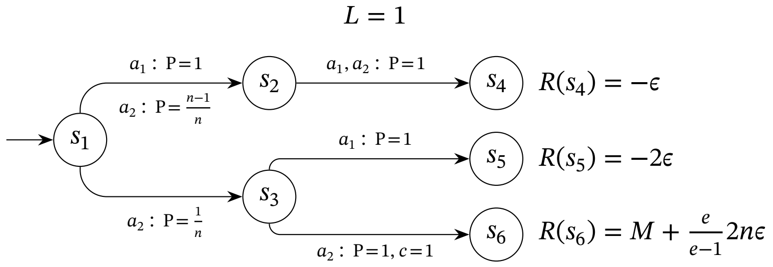
4.1.1 Worst-case analysis of replanning stochastic preallocations

Suppose we are given a problem containing n agents, all modeled by the MDP presented in figure 4.1. From the initial state s_1 there are three reachable outcomes, s_4, s_5 and s_6, depending on the chosen policy: a policy which chooses a_2 in the first stage may reach s_4 with probability 0.5, and either s_5 or s_6 with probability 0.5, depending on the action chosen in the second stage. Because the reward of reaching s_5 is much less than s_4 ($R(s_5) \ll R(s_4) < R(s_6)$), any agent that reaches s_3 should choose action a_2 to avoid incurring a large penalty. Unfortunately, by the resource constraint at most half the agents can issue a_2 in state s_3, which means that the optimal policy chooses a_2 on the agents with high id $i \geq \frac{1}{2}n$, and a_1 for the others, in both stages.

Unfortunately, the application of a stochastic preallocation algorithm like Column Generation results in all agents choosing a_2 in state s_1, because the expected consumption of an agent choosing a_2 in every state is

$$\mathrm{P}(s_3 \mid s_1, a_2) \cdot c(s_3, a_2) = \frac{1}{2} \cdot 2 = 1. \tag{4.1}$$

As a result, the expected of consumption of all agents is n, which fits within the constraint in expectation. Of course, once the agents transition, the number of agents reaching s_3 will exceed $\frac{1}{2}n$ with probability 0.5. In this case, re-planning on the basis of the realized state will force some of the agents to change policy, resulting in outcome s_5. As a result,



$L = 1$
$a_1 : \mathrm{P}=1$
$a_2 : \mathrm{P}=\frac{n-1}{n}$
$a_1, a_2 : \mathrm{P}=1$
$a_1 : \mathrm{P}=1$
$a_2 : \mathrm{P}=\frac{1}{n}$
$a_2 : \mathrm{P}=1, c=1$
s_1
s_2
s_3
s_4
s_5
s_6
$R(s_4) = -\epsilon$
$R(s_5) = -2\epsilon$
$R(s_6) = M + \frac{e}{e-1} 2n\epsilon$

Figure 4.2: Single-agent MDP model with horizon 2, 6 states, 2 actions and transition, reward, consumption and limit functions as indicated by the figure, annotated on the transition. Omitted functions have value 0. Variable i is the agent id, n is the number of agents, M is a large positive constant and ϵ a very small positive constant.

the expected value of re-planning in joint state s will be upper bounded by

$$\begin{aligned} m &= \left|\{\cdot \mid s_i = s_3\}\right| \\ V[s] &\leq \mathrm{P}(m > 0.5n) \cdot -M + \sum_{i=1}^{n} Q_i[s_3, a_2] \\ &\leq -0.5M + n \end{aligned} \tag{4.2}$$

Since we are free to choose the value of M, and the computed policy is independent of its value, we can make the expected value arbitrarily bad by increasing M to ∞.

4.1.2 Worst-case analysis of re-planning deterministic preallocations

Suppose we are instead given a problem where all agents are modeled by the MDP presented in figure 4.2. This model has similar dynamics as the model considered in the previous section, in that there are the same three reachable outcomes, under the same potential policies. However in this case the reward of reaching s_5 is only slightly less than s_4, while s_6 is highly valuable ($R(s_5) < R(s_4) \ll R(s_6)$). On the other hand, the constraints ensure at most one agent can transition from s_3 to s_6, and in expectation only one agent reaches s_3 if all of them try. Therefore, the optimal policy chooses a_2 on all agents in s_1, and awards the resource to one of the agents to reach s_3, if any of them do.

For this model, a deterministic resource allocation can only award the resource to one agent. The other agents will choose a_1 in the initial state, as this obtains a (fractionally) higher expected value than aiming for s_5. As such, at most one agent will reach s_3, with probability $\frac{1}{n}$. Because the resource utilization depends on the probability of at least one agent reaching s_3, re-planning the allocation after the state transition to s_2 and s_3 has occurred cannot increase the utilization. As such, (re-planning) a deterministic preallocation on this model obtains value

$$\begin{aligned}
V[s] &= \mathrm{P}(s_3 \mid s_1, a_2) \cdot Q(s_3, a_2) - \sum_{i=1}^{n-1} \epsilon \\
&= \frac{1}{n}(M + \frac{e}{e-1} 2n\epsilon) - \sum_{i=1}^{n-1} \epsilon \\
&= \frac{M}{n} + \frac{2e}{e-1}\epsilon - (n-1)\epsilon \\
&= \frac{M}{n} - n\epsilon + \frac{3e-1}{e-1}\epsilon
\end{aligned} \tag{4.3}$$

Because we are free to choose the number of agents n, we can make the value of the preallocation solution arbitrarily small. What remains to show is that the optimal value is independent of n in the limit, because when all agents choose a_2,

$$\begin{aligned}
m &= \Big|\{\cdot \mid s_i = s_3\}\Big| \\
V^*[s] &= \mathrm{P}(m > 0) \cdot Q(s_3, a_2) - \sum_{i=1}^{m-1} 2\epsilon - \sum_{i=1}^{n-m} \epsilon \\
&\geq \mathrm{P}(m > 0) \cdot Q(s_3, a_2) - 2n\epsilon \\
&= (1 - \mathrm{P}(m = 0)) \cdot Q(s_3, a_2) - 2n\epsilon \\
&= \Big(1 - \Big(\frac{n-1}{n}\Big)^n\Big) \cdot Q(s_3, a_2) - 2n\epsilon \\
&= \Big(1 - \frac{1}{e}\Big) Q(s_3, a_2) - 2n\epsilon \\
&= \frac{e-1}{e}(M + \frac{e}{e-1} 2n\epsilon) - 2n\epsilon \\
&= \frac{e-1}{e} M + 2n\epsilon - 2n\epsilon \\
&= \frac{e-1}{e} M
\end{aligned} \tag{4.4}$$

is independent of n. As a result, re-planning a deterministic preallocation can also result in solutions which are arbitrarily far from optimal.

4.2 Decoupling constraints by dynamic arbitrage

In a resource preallocation the agent subproblems are decoupled through the per-agent permissions to use resources. Unfortunately, the previous section has shown that such static allocations may lead to arbitrarily poor solutions. Therefore, in this section we investigate a decoupling based on a dynamical resource allocation scheme, which uses a resource *arbiter* to step in whenever the agents threaten to overuse the available resources.

We show that, from the perspective of a single agent, the presence of a resource arbiter decouples its planning problem from the other agents.

Suppose that we obtain a collection of decoupled, uncoordinated single-agent policies. If we jointly execute these policies, we may expect the agents to arrive at a joint state s, for which their policies jointly recommend an action which uses too much of the available resource capacity. If we can detect that this will be the case before the action is issued, we can revise the chosen action to avoid the constraint violation. We propose to let a resource arbiter use the expected values of the agents' policies to determine a suitable recourse action.

The resource arbiter takes as input the current state s, time t, and the k 'active' constraints $L_{t,k}$. It then computes the feasible action with the highest expected value by considering the expected values of each agent-action pair, and selecting the maximum-valued joint action satisfying the constraints. Formally:

$$
\begin{aligned}
&\textsc{arbitrage}(\pi, t, s) = \\
&\qquad \arg\max_{x_{i,j}} \sum_{i=1}^{n} \sum_{j=1}^{|A_i|} x_{i,j} \cdot Q_{\pi_i}[t, s_i, a_j] \\
&\qquad\quad \text{s.t.} \sum_{i=1}^{n} \sum_{j=1}^{|A_i|} x_{i,j} \cdot c(t, s_i, a_j) \leq L_{k,t} && \forall k \\
&\qquad\quad \sum_{j=1}^{|A_i|} x_{i,j} = 1 && \forall i \\
&\qquad\quad x_{i,j} \in \{0, 1\} && \forall i, j
\end{aligned}
\tag{4.5}
$$

The ARBITRAGE action-selection equation describes a binary (0-1) linear program, which is hard to solve optimally. However, the model has a generalized knapsack structure which has been studied in its own right, and several efficient heuristic solutions have been proposed for this multidimensional multiple-choice knapsack problem (Moser, Jokanovic, and Shiratori, 1997; Akbar et al., 2006). Additionally, if only one constraint is active ($\max k = 1$) the model reduces to the multiple-choice knapsack problem (Sinha and Zoltners, 1979), which admits a linear time $O(n)$ solution to its linear programming relaxation ($0 \leq x_{i,j} \leq 1$; by Zemel, 1984). Finally, if there is only one constraint *and* agent resource consumptions are binary ($\forall i, a : c(t, s_i, a) \in \{0, 1\}$), the optimal solution can be found in $O(n \log n)$ by sorting the agents by the maximum benefit of receiving the resource, and awarding resources to the first $\lfloor L_t \rfloor$ with positive benefit.

Because the value of the solution is itself only a heuristic estimate of the true value obtained by the policy, finding the optimal ARBITRAGE solution is not essential. On the other hand, the response time of the solution is important, because the algorithm is deployed at policy execution time. Therefore, we make use of the linear programming relaxation with a greedy rounding heuristic to efficiently obtain good solutions, unless the model properties allow us to find the optimal solution efficiently.

Thus far, we have assumed that we can compute decoupled single-agent policies which take into account the dynamic resource allocation induced by ARBITRAGE. When policies π_i are computed individually, without regard for the fact that the arbiter may influence the selected action, expected values Q_{π_i} may significantly overestimate the true value. This in turn affects the quality of the resource allocation computed by equation (4.5). Therefore, to make an arbitrage solution effective, agents should plan their policies anticipating the effect of the arbiter. We show in the following sections that agents can compute decoupled policies for dynamic resource allocation by taking into account the probability of being allocated an action (Section 4.3), or by considering the utility cost of using the resource (Section 4.4).

4.3 Planning a best-response to arbitrage

One approach to incorporate dynamic resource allocation in the planning routine of an agent is to incorporate the consequence of arbitrage into the model of agent. From the perspective of an agent i, in a policy which chooses action a_j, the arbiter changes the effect of the action to that of alternative action a_k with some probability $\mathrm{P}^{\mathrm{ARBI}}(a_k \mid \pi, s_i, a_j)$. This probability is derived from the joint states the other agents reach when i is in s_i, and the value of the resources used action by a_j to other agents in each of these states. Given this probability, we can compute an arbitrage-discounted value function V^{ARBI}:

$$\begin{aligned} V^{\mathrm{ARBI}}_{\pi_i}[h+1,\cdot] &= 0, \\ Q^{\mathrm{ARBI}}_{\pi_i}[t,s_i,a_j] &= R_i(t,s_i,a_j) + \sum_{s'_i \in S_i} \mathrm{P}(s'_i \mid s_i,a_j) V^{\mathrm{ARBI}}_{\pi_i}[t+1,s'_i], \\ V^{\mathrm{ARBI}}_{\pi_i}[t,s_i] &= \max_{a_j} \sum_{a'_j \in A_i} \mathrm{P}^{\mathrm{ARBI}}(a'_j \mid \pi,s_i,a_j) Q^{\mathrm{ARBI}}_{\pi_i}[t,s_i,a'_j]. \end{aligned} \tag{4.6}$$

The resulting policy is a best-response to the influence of the arbiter, and in turn of the policies of the other agents.

Of course, determining probability $\mathrm{P}^{\mathrm{ARBI}}$ is challenging: it is defined over all reachable joint states under a fixed set of agent policies, which is exponential in the number of agents, and depends on the policy of all agents being fixed up to time t. Therefore, we make two simplifications: first, we assume that agent i has a limited influence on the resource availability, meaning that changing the policy of only i does not change the probability of the joint states reached by the other agents. Second, we restrict our attention to joint states reached by Monte Carlo simulation.

The pseudo-code of the best-response approach is given in Algorithm 5. The algorithm starts with computing unconstrained policies for all agents on line 1. Then, as long as policies have not reached an equilibrium, we let each agent compute a best-response

Algorithm 5 Best-response planning for Constrained MMDPs

```
1: $\forall i : \pi_i \leftarrow$ PLAN($i$)
2: while NOTCONVERGED($\pi$) do
3:     for $i = 1$ to $n$ do
4:         $P(s) \leftarrow$ SAMPLEJOINTTRAJECTORY($\pi$)
5:         $\pi_i \leftarrow$ PLANBESTRESPONSE($i, \pi, P(s)$)
6:     end for
7: end while
8: return $\pi$
```

to all the others in turn on line 5, planning for the probabilities P^{ARBI} induced over π and the sampled $P(s)$. When the policies converge, the agents are in an equilibrium where none of the agents can improve their utility by changing its policy single-handedly.

4.4 Planning with marginal utility costs

The best-response algorithm described in the previous section accurately considers the effect of arbitrage on the policy of the agent under consideration; however, its convergence can be slow: the probability $P^{ARBI}(a_k \mid \pi, s_i, a_j)$ is only defined for states s_i that will be reached under the joint policies, and therefore does not generalize to unobserved states. Therefore, for states which are not reached, best-response may determine a strongly overestimated utility. As a result, agents may oscillate between disjoint policies in subsequent iterations. In order to generalize the effect of the arbiter over all states, we exploit the dual costs produced by solving the relaxed version of the ARBITRAGE LP in (4.5).

Given a current joint state s and set of policies π, we can solve the relaxed arbitrage LP to obtain a (fractional) action assignment to agents which satisfies the constraints, as well as the dual costs $\lambda_{t,k,s}$ that a potential action would need to exceed to be chosen in state s. In other words, the arbitrage will only select an agents' preferred action when its utility exceeds $\lambda_{t,k,s}$. Of course, an agents state s_i can occur in multiple joint states, so instead we propose to compute the average dual costs in all joint states,

$$\lambda_{t,k} = \sum_{s} P(s) \cdot \lambda_{t,k,s}. \tag{4.7}$$

We use Monte Carlo sampling as before to avoid the exponential complexity of determining $P(s)$ for all s.

Given these arbitrage-based dual costs, we can compute best-response policies using the cost-based Bellman equation, in the same way it was used in Column Generation. The difference in this case is that the $\lambda_{t,k}$ costs are based on actual consumption in states s instead of expected consumption in Column Generation. It is easy to imagine that the

Algorithm 6 Fictitious Play for Constrained MMDPs

```
 1: $\pi \leftarrow \langle \pi_1, \pi_2, \dots, \pi_n \rangle$
 2: $\lambda_{t,k} \leftarrow 0$                                                    $\forall t$
 3: $l \leftarrow 1$
 4: $\mathbb{P} \leftarrow \emptyset$
 5: while NOTCONVERGED($\lambda_{t,k}$) do
 6:     $V_{\pi_i}[h+1, s] \leftarrow 0$                                     $\forall i, \forall s \in S_i$
 7:     for $t = h$ down to 1 do
 8:         $\lambda_{t,k,s} \leftarrow$ ARBITRAGE($\pi, t, s$)                          $\forall s \in \mathrm{P}^l_t \in \mathbb{P}$
 9:         $\lambda_{t,k} = \sum_{\mathrm{P}^l_t \in \mathbb{P}} \sum_{s \in \mathrm{P}^l_t} \frac{\mathrm{P}^l_t(\mathbf{s})}{|\mathbb{P}|} \cdot \lambda_{t,k,s}$
10:         $V_{\pi_i}[t, s] \leftarrow \max_{a_j} \big(Q_{\pi_i}[t, s_i, a_j] - \lambda_t c(t, s_i, a_j)\big)$       $\forall i, s \in S_i$
11:     end for
12:     $\mathrm{P}^l \leftarrow$ SAMPLEJOINTTRAJECTORY($\pi$)
13:     $\mathbb{P} \leftarrow \mathbb{P} \cup \mathrm{P}^l$
14:     $l \leftarrow l + 1$
15: end while
16: return $\pi$
```

cost function may oscillate between extremes if we only consider the $\mathrm{P}(s)$ resulting from the previous policies. Therefore, to ensure convergence, we keep the history of all past samples, inspired by fictitious play (Berger, 2007). Each prior can be seen as the adversary 'nature' performing her actions as a consequence of our choices. By remembering all past plays, eventually the full strategy of nature is obtained. Thus, let P^l be the probability distribution over states in iteration (or play) l, then we maintain the set $\mathbb{P} = \langle \mathrm{P}^1, \mathrm{P}^2, \dots, \mathrm{P}^l \rangle$, and compute the expected cost as

$$\lambda_{t,k} = \sum_{\mathrm{P}^l_t \in \mathbb{P}} \sum_{s \in \mathrm{P}^l_t} \frac{\mathrm{P}^l_t(\mathbf{s})}{|\mathbb{P}|} \cdot \lambda_{t,k,s}. \tag{4.8}$$

Algorithm 6 presents the pseudo-code bringing all steps together. The algorithm makes use of three subroutines. Subroutine NOTCONVERGED tests if expected cost $\lambda_{t,k}$ changed more than ϵ, relative to the previous iteration. We also pass a maximum number of iterations, which allows to cut off computation before convergence to limit maximum run-time. The subroutine ARBITRAGE(π, t, s) implements the relaxed ARBITRAGE LP (4.5) to find the marginal utility cost in joint state s visited in at least one prior sample. Finally, subroutine SAMPLEJOINTTRAJECTORY(π) performs a number of Monte Carlo samples of new trajectories for current policy π. The number of samples to perform is a parameter controlling a runtime vs. accuracy trade-off.

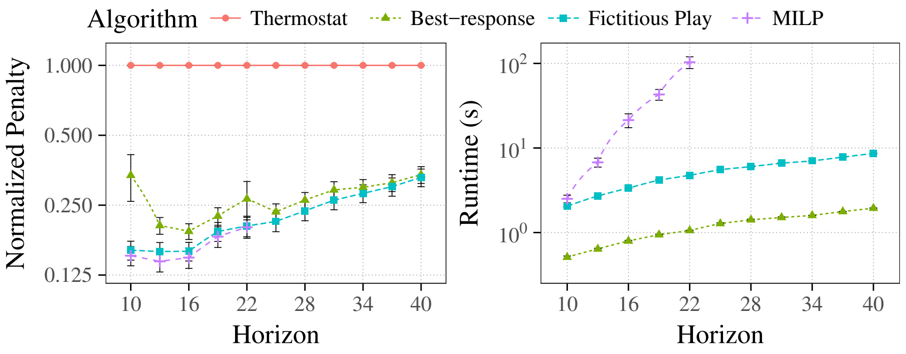


Figure 4.3: Performance of the algorithms on randomly generated TCL instances of increasing horizon. The left plot shows the quality of the policy, normalized to the thermostat policy. The right plot shows the wall-clock runtime in seconds. Both plots are on a log-y scale. Reported values are means and standard errors over 50 instances per setting of agents. Lower values are better.

4.5 Empirical Evaluation

To evaluate the performance of the proposed arbitrage-based algorithms, we compare them on the TCL domain described in Section 3.4.2. This problem is particularly suitable because it proved to be the most difficult to solve by deterministic preallocation algorithms such as the preallocation MILP, while at the same time its single binary constraint per time step allows us to solve ARBITRAGE exactly.

In the experiments we measure the (wall-clock) runtime to compute a policy, and its quality. Policy quality is measured by performing 50,000 Monte Carlo simulations of the model, subject to the final policy computed by the algorithm. The Fictitious Play and Best-response algorithms are set to perform at most 10 iterations, computing 1000 priors after each iteration. The MILP formulation is optimized in Gurobi version 6.5. All algorithms are subjected to a 30-minute time limit. Algorithms that are unable to solve an instance within this limit are excluded from solving larger instances.

We compare our arbitrage-based algorithms with the preallocation MILP, which like the arbitrage solutions also guarantees safe policies. Figure 4.3 presents the mean and standard error of both policy quality and required run-time to compute the final policy. Previously we showed that the MILP preallocation algorithm faces exponential complexity in the number of agents. Here we observe that as expected by the increase in integer variables, the same holds for increasing the length of the horizon. On the other hand, our arbitrage solutions are able to find high-quality safe policies in polynomial time. Between the two solutions, we observe that Fictitious Play has a higher run-time complexity than Best-response, because it computes its arbitrage influence over a larger history of reachable states in its set of plays $\mathbb{P}$. On the other hand, by generalizing over resource costs Fictitious Play is able to converge to a better solution in only a limited number of iterations compared to Best-response, which results in solutions with a lower penalty. Compared to the preallocation MILP, the solutions computed by Fictitious Play are not significantly different in quality, making our approach highly effective for its complexity.

4.6 Related Work

Decoupling many weakly coupled agents was studied by Meuleau et al. (1998) in the context of global and instantaneous resource constraints. Their instantaneous resource constraints are equivalent to our constraints, but their approach to tackle them is much less sophisticated. They propose to only enforce the instantaneous constraints during the on-line policy execution phase, which is equivalent to using the initial policy of the fictitious play algorithm. This is likely to result in overly optimistic policies, which we expect to cause significantly more resource allocation conflicts compared to the solutions computed by the algorithms proposed in this chapter.

Influence-based abstraction proposed by Witwicki and Durfee (2010) and extended by Oliehoek, Witwicki, and Kaelbling (2012) focuses on finding optimal algorithms to decouple and solve loosely coupled planning problems. Interactions between pairs of agents are decoupled by summarizing their interaction in a mutually modeled variable. This only leads to a reduction of the search space when some agents do not interact at all, which is not the case for our all-to-all resource constraints. Therefore influence-based abstraction cannot be used to reduce the complexity of finding optimal policies for resource constrained problems.

4.7 Conclusions

In this chapter we investigated algorithms to allocate resources dynamically. Static preallocation algorithms such as those considered in the previous chapter find solutions which cannot adapt to realized state trajectories of the stochastic transitions. We show in section 4.1 that this is the case even when we are able to re-plan the preallocation in each time step. Therefore we propose in section 4.2 to employ dynamic resource allocation on-line, through use of a centralized resource arbiter.

The use of an arbiter decouples the agents' control problem, but during planning they must still consider how the behavior of the other agents influences their chances at receiving resources from the arbiter. We propose two alternative approaches for agents to plan policies while taking into account the presence of the arbiter: the first approach shown in section 4.3 explicitly models the chance that an agent will receive the permission to use its chosen action, taking into account the probability of alternative actions being performed. By planning agents one at a time, they are able to compute a best response to each other's policies. The second approach presented in section 4.4 instead exploits the fact that arbitrage solutions also result in dual costs, which the agents can include in their planning problems to estimate whether their resource-using actions provide sufficient utility to be awarded by the arbiter.

Because both proposed algorithms sample actual constrained trajectories, the resulting policies are aware of the consequences of failing to obtain resources, making them robust against the examples on which naive replanning approaches fail. We empirically evaluate the arbitrage-based algorithms with the optimal static preallocation MILP on a more realistic problem in section 4.5, showing that these algorithms find high-quality dynamic allocations efficiently; the resulting policies are not only safe with respect to the constraint, but they are also efficiently computable.

Chapter 5

Constrained Multi-agent Learning

The previous chapters explore different coordination algorithms, but always under the assumption that the models of the agents, meaning their transition and reward functions, are known exactly. This assumption is quite strong considering the uncertainties present in the real world. In the recurring example of controlling thermostat devices, we likely do not know the thermal response of every building initially; to address this challenge requires the use of a learning agent (Ruelens et al., 2015). However, in this setting we only need to perform the learning once, as part of the initialization of the device. Learning unknown models of agents is especially relevant in environments where new agents continually appear, such as recommender systems. A recommender system assists users by narrowing down huge collections to a shortlist of items which are most likely to be of interest (Resnick and Varian, 1997). Because recommendations are targeted to the preference of an individual, their effect on a collective of users can unintentionally overload infrastructural capacity. For example, the use of an uncoordinated route guidance system can adversely affect the average waiting times in theme parks (Cheng et al., 2013). However, capacity constraints on recommended items may also serve an operational purpose: in virtual items like news articles, limiting recommendations for naturally popular items can promote recommendation diversity. Therefore, in this chapter we investigate how to coordinate agents when we relax the assumption that agents' models are known, in order to compute policies for a large-scale personalized recommendation domain.

Before knowing an agent's dynamics, a controller is bound to make mistakes while exploring how its model responds. Nevertheless, in recommending items to users, we should minimize these errors in order to keep them engaged. Ideally, we would make *optimal* learning decisions at each point in the trajectory (Duff, 2002). The optimal learning problem can be cast as solving a continuous state Partially Observable MDP (POMDP). Unfortunately, the complexity of solving continuous-state POMDPs prevents us from finding optimal policies for anything but toy problems. However, in reality we often do not have to learn from a blank slate. If prior data of users' interactions is available, we may be able to cluster their behaviors into categorical descriptions. This simplifies the learning problem to identifying the true model of a hidden-model MDP (Chadès et al., 2012, hmMDP), which removes the continuous-state aspect of the problem.

Work by Guez, Silver, and Dayan (2013) suggests two approaches to arrive at an optimal learning policy: (i) On-line sparse sampling algorithms such as Posterior Sampling Reinforcement Learning (PSRL; Strens, 2000), which uses an optimistic heuristic to eventually converge to the optimal policy, or (ii) Off-line planning of an optimal learning policy, by following Chadès et al. (2012) in casting the parametric MDP to a stationary Mixed-Observable MDP (MOMDP; Ong et al., 2010). Unfortunately, neither approach can be applied directly to our capacity-aware recommendation problem; to the best of our knowledge no version of PSRL exists which incorporates constraints in the learning process, and it is not clear under what conditions the multi-agent case converges to a policy satisfying the constraints. On the other hand, computing an optimal policy for a MOMDP is a PSPACE-complete problem (Papadimitriou and Tsitsiklis, 1987), limiting its practical scalability.

Therefore, in this chapter we propose two novel algorithms: the first algorithm is an extension of PSRL to the multi-agent, constrained setting. The second algorithm exploits the structural properties of the problem to approximately solve the constrained MOMDP itself, by bounding the belief space expansion to states where the regret of switching to the best type's MDP policy is low. We show how both algorithms can be used as subroutine in the Column Generation stochastic preallocation algorithm studied in Chapter 3.

In order to demonstrate the broader applicability and scalability of our proposed algorithms, we evaluate them on a large-scale personalized recommendation domain. We develop a recommendation model of tourists visiting the city of Melbourne on the basis of a real dataset. Compared to SARSOP (Kurniawati, Hsu, and Lee, 2008), a state-of-the-art approximate algorithm for MOMDPs, both our approaches scale to significantly larger models. Our experiments additionally show that both approaches can produce higher quality solutions on the recommendation domain.

The chapter is organized as follows: first Section 5.1 introduces additional background information, presenting the parametric MDP model that underlies the constrained learning problem considered in this chapter. The following Section 5.2 defines the model of the constrained learning problem formally, showing how the multi-type parametric MDP maps to this formalism. Section 5.3 presents the adaptation of PSRL to this multi-agent, constrained setting. In Section 5.4 the construction of our bounded regret algorithm is explained. Then, Section 5.5 makes the planning problem considered in this chapter concrete by showing how the tourist recommendation problem can be represented by the model. Section 5.6 presents the results obtained on this recommendation domain. Section 5.7 discusses work which is closely related to the problem and methods discussed in this chapter. Finally, Section 5.8 concludes the chapter and presents discussions on the open challenges.

5.1 Background

This section presents additional background material specific to this chapter. In particular, we present an overview of models and formalisms which have been proposed to control systems with hidden or imperfect information. Such problems broadly fall under the field of reinforcement learning (Section 5.1.1), in which a controller is tasked with learning to control the system through interactions with the environment. Learning hidden information *optimally* (Section 5.1.2) requires the system to reason over information states, which describe the beliefs of the controller over the behavior of the environment. In the general case, this amounts to solving a continuous-state partially observable Markov decision process over all possible transition matrices, which is highly intractable. Fortunately, we frequently have prior knowledge available to us on the types of models which are plausible. When the environment behaves as one of a finite number of potential models, the result is a hidden model Markov decision process (Chadès et al., 2012), which can itself be learned optimally by solving a mixed-observability Markov decision process (Section 5.1.3).

5.1.1 Bayesian reinforcement learning

Broadly seen, reinforcement learning asks how an agent should (learn to) act, when its feedback primarily comes from interactions with the environment (Sutton and Barto, 2018). Reinforcement learning problems are usually also modeled as Markov decision processes, but unlike the planning problems we considered previously, in learning problems we do not assume that the decision maker has access to the transition or reward functions. Instead, the agent must discover these functions through interactions with the environment, asking the agent to solve the exploration/exploitation trade-off: when should it attempt to learn more of the environment, and when should it ‘cash in’ on its knowledge by choosing the best-known actions?

Unfortunately, general reinforcement learning tasks have a high sample complexity, requiring long interaction periods before a good policy is obtained (Kakade, 2003). Although several algorithms have been proposed which can attain near-optimal solutions in a number of samples polynomial in the number of states and actions (Brafman and Tennenholtz, 2002; Kearns and Singh, 2002), such algorithms are not yet practically usable. In many cases we can do better by informing the learning algorithm with prior knowledge. For example, the algorithm may be informed that the environment behaves according to a specific instantiation of a general parametric model such as a Parametric Markov decision process.

Parametric Markov decision processes

In many situations where an automated control policy must be learned, we have a general model of the behavior available to us based on the knowledge required to engineer the system in the first place. For example, turning the steering wheel of a car has the effect of rotating the entire car in the same direction. However, the degree to which this happens depends on structural factors such as the steering geometry and weight of the vehicle. In this example, we can interpret the different car models as different instantiations of the same abstract behavior. A Parametric Markov Decision Process captures this intuitive idea by letting the transition and reward functions of the model depend on parameters.

Recall that a basic Markov Decision Process (MDP, Section 2.1) is defined by a tuple $\langle S, A, R, T, h\rangle$, having reward function R and transition function T. Parametric MDPs define one or both of these functions to additionally be dependent on structural parameters (Dearden, Friedman, and Andre, 1999). Let Θ stand for a parameter space, with θ representing a specific parameter setting. Then a parametric MDP has tuple $\langle \Theta, S, A, \bar{R}, \bar{T}, h\rangle$ with

$$\begin{aligned} \bar{R} &: \Theta \times S \times A \to \mathbb{R}, \\ \bar{T} &: \Theta \times S \times A \times S \to [0,1]. \end{aligned} \tag{5.1}$$

Fixing parameter θ will *instantiate* a parametrized MDP$_\theta$, having $\langle S, A, R_\theta, T_\theta, h\rangle$ with functions

$$\begin{aligned} R_\theta(s,a) &= \bar{R}(\theta, s, a), \\ T_\theta(s,a,s') &= \bar{T}(\theta, s, a, s'). \end{aligned} \tag{5.2}$$

Given a parametric MDP, we would like to construct a controller which works for *any* instantiation, without revealing to the controller the parameters $\hat{\theta}$ used to instantiate the model. For example, if the parameters represent human behavior it may be impossible for users to self-report this information. Similarly, it would be unreasonable to measure the specific insulation level of each house where a smart thermostat is to be deployed. Solving this problem amounts to solving a reinforcement learning problem with prior information, and for this setting the posterior sampling algorithm is known to work well.

Posterior sampling reinforcement learning

In order to identify the true dynamics of a parametric MDP, the posterior sampling algorithm (PSRL; Strens, 2000) iteratively refines a probability density over the parameters of the MDP. It does so through application of Bayes' Theorem on the likelihood of the

Algorithm 7 Posterior sampling reinforcement learning.

```
    Given prior $\phi = \mathrm{P}(\theta_j)$, epoch length $\tau$, initial state $s_1$
    Set time $t \leftarrow 1$, state $s \leftarrow s_1$, belief $b \leftarrow \phi$
 1: for episode $k = 1 \rightarrow \left\lceil \frac{h}{\tau} \right\rceil$ do
 2:     sample $\theta_j \sim b$
 3:     plan $\pi_j$ for mdp$_{\theta_j}$
 4:     for timestep $l = 1 \rightarrow \tau$ and $t \leq h$ do
 5:         select action $a = \pi_j(t, s)$
 6:         observe next state $s' \sim \mathrm{P}(\cdot \mid \hat{\theta}, s, a)$        ▷ Sampling true environment $\hat{\theta}$
 7:         update $b$ by Bayes' rule, computing $\mathrm{P}(b' \mid s, a, s', b)$
 8:         $s \leftarrow s'$, $b \leftarrow b'$, $t \leftarrow t + 1$
 9:     end for
10: end for
```

observed state given the prior probability over typed transition functions. The algorithm applies the Thompson sampling heuristic (1933) in selecting actions, by optimistically following the optimal policy computed for a hypothesized MDP, sampled from the prior probability over types. Although the PSRL algorithm is straightforward to state and based on an optimistic heuristic, it has strong performance guarantees. The algorithm has sample complexity polynomial in the number of parameters when learning the model of factored MDPs (Osband and Van Roy, 2014), as well as the guarantee of finding the optimal policy in a logarithmic number of time steps with high probability in the continuous (non-episodic) setting (Gopalan and Mannor, 2015).

Algorithm 7 presents the steps of PSRL formally. The algorithm proceeds in episodes of length τ, during which the algorithm optimistically assumes that the user behaves according to model θ_j sampled from the current belief over models b. The assumed type for the current episode is sampled on line 2, and an optimal control policy for the assumed type is subsequently computed on line 3. Then, policy π_j is used to select actions for τ steps, during which the belief over the user's true type is updated with every observed transition on line 7.

5.1.2 Optimal learning

Although the PSRL algorithm eventually converges to the optimal policy, its use of a heuristic raises the question of optimality of the trajectory leading up to convergence. For example, if there exists an action which is not part of the optimal policy for any mdp$_\theta$, this action will never be chosen by PSRL. This is the case even if taking this action immediately reveals the true parameters of the MDP. In order to reason about such information gathering actions, an algorithm should explicitly reason about the decision-theoretic value of information (Howard, 1966). Bayes-adaptive Markov decision processes explicitly include learned information in their state, allowing an agent to make optimal learning decisions by computing the expected value of information (Martin, 1967; Duff, 2002).

Bayes-adaptive Markov decision processes

Suppose that we are given a Markov decision process with an unknown transition function. For this general case Duff (2002, section 1.3) shows that optimal learning is possible, when we include additional information in the state of the MDP. This additional information takes the form of a matrix of Dirichlet distribution parameters, having one parameter for each $\langle s, a, s' \rangle$-pair. The Dirichlet distribution follows because a transition function is a multinomial distribution, for which the Dirichlet distribution is the conjugate prior. The Dirichlet distribution can be seen to encode a probability density function over the space of multinomial distributions, which are the candidate transition functions of one of the transitions of the underlying MDP.

Formally, the uncertainty in a single transition $T(s, a)$ of the MDP is given by a vector of parameters β of an $|S|$-dimensional Dirichlet distribution. By combining these parameter vectors in a matrix M^a recording all possible s to s' transitions of a single action a, we can keep track of the uncertainty over the consequences of this action. A Bayes-adaptive Markov decision process (BAMDP) explicitly keeps track of the uncertainty in all actions, by augmenting the states of the original MDP with these matrices of Dirichlet parameters, resulting in generalized states $\langle s, M \rangle$. The available actions are the same as in the original MDP, and the reward function of the BAMDP is simply the original reward function of the base MDP applied to the unboxed base state.

The transitions over these generalized states can be derived from the Bayes update of observing s' as a result of transition $T(s, a)$: for a Dirichlet distribution this corresponds to increasing the value of parameter $\beta_{s'}$ by 1. This means that for an example 2-state, 2-action MDP with a uniform prior over the transition matrix, a transition $s_1 \xrightarrow{a_1} s_2$ results in the augmented transition:

$$\left\langle s_1, \overbrace{\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}}^{M^1} \overbrace{\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}}^{M^2} \right\rangle \xrightarrow{a_1} \left\langle s_2, \begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \right\rangle. \tag{5.3}$$

The probability of such a generalized transition to occur can be derived from the state itself. By the definition of the Dirichlet distribution, the mean probability of outcome i is given by $\frac{\beta_i}{\sum_k \beta_k}$. For the transition probabilities of state s_i and action a_j we extract the parameters from row i of matrix M^j. Therefore, in our example the prior probability of the transition is $\frac{1}{2}$, while the posterior probability of the same transition as a consequence of the update to M becomes $\frac{2}{3}$. As the update to M is deterministic, these probabilities also apply to the generalized transition.

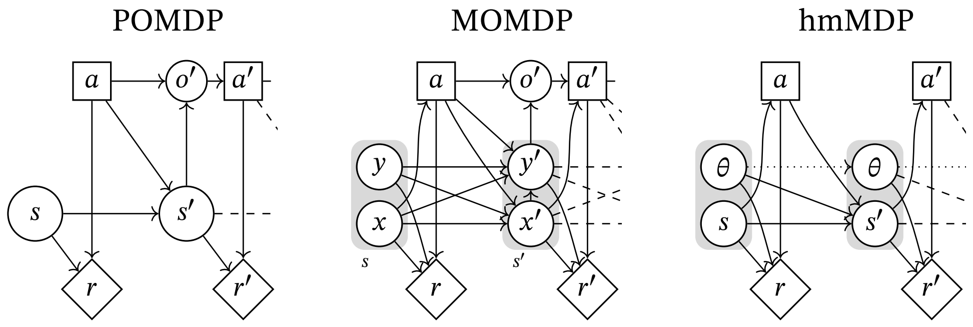


Figure 5.1: Comparison of the models of partial observability through their dynamic Bayesian networks (Boutilier, Dean, and Hanks, 1999). Nodes represent states and observations (circles), decisions (squares) and rewards (diamonds), while edges encode direct stochastic influences.

Thus, by augmenting the states of the original unknown-transition MDP with information about the transition uncertainty, we again obtain an MDP. The BAMDP can be solved using regular MDP algorithms to determine a policy which optimally trades off exploration with exploitation of knowledge. Unfortunately, the size of the augmented state space is exponential in the number of time steps, which means that this direct approach is not feasible to use in practice. However, the BAMDP model can be represented by the generalization of Markov decision processes to the setting where the state is not directly observable (Duff, 2002), allowing the application of approximation algorithms developed for such partially observable MDPs.

Partially observable Markov decision processes

Compared to fully observable MDPs, a Partially Observable MDP (POMDP) adds a finite set of observations $o \in O$ as well as an observation function $\Omega = \mathrm{P}\left(o \mid a, s'\right)$, resulting in tuple $\langle S, A, O, T, R, \Omega, h\rangle$ (Kaelbling, Littman, and Cassandra, 1998). A decision maker for a POMDP is not informed of the state directly, but must instead condition its actions on the received observations. The observation function correlates the observations to the state transition, giving the probability of observing o based on the action and the (unobserved) state. This means that the controller must choose actions based only on these (current and past) observations. Figure 5.1 (left) presents the interactions between transition, observation and reward functions graphically. Computing an optimal policy for a finite-horizon POMDP is a PSPACE-complete problem (Papadimitriou and Tsitsiklis, 1987), making it a harder problem than the fully observable planning problems considered previously (unless NP = PSPACE).

To plan for POMDP models it is often convenient to reason about belief states (Kaelbling, Littman, and Cassandra, 1998). A belief state b records a probability distribution over the possible states S, with $b(s)$ indicating how likely the agent expects to be in state s. Given a belief state b, the action taken a, and the observation received o, the subsequent belief state $b'(s)$ can be derived using application of Bayes' theorem. For a finite-horizon POMDP planning problem, the number of reachable belief states B is also finite, as (in the worst case) they form a tree of depth h with a branching factor of $|A||O|$ at each node. This belief-state tree can be used as the state space of a belief-state MDP that is equivalent to the POMDP, which can in principle be solved by an application of the Bellman equation (2.2), although the tractability of this approach is limited by the exponential growth of B in the horizon h.

Duff (2002, section 5.3) shows that the optimal learning problem can be encoded as a POMDP with *continuous* state space. In particular, the states take the factored form $\langle s, P\rangle$, where s is the discrete state of the original MDP and P the (stationary) matrix of continuous transition probabilities. The transition function takes the probabilities from the state matrix, $T(\langle s', P\rangle \mid \langle s, P\rangle, a) = P_{s,a,s'}$. As in the BAMDP case, the reward function comes from the original MDP reward function applied to the unboxed state, $R_{\text{POMDP}}(\langle s, P\rangle, a) = R_{\text{MDP}}(s, a)$. Finally, the observation function simply reports the MDP state, $\Omega(a, \langle s', P\rangle) = s'$. Although studied less frequently than the discrete state version, several approximation algorithms to solve continuous state POMDPs exist (Porta et al., 2006; Bai et al., 2011). Nevertheless, learning in such a general setting where no structural knowledge is assumed can take many time steps to converge to an accurate model of the MDP. When structural information is available, both the time to learn and the time to compute an optimal learning policy can be reduced.

5.1.3 Hidden model Markov decision processes

Suppose that we are given structural information about the underlying MDP we are trying to learn, such as in the case of Parametric MDPs. A natural assumption in this case is that there exist only a finite number of parameter settings (e.g. car models or housing categories). To our knowledge, Silver (1963, chapter 2) is the first to investigate how to make decisions when the true transition matrix of such a 'multi-matrix' MDP must be identified, while keeping the reward function fixed. Chadès et al. (2012) extend the scope of the problem to the setting where the reward function is also uncertain, resulting in a hidden model Markov Decision Process.

A hidden model MDP (hmMDP; Chadès et al., 2012) is a Parametric MDP where the parameter space is restricted to be a finite set, resulting in a number of possible candidate MDPs with type $\theta \in \Theta$, each with their own transition and reward functions. The decision maker for a hmMDP does not observe the true type $\hat{\theta}$, but must instead learn this parameter from the observed transitions. Figure 5.1 (right) presents the interactions of elements of a hmMDP graphically. Although it appears to be a much simpler model than POMDPs, Chadès et al. prove that computing an optimal policy for hmMDPs also falls in the PSPACE complexity class. In order to compute an optimal policy for hmMDPs while leveraging existing solvers, it is useful to cast the problem to a Mixed-observability Markov decision process.

Mixed-observability Markov decision processes

Models with mixed observability have states which can be factored into a partially observable and a fully observable factor (Hauskrecht and Fraser, 2000; Ong et al., 2010). Therefore, a Mixed-Observability MDP (MOMDP) consists of a set of observable state factors $x \in X$ and a set of unobservable state factors $y \in Y$, each with their own transition functions, $T_X(x' \mid x, y, a)$ and $T_Y(y' \mid x, y, a, x')$. As in the partially observable case, an observation function $\Omega(o \mid a, y')$ exists to inform the decision maker about transitions of the hidden factor. However in addition to the observations, the decision maker also conditions his actions on the observable factor x. The resulting tuple of a MOMDP is thus $\langle X, Y, A, O, T_X, T_Y, R, \Omega, h\rangle$. Figure 5.1 (middle) shows the interactions of the functions graphically.

Any POMDP model can be turned into an equivalent MOMDP by adding a single dummy state $X = \{x\}$, letting $Y = S$. Therefore, the models are equally expressive, which means that they share the PSPACE-complete complexity of planning a policy. On the other hand, turning a MOMDP model into a POMDP model requires increasing the size of the state and observation space by setting $S = X \times Y$ and $O = O \times X$, and modifying the transition and observation functions appropriately. This means that for models which exhibit mixed-observability, MOMDP solvers are able to use the factored state space to reduce the dimensionality of the value function, thereby reducing the solve time by orders of magnitude relative to POMDP solvers (Araya-López et al., 2010).

Optimal learning in hidden model MDPs

Chadès et al. (2012) show that hmMDPs can be encoded as an equivalent MOMDP model. By using this transformation, state-of-the-art approximate POMDP solver SARSOP (Kurniawati, Hsu, and Lee, 2008) can be employed to compute high-quality policies, indirectly solving the optimal learning problem for hmMDPs. The transformation is as follows. Given a Parametric MDP $\langle \Theta, S, A, \bar{R}, \bar{T}, h\rangle$ with a finite set of parameters Θ, we derive a MOMDP $\langle X, Y, A, O, T_X, T_Y, R, \Omega, h\rangle$ having elements

$$\begin{aligned} X &= S, & T_X(s' \mid s, \theta, a) &= T_\theta(s' \mid s, a), \\ Y &= \Theta, & R(s, \theta, a) &= R_\theta(s, a), \\ O &= \{o_{\text{NULL}}\}, & \Omega(o_{\text{NULL}} \mid a, \theta') &= 1, \end{aligned} \tag{5.4}$$

$$T_Y(\theta' \mid s, \theta, a, s') = \begin{cases} 1 & \text{if } \theta = \theta', \\ 0 & \text{otherwise.} \end{cases} \tag{5.5}$$

Intuitively, this transformation sets the observable factor transition function to the state transition function of the instantiated MDP_θ corresponding to the hidden factor θ. The same holds for the reward function. Because the true parameters are hidden to the decision maker, only a dummy observation is provided after each transition. Finally, the hidden factor transition function T_Y is stationary because the parameters of the MDP remain fixed, making the resulting model a Stationary MOMDP (Martin et al., 2017).

The stationary property admits the use of the expected value of MDP policies as a lower bound on the MOMDP value function. Martin et al. (2017) propose to compute the optimal MDP policy for each type, and subsequently apply the optimal policy of each type to all other types. The resulting values become an alpha-vector per policy, which together constitute a lower bound on the true value function. Initializing solvers such as SARSOP with this lower bound speeds up the convergence of the algorithm, by providing tighter bounds for pruning vectors.

5.2 A constrained multi-agent learning problem

The learning algorithms discussed in the previous section investigate how an agent should learn when it is otherwise unconstrained. However the environment of an agent frequently imposes some limitations on its behavior. Especially when *multiple* learning agents interact, sharing available capacity is challenging because the exploitation/exploration trade-off couples across the agents: should an uncertain agent be awarded a scarce resource in order to learn, or should it be used by another agent to obtain reward with high certainty? In this chapter we investigate how a collection of agents should learn when they are constrained by resource limits restricting their joint actions.

We consider multi-agent systems consisting of n functionally similar agents, which we represent by a single Parametric MDP description, with each agent i behaving according to the MDP instantiated from its type θ_i. Like in the hmMDP case, we assume agent types to be sampled from a finite set of potential types, according to a known prior probability $\phi = \mathrm{P}(\theta_i = \theta)$. The controller for each agent must learn what the type of the agent is, while ensuring that the agents jointly satisfy the global constraints. As in the previous chapters, constraints are modeled as a vector of capacities $\vec{L}$, with the consumption of an agent's action given by a consumption function $c(s,a) = \vec{u}$. The result is a constrained multi-agent hmMDP problem having tuple $\langle n, \phi, \Theta, S, A, \bar{R}, \bar{T}, h, c, \vec{L} \rangle$. Figure 5.2 presents the interactions between two agents' individual functions and the two global constraints.

Limitations of state of the art

To solve such constrained multi-agent hmMDP models, existing work suggests two options: (i) Apply PSRL per agent to eventually converge to the optimal policy, or

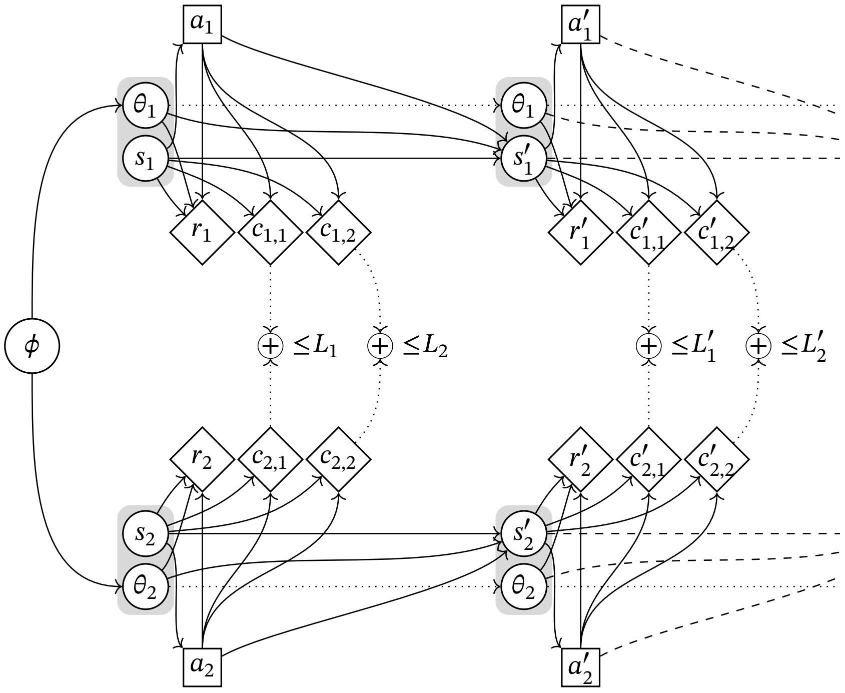


Figure 5.2: Dynamic Bayesian network of the constrained multi-agent learning problem having two agents and two constraints. Solid lines indicate stochastic influence, dashed lines project stochastic influences to the next stage, while dotted lines indicate deterministic transitions.

(ii) Solve the constrained Stationary MOMDP to obtain an optimal learning policy for each agent. Unfortunately, both approaches cannot be directly applied to this problem; for the case of PSRL, to the best of our knowledge no version exists which can incorporate constraints in the learning process. It is also not clear under what conditions the multi-agent case will converge to the optimal policy satisfying the constraints.

For adding constraints to a Stationary MOMDP, we can employ methods from the Constrained POMDP literature. The Column Generation algorithm which we studied and found effective in Chapter 3 was originally proposed for Constrained POMDPs (Yost and Washburn, 2000). This approach could be combined with SARSOP for solving the sub-problems, following the results of Martin et al. (2017) on its effectiveness for Stationary MOMDPs. Nevertheless, this approach suffers several drawbacks for our problem. SARSOP is an infinite-horizon solver, which means that the solver employs discounting to keep the required lookahead bounded. This incurs approximation errors on non-stationary problems, even if we annotate the state space with an additional time factor (thereby increasing its size by a factor h). Additionally, the approximate nature of the solver means that the expected value of the policy computed by the solver may not correspond with the true expectation obtained by executing the policy, which is problematic because we need true expectations for the integration with Column Generation. Finally, the scalability of SARSOP is still relatively limited compared to the size of problems we consider.

Contributions

In order to overcome these limitations, we propose two new algorithms for solving constrained multi-agent hmMDPs, resulting in the following contributions:

(i) We show that we can combine Column Generation with PSRL to obtain an efficient heuristic learning algorithm under constraints (Section 5.3).

(ii) We exploit the stationary structure of the MOMDP in computing an approximate continuation for any belief point, based on the minimal regret MDP policy. As we can compute exact expected values for this continuation, we are able to compute exact expectations for policies computed on a truncated belief space. The resulting approximate solutions can thus be embedded in Column Generation directly to perform nearly optimal learning for constrained multi-agent hmMDPs (Section 5.4).

(iii) We use the expected minimal regret to propose an efficient belief space truncating condition, which results in a highly scalable approximation algorithm for Stationary MOMDPs (Section 5.4.2).

Algorithm 8 Multi-agent constrained PSRL.

```
    Given prior $\phi = \mathrm{P}(\theta_j)$, epoch length $\tau$, initial state $s_1$
    Set time $t \leftarrow 1$. For all $i$, set state $s_i \leftarrow s_1$, belief $b_i \leftarrow \phi$
 1: plan $\langle x, Z \rangle = \textsc{colGen}(\mathrm{MDP}_{\theta_j}, n, \phi)$        ▷ Alg 1 using LP (5.6)
 2: for episode $k = 1 \to \lceil \frac{h}{\tau} \rceil$ do
 3:     sample $\forall i:\ \theta_i \sim b_i$
 4:     sample joint $\vec{\pi}$ by $\pi_i \sim \langle x_{\theta_i}, Z_{\theta_i} \rangle$
 5:     for timestep $l = 1 \to \tau$ and $t \leq h$ do
 6:         select joint action $\vec{a} = \vec{\pi}(t, \vec{s})$
 7:         observe next state $\forall i:\ s'_i \sim \mathrm{P}(\cdot \mid \hat{\theta}_i, s_i, a_i)$        ▷ Sampling true type $\hat{\theta}_i$
 8:         update $b_i$ by Bayes' rule, computing $\forall i:\ \mathrm{P}(b'_i \mid s_i, a_i, s'_i, b_i)$
 9:         $\vec{s} \leftarrow \vec{s}\,', \vec{b} \leftarrow \vec{b}', t \leftarrow t + 1$
10:     end for
11: end for
```

5.3 Multi-agent constrained PSRL

Previously, we have seen that Column Generation is an effective algorithm for constrained multi-agent MDPs (Chapter 3). At the same time, we know that PSRL is an effective heuristic algorithm to learn the true type of a parametric MDP. Therefore, we propose to combine these two algorithms to obtain an effective heuristic for constrained learning problems. Because the Thompson sampling heuristic samples hypothesized MDPs from the parametric description which are eventually correct, we may compute policies for these converged MDPs using Column Generation to obtain a joint policy which eventually satisfies the constraints. While belief has not converged, the expected consumption of an agent's policy may not be attained because its true type does not match the sampled type. Nevertheless, we expect this strategy to work well in practice because every correctly identified agent behaves according to its constraint-respecting policy, and eventually all agents converge to their type.

Algorithm 8 presents the proposed approach. Already on line 1 we apply column generation, to compute the optimal mix of resource-satisfying policies over the *expected* number of agents of each type. Recall that in the multi-agent Column Generation Linear Program (LP; Eq. (2.17)), we assume agents to have heterogeneous models, necessitating planning an MDP policy per agent in each iteration of the algorithm. Because here our agents behave according to homogeneous types, we can apply the simplification of Yost and Washburn (2000): objects with the same model can be added together. In this case we can limit ourselves to planning a policy per type in each iteration, which may give a significant speed-up when $|\Theta| \ll n$. The resulting LP becomes

$$\begin{aligned}
\max_{x_{i,j}} \quad & \sum_{i=1}^{|\Theta|} \sum_{\pi_j \in Z_i} x_{i,j} \cdot \mathrm{E}[V_{\theta_i,\pi_j}(s_1)], & \\
\text{s.t.} \quad & \sum_{i=1}^{|\Theta|} \sum_{\pi_j \in Z_i} x_{i,j} \cdot \mathrm{E}[c_{\theta_i,\pi_j,r}(t,s_1)] \leq L_{r,t} & \forall r, \forall t, \\
& \sum_{\pi_j \in Z_i} x_{i,j} = n \cdot \mathrm{P}(\theta_i), & \forall i, \\
& x_{i,j} \geq 0, & \forall i, \forall j.
\end{aligned} \tag{5.6}$$

The relative frequencies $x_{i,j}$ computed by column generation define a probability distribution over policies: for a policy $\pi_{i,j}$ in set Z_i, $\mathrm{P}(\pi_{i,j}) = \frac{x_{i,j}}{n \cdot \mathrm{P}(\theta_i)}$. The policy the agent will use is sampled according to this probability distribution on line 4, choosing Z_i according to the agents' hypothetical MDP type sampled on line 3. The remaining structure of the algorithm follows from PSRL directly, accounting for the multiple agents in each step.

At the start and while converging there may be overconsumption due to incorrectly hypothesized agent types. However, as the number of agents of true type $\hat{\theta}_i$ is in expectation $n \cdot \mathrm{P}(\hat{\theta}_i)$, provided the prior ϕ is accurate, the sampled set of agents eventually converges to the distribution used to compute the constraint-satisfying policies. If prior ϕ is inaccurate or the number of agents n is too small to rely on the expectation, column generation can instead be invoked on the sampled types, after line 3. While this may seem to interleave a potentially expensive centralized planning step with the on-line execution of the policy, in practice we can execute warm restarts of column generation by initializing the LP of episode k with the policies computed for episode $k-1$.

5.4 Bounded-regret belief space algorithm

In PSRL, the action a_i to execute is selected on the optimistic assumption that the sampled type θ_j is (close to) the true type of the agent, $\hat{\theta}_i$. Especially early on, before belief b_i has converged sufficiently, this ignores that there may be other actions which are more informative with respect to the belief update. This exploration-exploitation trade-off can be addressed optimally by computing an optimal policy for the Stationary MOMDP model, as shown in Section 5.1.3. Solving a general MOMDP model to optimality is a hard problem. However our models are built out of a combination of MDPs, which enables exploiting this structure during solving. We propose a novel algorithm for these problems, which obtains a bounded approximation error by switching from belief-space MOMDP policy to a regular MDP policy at belief points where the regret of such a switch is low. Because we intend to use this algorithm in Column Generation, we need to take special care that the expected value and consumption computed by this algorithm remain correct for these approximate solutions, which we address in the following section.

5.4.1 Computing exact expectations for a reduced belief space

Recall from Section 5.1.2 that we can construct an equivalent belief-state MDP from a given POMDP, by exhaustively enumerating reachable beliefs b into a state space B, spanning (in the worst case) a tree of depth h with a branching factor of $|A||O|$ at each belief. In principle, this belief-state model of a POMDP can be used to compute an optimal policy, but the exponential size of B prohibits doing so tractably. Therefore, approximation algorithms generally attempt to reduce the size of B, focusing on a subset of the space B'.

Because the belief state space B' is an approximation of the exact state space B, we expect to obtain potentially suboptimal policies. Nevertheless, we require exact expectations of the consumption to use in the Column Generation program, as the satisfaction of the constraints depends on the selected policies using the resources to the reported levels. This can be achieved if we know the exact expected consumption of the policy at each 'missing' belief point *not* in B'. We propose to use the stationary structure of the model to compute an approximate continuation at any belief point.

The belief points $\langle t, s, b \rangle$ of our MOMDP are factored into a time t, MDP state s, and belief b over possible types θ. For states at the corners of the belief where $b(\theta_i) = 1$ (and $b(\theta_j) = 0$ for $i \neq j$), the stationary condition ensures that the optimal continuation is the optimal MDP policy computed for the model instantiated with parameter θ_i. Thus, the expected value of such corner-point immediately follows; if π_i^* is the optimal policy for MDP$_{\theta_i}$, then $V^*[\langle t, s, b \rangle] = V_{\pi_i^*}^{\theta_i}[t, s]$. We propose to approximate missing belief points using the same principle, by selecting the best policy from the optimal policies of each type. Intuitively this follows from the idea that for points which are very close to a corner, choosing policy π_i^* will almost always be correct. In the rare case this choice is incorrect, policy π_i^* is instead applied to another MDP$_{\theta_j}$, resulting in value $V_{\pi_i^*}^{\theta_j}[t, s]$. The probability that this value occurs is $b(\theta_j)$. Thus, the total value of choosing policy π_i^* in belief point $\langle t, s, b \rangle$ is

$$Q\big[\langle t, s, b \rangle, \pi_i^*\big] = \sum_{j=1}^{|\Theta|} \Big(b(\theta_j) \cdot V_{\pi_i^*}^{\theta_j}[t, s] \Big). \tag{5.7}$$

The optimal value of sticking to a given fixed MDP policy in point $\langle t, s, b \rangle$ is then

$$\bar{V}\big[\langle t, s, b \rangle\big] = \max_{\pi} Q\big[\langle t, s, b \rangle, \pi\big]. \tag{5.8}$$

While the expected value $\bar{V}\big[\langle t, s, b \rangle\big]$ is an approximation of the optimal MOMDP expected value $V^*\big[\langle t, s, b \rangle\big]$, it remains a correct expectation because it is based on the belief state b and the exact MDP expectations. Therefore we can use the value of $\bar{V}$ as approximation for any belief point $\langle t, s, b\rangle \notin B'$.

In principle we could compute $\bar{V}\big[\langle t, s, b\rangle\big]$ exactly, by deriving the optimal policy for the resulting belief-weighted Bellman equation,

$$\begin{aligned}\bar{V}\big[\langle t,s,b\rangle\big] &= \max_{\pi} Q\big[\langle t,s,b\rangle,\pi\big] = \max_{\pi}\sum_{j=1}^{|\Theta|}\Big(b(\theta_j)\cdot V_{\pi}^{\theta_j}[t,s]\Big)\\ &= \max_{\pi}\sum_{j=1}^{|\Theta|}\Big(b(\theta_j)\cdot Q^{\theta_j}[t,s,\pi(t,s)]\Big) = \max_{a\in A}\sum_{j=1}^{|\Theta|}\Big(b(\theta_j)\cdot Q^{\theta_j}[t,s,a]\Big) \qquad (5.9)\\ &= \max_{a\in A}\sum_{j=1}^{|\Theta|}\Big(b(\theta_j)\cdot\Big(R_{\theta_j}(s,a)+\sum_{s'\in S}T_{\theta_j}(s,a,s')\bar{V}\big[\langle t+1,s',b\rangle\big]\Big)\Big),\end{aligned}$$

which is exactly the Bellman equation of the stationary MOMDP *without* the belief update. However, this would come down to computing an MDP policy for every belief point not in B' that is reachable from the points in B'. We can avoid this computational burden by the following observation: for points which are very close to corner i, policy π_i^* will be the optimal policy with high probability. If we take care to construct B' such that the reachable points are close to corners, we can limit our search to the optimal policies of each type,

$$\bar{\bar{V}}\big[\langle t,s,b\rangle\big] = \max_{\theta_i\in\Theta} Q\big[\langle t,s,b\rangle,\pi_i^*\big]. \qquad (5.10)$$

As the number of types is fixed, this comes down to computing $|\Theta|$ MDP policies initially, and determining for each of these policies the expected values of applying it to the other types.

Algorithm 9 lists the exact expectation belief space planner. It starts by computing the optimal MDP policy π_j^* for each type θ_j on line 1, followed by determining the exact expected values V_{θ_i,π_j^*} of these policies on every type θ_i on line 2. The remainder of the algorithm computes expected values at each of the generated belief points backwards over time, according to the typical dynamic programming algorithm, except in case a value is needed for a missing belief point on line 13. In case of a missing point b', the best policy π_j^* is selected on line 14, and the expected value of using this policy is computed according to the belief state.

The resulting policy returned on line 26 consists of two stages. For every belief point b in the collection B', the maximally valued action stored in $\pi[b]$ on line 21 is selected. However, in case a $b' \notin B'$ is reached during execution, the policy π_j^* stored on line 15 is used as replacement for $\pi[b']$. Because the expected value of the MDP policies is exact, and b' describes the state distribution that is reached in expectation (Kaelbling, Littman, and Cassandra, 1998), the expected value at any such 'missing' belief state is also exact. Therefore, $V[b_0]$ is the true expectation of the (potentially suboptimal) value obtained by executing the policy computed by Algorithm 9. Therefore, this algorithm avoids all three

Algorithm 9 Bounded belief state space planning.

```
    Given parametric MDP $\langle \Theta, S, A, \bar{R}, \bar{T}, h \rangle$ and approximate belief space $B'$
 1: Plan $\pi^*_j$ for all $j$
 2: Compute $V_{\theta_i,\pi^*_j}$ for all $i$, $j$
 3: Create policy $\pi[b]$
 4: for time $t = h \rightarrow 1$ do
 5:     for belief point $b \in B'(t)$ do
 6:         $V[b] = -\infty$
 7:         for action $a \in A$ do
 8:             $Q[b,a] = R(b,a)$
 9:             for observed next state $s' \in S$ do
10:                 $b' = \textsc{updateBelief}(b,a,s')$
11:                 if $b' \in B'$ then
12:                     $Q[b,a] = Q[b,a] + \mathrm{P}(s' \mid b,a) \cdot V[b']$
13:                 else
14:                     $j = \arg\max_j Q[b', \pi^*_j]$
15:                     $\pi[b'] = \pi^*_j$
16:                     $Q[b,a] = Q[b,a] + \mathrm{P}(s' \mid b,a) \cdot \bar{\bar{V}}[b']$
17:                 end if
18:             end for
19:             if $Q[b,a] > V[b]$ then
20:                 $V[b] = Q[b,a]$
21:                 $\pi[b] = a$
22:             end if
23:         end for
24:     end for
25: end for
26: return $\langle \pi, V[b] \rangle$
```

weaknesses of existing approximate MOMDP solvers: it is a finite-horizon solver without discounting, it computes exact expectations, and it remains tractable by operating on a reduced belief state space by using the properties of our models.

5.4.2 Using expected regret to bound the belief state space

To determine an approximate belief space B' for Algorithm 9, we use the *expected regret* of switching to a fixed MDP policy as criterion for pruning a belief point. As we have seen, at the corners of the belief space, the optimal policy is the MDP policy computed for model instantiated on θ_i, at which point there is no regret. While we could develop the belief state space until a corner is reached, the size of the result typically still remain intractably large. Further reduction of the belief state space can be obtained by switching over to the MDP policy earlier, before the belief has completely converged. At this point, we incur regret proportional to the probability that we are in fact applying the policy for θ_i to the model of θ_j. If it turns out we apply π_i^* to MDP_{θ_j}, we obtain the expected value $V_{\pi_i^*}^{\theta_j}$, for which by definition of optimality $V_{\pi_i^*}^{\theta_j} \leq V_{\pi_j^*}^{\theta_j}$. Thus, the use of policy π_i^* incurs a regret of

$$\textsc{regret}(\langle t, s, b\rangle, i) = \sum_{j=1}^{|\Theta|} \Big(b(\theta_j) \cdot \Big(V_{\pi_j^*}^{\theta_j}[t, s] - V_{\pi_i^*}^{\theta_j}[t, s] \Big) \Big). \tag{5.11}$$

At a given belief point $\langle t, s, b\rangle$, the optimal MDP policy for type i found in (5.10) minimizes this regret, therefore

$$\textsc{regret}(\langle t, s, b\rangle) = \min_i \big(\textsc{regret}(\langle t, s, b\rangle, i) \big). \tag{5.12}$$

Because the MDP policies are computed over the entire horizon, regret is also defined for the prior $b_0 = \langle 1, s_1, \phi\rangle$. The value of $\textsc{regret}(b_0)$ gives an upper bound with which we can compare the regret at any subsequent belief state.

Only pruning belief points with a low absolute regret may not be sufficient to significantly reduce the size of B' in domains which exhibit low-probability observations returning to the initial belief. As motivation, consider the canonical Tiger problem proposed by Kaelbling, Littman, and Cassandra (1998). In this problem, a decision maker is faced with two doors: one hiding a reward, the other a large penalty in the form of releasing a tiger. The actions available to the agent are to open the left door, or the right door, or to listen for the tiger. Listening gives an imperfect observation on its location, either hearing the tiger on the left, or on the right. If, after a period of listen actions the decision maker has received equally many observations left and right, no information has been gained by the agent. While this means that the regret of such a sequence would be equal to the root regret, this situation is highly unlikely to occur. As such, acting optimally in this situation would be inconsequential for the overall expected value of the policy. Therefore, we may limit the growth of B' by also omitting belief points which are exceedingly unlikely to be reached. Let $\mathrm{P}(b)$ stand for the probability of belief point b, then we generate all subsequent belief points from b_0 meeting a threshold parametrized by minimum probability p and shape α:

$$\textsc{regret}(b) > \big(e^{-\alpha(\mathrm{P}(b)-p)} - e^{-\alpha(1-p)} \big) \cdot \textsc{regret}(b_0). \tag{5.13}$$

Threshold (5.13) is based on an exponential decay function over probability $\mathrm{P}(b)$ which attains 0 at $\mathrm{P}(b) = 1$ and approximately $\textsc{regret}(b_0)$ at $\mathrm{P}(b) = p$. Figure 5.3 shows how this threshold condition applies to an instance, highlighting the importance of a smooth trade-off between regret and likelihood of a belief: applying simple conditions results in belief spaces with significant numbers of low regret or low likelihood points.

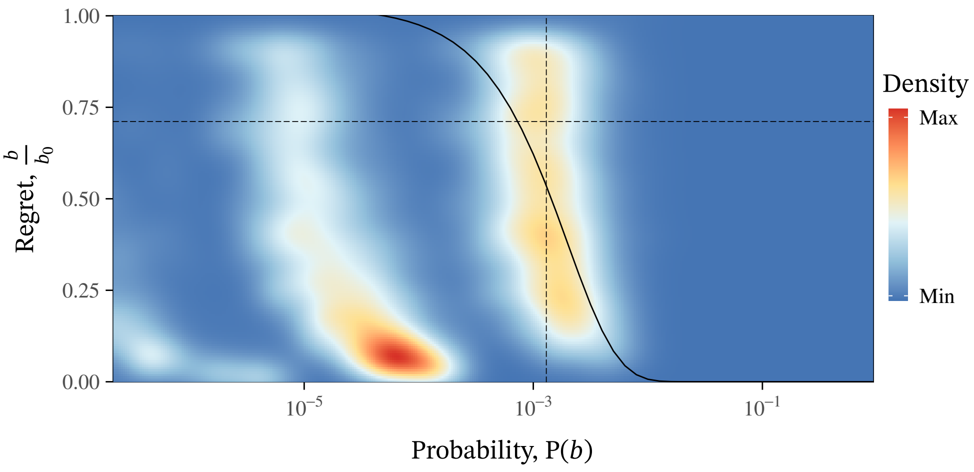


Figure 5.3: Plot comparing the threshold condition (solid curve, $\alpha = 500$, $p = 5 \cdot 10^{-5}$) with the simple thresholds that result in equal-sized belief state spaces (dashed straight lines) on a small instance (log x-axis). Curves overlaid on a 2-D kernel density estimate over the complete belief state space B. Belief points above and to the right of each line are placed in B'.

As these points have a small impact on the expected value, we expect that for the same computation time, policies computed for threshold (5.13) will have significantly higher expected value.

5.5 Capacity-aware sequential recommendations domain

We evaluate the algorithms proposed in the previous sections on a tourist recommendation problem modeled on data of visitors to Melbourne. First we motivate the importance of taking a sequential view to such a recommendation problem, followed by a description of how we fit user models to the dataset.

5.5.1 Motivating capacity-aware sequential recommenders

Personalized recommendations are an increasingly important approach to engage users and to help to filter collections of objects which are otherwise too large to explore (Bell, Koren, and Volinsky, 2007). In many cases, recommendations should also take into account relations between objects and the history of the user, which imposes a sequence relation over the objects. For example, when recommending news articles to readers, the user's history informs his familiarity with a topic and thereby the value of a contextual article over a latest update. Also in recommending points-of-interest to tourists, we need to consider recommendations as a sequence, in order to exploit locality and avoid asking the user to backtrack its path. In both cases, there is also a clear *capacity limit*: servers hosting news articles may be overwhelmed by a sudden increase in traffic from recommended users, and popular points-of-interest can easily become overcrowded. On the other hand, recommendations also provide the potential to steer users around constrained points, motivating the need for capacity-aware sequential recommendations.

One of the primary challenges that a recommender system faces is the discovery of a user's preferences. Existing recommender systems are typically modeled as bandit models or click models. Such models aim to minimize regret incurred from taking exploratory actions (Steck, 2013). Unfortunately, these models cannot capture *contextual history* in actions taken over several time steps (Shani, Heckerman, and Brafman, 2005). This stands in contrast to our Markov Decision Process models, which allow us to encode history in the state of the user.

Because recommendations are intended to influence an individual user's behavior, their effect on a collective of users can unintentionally overload infrastructural capacity. For example, Cheng et al. (2013) demonstrate that the use of an uncoordinated route recommendation system can adversely affect the average waiting times in theme parks. To avoid this negative effect, capacity respecting recommendations should be computed for all agents simultaneously, while respecting their individual preferences.

5.5.2 Modeling points-of-interest recommendations

We evaluate the algorithms proposed in the previous sections on a tourist recommendation problem modeled on data of visitors to Melbourne, derived from a dataset[1] of photograph meta-data from tourists visiting the city (Thomee et al., 2016). Given a finite set of locations l to be viewed one at a time, we model a system recommending a user the next item to view. Although each user has its own goals in visiting, we assume that visitors' interests can be clustered into a set of discrete user types $\theta \in \Theta$. Each type θ defines a valuation over the items, awarding value according to a reward function $R_\theta(l)$ for seeing item l. We first cluster the historic visitor data into types θ based on the types of points photographed, setting the value $R_\theta(l)$ of visiting a point l by the relative frequency with which l is visited by visitors in cluster θ.

From the perspective of a recommender system, the user's interactions result in a history of user actions. At one point, a user may have first seen item l_i, followed by l_j, resulting in a history $\langle \ldots, l_i, l_j \rangle$. Such a history may be summarized in a higher level 'context state' s_k. Given a current context, we assume that the next item user of type θ will visit can be modeled by a probability distribution over the items $\mathrm{P}_\theta\ (l \mid s_k)$. In order to obtain P_θ from the dataset, we fit a Probabilistic Suffix Tree (PST) to each cluster of users. A PST predicts the probability of observing the next symbol in a sequence,

[1] Original dataset publicly available on `https://github.com/arongdari/flickr-photo`

conditional on a variable-length, bounded history of previously observed symbols (Ron, Singer, and Tishby, 1996). Such a PST defines a Markov Chain over the set of possible history states S, which is finite by the maximum depth of the PST. We write $s_{i,j}$ for a history-state recording the sequence $\langle l_i, l_j \rangle$, specifying a user which is now at l_j after first visiting l_i. State s_0 represents the initial empty history $\langle\rangle$. Then, after fitting a PST of depth 2, we construct a closed Markov chain T_θ:

$$\begin{aligned} T_\theta(s_i \mid s_0) &= \mathrm{PST}_\theta(l_i \mid \langle\rangle) && \forall l_i \in P, \\ T_\theta(s_{i,j} \mid s_i) &= \mathrm{PST}_\theta(l_j \mid \langle l_i \rangle) && \forall l_j \in P, \\ T_\theta(s_{j,k} \mid s_{i,j}) &= \mathrm{PST}_\theta(l_k \mid \langle l_i, l_j \rangle) && \forall l_k \in P. \end{aligned} \tag{5.14}$$

In order to control the total size of the state space, we have two options: (i) we can select the number of locations to consider, by limiting to the top-x most frequently visited points in the dataset, and (ii) we can limit the depth of the PST, thereby reducing the number of history states induced over the x locations.

The Markov chain defined by (5.14) is transformed into a Markov Decision Process by including recommendation actions. An important challenge in designing a recommender system is that it is typically not known how agents will change their behavior when receiving a recommendation, because no such recommendation system is in place yet to observe the effect of recommendations on users. We follow Theocharous, Vlassis, and Wen (2017) in assuming that users boost their probability of viewing recommended item l_i in accordance to a (type-specific) propensity to listen $\mu(\theta)$.

We consider two models of sequential recommendation systems: a 'take-it-or-leave-it' model which issues at most a single recommendation at a time, and an 'alternatives' model in which the system can issue at most two recommendations. In both cases, the set of potential recommendation actions A contains a 'no recommendation' action a_0, which behaves as the original Markov chain, and a recommendation action a_i for each item l_i. The 'alternatives' model also contains dual recommendation actions $a_{i,j}$ recommending the visitor to select either item l_i or l_j. In case the user receives a dual recommendation, the user behaves as if it received the recommendation for the more valued of the two, thus

$$\begin{aligned} T_\theta(s' \mid s, a_0) &= T_\theta(s' \mid s) \\ T_\theta(s' \mid s, a_i) &= \begin{cases} T_\theta(s' \mid s, a_0)^{\frac{1}{\mu(\theta)}} & \text{if } l_i \text{ selected in } s' \\ T_\theta(s' \mid s, a_0)/z & \text{otherwise} \end{cases} \\ T_\theta(s' \mid s, a_{i,j}) &= \begin{cases} T_\theta(s' \mid s, a_i) & \text{if } R_\theta(l_i) \geq R_\theta(l_j) \\ T_\theta(s' \mid s, a_j) & \text{otherwise} \end{cases} \end{aligned} \tag{5.15}$$

In this equation z is a normalizing factor to ensure T remains a probability distribution.

The value of a recommendation depends on its quality; good recommendations send the user to locations with a high $R_\theta(l)$ value, while avoiding locations that the user has recently visited. Therefore, we shape the reward of issuing a recommendation by multiplying with a shape function $\sigma(\mathbb{I}(a_i))$, where $\mathbb{I}$ is an index function computing the number of $R_\theta(l_j) > R_\theta(l_i)$. To prevent the system issuing repeat recommendations, we add a penalty term $\rho(s, a)$ when recommendation a is present in the history s. The reward value of a dual recommendation is the average of the two options:

$$\begin{aligned}
\rho(s_h, a_i) &= \begin{cases} \sigma(0)\max_j R_\theta(l_j) & \text{if } i \in h \\ 0 & \text{otherwise} \end{cases} \\
R_\theta(s_{\dots,j}, a_0) &= 0 \\
R_\theta(s_{\dots,j}, a_i) &= \sigma(\mathbb{I}(a_i))R_\theta(l_i) - \rho(s_{\dots,j}, a_i) \\
R_\theta(s_{\dots,j}, a_{i,k}) &= \frac{R_\theta(s_{\dots,j}, a_i) + R_\theta(s_{\dots,j}, a_k)}{2}
\end{aligned} \tag{5.16}$$

Finally, we formalize the constraints by letting $L_{l,t}$ be the maximum number of users allowed to simultaneously view item l at a time. Then, because a user's state reports its current location, we can derive consumption function by letting $C_l(s_{i,j}) = 1$ if state $s_{i,j}$ sees the user currently viewing l.

5.6 Experimental Evaluation

In this section we empirically evaluate our proposed algorithms on the tourist location recommendation problem. Our first experiment evaluates the performance of the proposed belief space bounding condition (5.13). As we expect this Regret condition to result in a high quality subset B' of the full belief space B, we compare it with two typical approaches to produce a bounded-size subset: (i) Sampling, resulting in a depth-first exploration of B through recording belief points along m randomly sampled action-observation trajectories. (ii) Threshold, resulting in a breadth-first exploration of B through recording all belief points which are more likely than a certain probability, $\mathrm{P}(b) \geq p_{\min}$, such as all belief points to the right of the vertical line in Figure 5.3. All three approaches use parameters to control the size of the returned subset, ranging from 1 point up to the *entire* belief state space. As such, we are interested in comparing the *quality* of the returned set as a function of its size, since the size in turn determines the runtime of the Bounded-regret algorithm.

To compare the quality of the sampled subsets, we tune the parameters of each method to return belief spaces which can be processed by Algorithm 9 in a given time, aiming for 15 seconds up to 12 minutes (the largest trees contain about 10 million belief points, which due to memory-to-disk swap effects causes the actual runtime to become slightly longer). Figure 5.4 presents the quality of the policy as a function

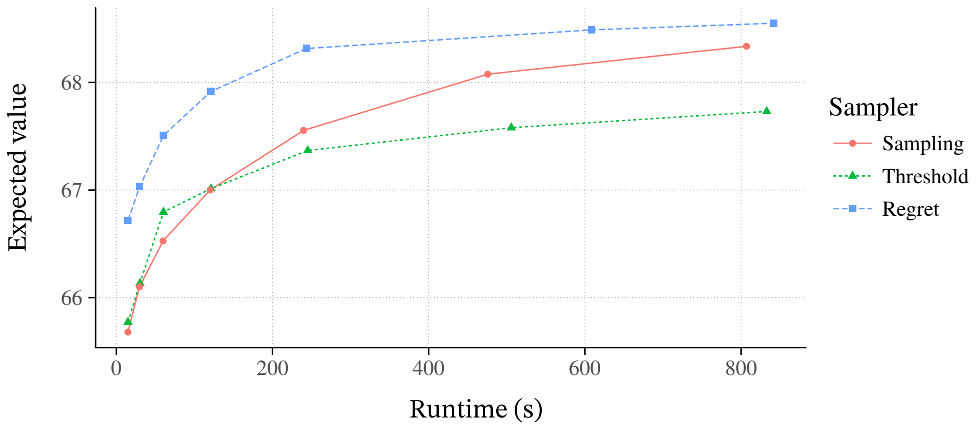


Figure 5.4: Comparison of approaches producing a bounded-size subset B' of the belief state space. Plot shows policy solution quality as a function of the size of B'.

of the plan time, on 5 generated instances of the dual recommendation domain with 7 points of interest, PST depth 1 and 2 types. We observe that the Regret condition finds significantly better quality solutions than the other two approaches for each target time. As a result, it is at least four times faster than the baseline samplers at returning equivalent quality solutions. This is especially important for larger instances, which may be poorly characterized by a limited number of randomly sampled trajectories.

Having shown that the regret condition is effective at finding good-quality bounded state spaces, we now investigate its suitability at solving the *constrained* recommendation problem. We compare the performance of the bounded-regret belief space planning algorithm with SARSOP, a state-of-the-art approximate planner that is capable of exploiting mixed-observability models (Kurniawati, Hsu, and Lee, 2008); for our experiments we used the implementation available on-line.[2] Because SARSOP is an infinite-horizon solver, we take care to explicitly include time in the state space as an observable factor. In addition, we must choose an appropriate value for the discount factor γ. The choice of γ affects the amount of look-ahead that the solver performs, effectively trading off computation time for more myopic behavior. Therefore, we compare two settings: (i) $\gamma = 0.95$, resulting in essentially optimal policies for all solvable horizon lengths (rewards at $t = 9$ valued at 0.63 of rewards at $t = 1$), and (ii) $\gamma = 0.5$, resulting in significantly reduced computation time at the cost of potentially myopic policies. To integrate SARSOP with Column Generation, we must determine the expected value and expected consumption of the policy. We obtain estimates of these expected values through simulation, computing means over 100,000 Monte Carlo samples.

We expect to observe the following trends in comparing the Bounded-regret algorithm

[2] At `http://bigbird.comp.nus.edu.sg/pmwiki/farm/appl/`, APPL Offline, dated 9 June 2014.

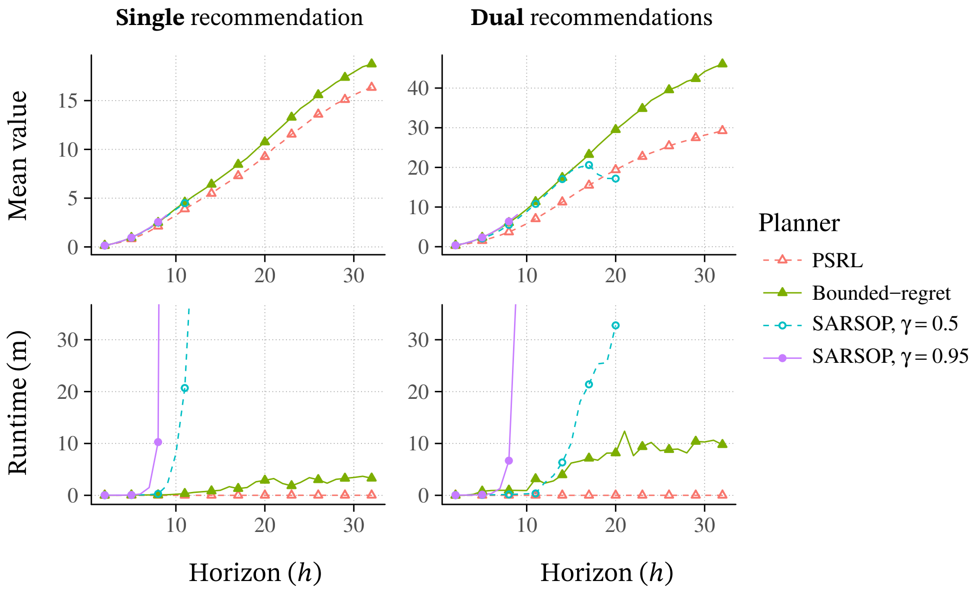


Figure 5.5: Solution quality and planning time of the different sequential recommendation planners, as a function of the horizon.

with SARSOP and PSRL. In the first place, as SARSOP $\gamma = 0.95$ computes *optimal* solutions, we expect it to quickly become intractable as instance size grows. By contrast, SARSOP with $\gamma = 0.5$ should be significantly more tractable while computing equally optimal solutions for short horizon instances, although it may give poor myopic behavior if the horizon becomes too long. Our Bounded-regret solver should remain tractable by comparison for long horizon problems through its bounding of the state space. In addition we expect its solution quality to remain stable because it computes a policy for the entire horizon. Nevertheless, PSRL should be the most tractable algorithm, since it computes MDP policies directly. We expect PSRL to perform (nearly) optimally on problems without significant value of information in the actions. On domains where some actions have information value, we expect the optimal learning approaches to consider exploiting this to converge to the optimal MDP policy significantly faster.

We compare the algorithms on an instance of the tourist recommendation problem consisting of 5 locations, 3 user types, 50 users and PST depth 1. For this experiment we measure the quality of the computed policy as the mean over 1,000 simulations per instance, solving 5 instances per setting of the horizon. The computation time is measured by mean elapsed wall-clock time per instance, with a 30 minute timeout. For the comparison we set the regret bounding parameters to $\alpha = 500$ and $p = 0.005$, which has shown to result in a good trade-off between state-space size and eventual bounding of growth based on preliminary experiments.

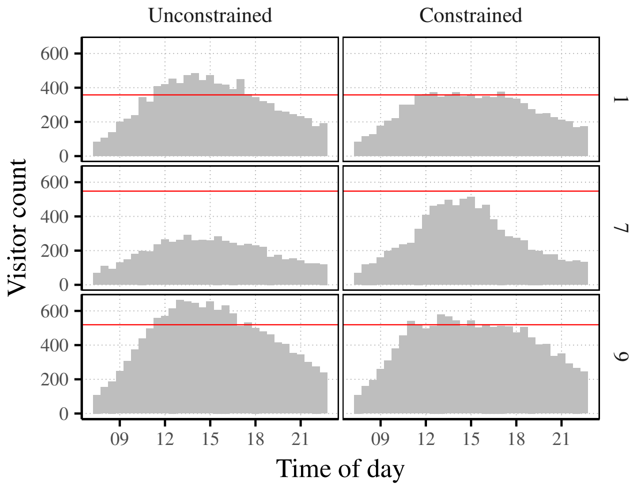


Figure 5.6: Effect of applying constrained recommendations on number of agents visiting locations in the city.

Figure 5.5 presents the results, with the left-hand graphs corresponding to the setting where at most a single recommendation can be issued at a time, while the right-hand graphs are for the domain allowing recommendations with an alternative. The top row presents the observed mean reward, while the bottom row presents the plan time in minutes. We note that we observe all the expected trends in the figure; we highlight three main observations: (i) For these constrained finite-horizon problems, SARSOP quickly becomes intractable, even when the discount factor is set very low. (ii) PSRL indeed returns nearly optimal solutions for the (low information value) single recommendation instances, in a fraction of the time of the other solvers. On the dual recommendation problem it incurs larger regret, but less than the approximate SARSOP solution at $h = 20$. (iii) Bounded-regret finds essentially optimal policies, while at the same time remaining tractable through its effective bounding condition on the state space growth. We note that its runtime stops increasing significantly beyond $h = 20$, as a result of the bounded growth of the state space.

To demonstrate the effect of considering constraints on the crowd dynamics, we perform an experiment on the bounded-regret algorithm as subroutine of the column generation algorithm on a large-scale problem. Figure 5.6 shows a simulation of the number of visitors at three different points of interest, with the red line indicating the constraint level, on a problem with 10 locations, 3 types, PST depth 2 and 5000 visitors during the entire day. The constraint-satisfying policy is able to redirect visitors effectively from crowded points 1 and 9 to 7. While computing this policy required solving over a thousand MOMDPs, by using the Bounded-regret algorithm the capacity-aware recommendation policy was computed within one hour.

5.7 Related Work

In this chapter we explored a constrained, multi-agent optimal learning problem, which in essence amounts to solving a Constrained POMDP. While constrained MDPs are extensively studied, research into its extension to partial observability only started recently. Therefore, in this section we make an effort to survey the developing field in its entirety.

One broad line of work treats constraints on *risks* due to uncertainty, usually called Chance-Constrained POMDPs (CC-POMDPs). Solutions for CC-POMDPs aim to satisfy constraints on, or minimize, the probability of dangerous or expensive states and outcomes. Problem domains where such models have been applied include liver transplantation (Tusch, 2000), the dynamic allocation of radio spectrum while minimizing the risk of collisions (Tehrani, Liu, and Zhao, 2012), as well as planning autonomous planetary exploration (Santana et al., 2016). A state-of-the-art algorithm for finding satisfying solutions for CC-POMDPs is RAO* (Santana, Thiébaux, and Williams, 2016), which performs a forward search over the belief-point state space. It uses an admissible heuristic to find promising actions, while pruning risk-violating actions at each belief point according to risk bounds. Another recent work by Chatterjee et al. (2017) investigates a risk avoiding objective function, in the form of guaranteed payoff optimization. Unfortunately, direct application of chance constraints inherently *couples* multi-agent problems, because chance constraints are not additive in expectation (Ono et al., 2015). This forces algorithms for chance constraints to consider agent states jointly, greatly limiting scalability.

The other broad line of work, to which this chapter also belongs, deals with resource constraints; these models, usually called Constrained POMDPs (CPOMDPs), aim to find the best policy which does not overuse a limited capacity or budget. Besides our constrained personalized recommendation domain, CPOMDPs have also been applied to avionics sensor management optimization (Castañon, 1997), as well as to optimize the deployment of early diagnosis scans for breast cancer, resulting in a significant increase in expected quality-life years over fixed allocations (Cevik et al., 2018). Table 5.1 lists an overview of papers proposing algorithms to solve Constrained POMDPs, annotated with the features that the algorithms exhibit. Algorithms which are specific for Bayesian Reinforcement Learning (RL) have been proposed to solve the constrained optimal learning problem, and as such they can exploit the stationary latent state property to reduce complexity. Algorithms which do not decouple the constraints face an exponential complexity in the number of agents; these works have usually been proposed for single-agent problems (e.g., Undurti and How, 2010) or with sparse agent interactions in mind (Chen et al., 2016). Finally, algorithms such as Column Generation also decouple the agent subproblems, allowing each agent to solve their own policy using modified versions of

		Decouples	
Reference	Bayesian RL?	constraint?	subproblems?
Castañon (1997)	✓	✓	✓
Yost and Washburn (2000)		✓	✓
Isom, Meyn, and Braatz (2008)			
Undurti and How (2010)			
Kim et al. (2011)			
Kim, Kim, and Poupart (2012)	✓		
Wu, Jennings, and Chen (2012)			
Poupart et al. (2015)		✓	
Chen et al. (2016)			✓
Lee et al. (2017)	✓	✓	
Walraven and Spaan (2018)		✓	✓
This chapter	✓	✓	✓

Table 5.1: Survey of algorithms to solve Constrained POMDP models.

single-agent solvers. We show in Chapter 3 that Column Generation outperforms the CMDP LP model because Column Generation decouples the subproblems; this benefit is likely to be significantly more prominent in CPOMDPs because POMDP models are significantly larger than MDPs.

Based on this overview of Constrained POMDP solvers, three works are most closely related to the algorithms presented in this chapter:

(i) Castañon (1997) studies a sensor management problem, where a sensor platform aims to identify the specific features of a known number of objects. The sensor platform has a limited number of available sensors, which must be targeted to obtain sufficient information about each object to classify them. They decouple the sensor allocation problem through a Lagrangian relaxation, resulting in an optimal learning POMDP problem per object. Because their objects have no state features beyond their stationary latent type, they are able to solve the subproblems optimally, which we show is intractable in our domain.

(ii) Lee et al. (2017) investigate a single-agent version of the constrained optimal learning problem. Their algorithm tackles the full Bayesian RL case, without the assumption of a finite number of types, and therefore their algorithm operates on the Bayes-Adaptive MDP model of Duff (2002). As the number of belief-states grows exponentially fast, they propose a belief-state approximation scheme based on merging two belief states when they are ϵ-close to each other. As this approximation is complimentary to our regret-based pruning, we could employ this approach to extend our algorithm to the full BAMDP case as well. Although they also propose to use a belief-state perspective, they handle the constraints differently, by integrating the belief-state MDP into the CMDP LP. Since this LP does not decouple the subproblems, we expect our approach to be significantly more scalable.

(iii) Walraven and Spaan (2018) propose a novel approximate algorithm for constrained POMDPs on the basis of Column Generation and point-based approximate solvers. They solve the inexact expected-value problem by converting α-vector policies to policy graphs. This algorithm is directly applicable to our domain, but because it does not consider the stationarity and mixed-observability inherent in our domain, we expect this approach to be less scalable than our Bounded-regret algorithm.

Beyond the Constrained POMDP subfield, Zhang, Durfee, and Singh (2017) study a multi-agent problem where agents compute policies which are guaranteed to satisfy commitments, despite the fact that agents have uncertainty about their model (corresponding to a hidden-model MDP). The uncertainty in their model also distributes over a finite number of types, however, their constraints are over the achievement of specific states with a minimum probability. While commitments could in principle be used to satisfy resource constraints, their solution framework uses a Mixed-Integer Linear Program having number of binary variables equal to the number of belief states, resulting in an exponential complexity in the number of belief states.

We investigate a Bayesian reinforcement learning approach in this chapter, which enables us to plan an optimal learning policy completely off-line, by reasoning about a probability distribution over possible models of the world as a part of the state space. This is in contrast with the more typical model-free learning approaches, which usually just perform greedy exploration while keeping track of the expected value of state-action pairs (the $Q(s, a)$ values). However, recently Bellemare, Dabney, and Munos (2017) proposed a sophistication of the model-free learning approach, by instead learning probability distributions over the state-action values. These distributions do not just capture the transition uncertainty that is always present in the world due to the transition function, but also the agent's *model uncertainty* over the actual reward and transition functions. Therefore, this method conceptually sits in between the model-based and the model-free perspectives. Because this approach can capture model uncertainty, a promising future work would be to add resource consumption distributions to incorporate constraint handling in model-free reinforcement learning. This would allow for safe reinforcement learning when the rich prior information we use to create the parametric model is not available. However, we show that when such information is available, our approach can be used to learn to act optimally in the minimum number of samples.

5.8 Conclusions and Discussion

In this chapter we studied, developed and evaluated algorithms to solve resource-constrained, multi-agent learning problems. When the exact transition and reward dynamics are unknown, agents must learn to adapt their models on the basis of observations. Bayesian reinforcement learning proposes how an agent should reason optimally about the value of an additional observation. Unfortunately, the complexity of this approach is too high for practical systems, especially when learners are additionally constrained.

Even with the simplifying assumption that the learner only has to identify its own model from a finite set of potential models, solving the resulting constrained MOMDP is not practical. We therefore propose two novel approximate algorithms to compute high-quality solutions for this problem tractably. The first algorithm, multi-agent constrained PSRL, connects Column Generation to the PSRL algorithm, resulting in a highly tractable solution that inherits existing results on logarithmic regret and polynomial time to convergence. The second algorithm, bounded-regret planning, starts from the optimal learning principle of solving the constrained MOMDP; however, it remains tractable by truncating the belief state space, using the finite types assumption to compute the expected regret of truncating the state space at each decision point. Because the regret at each truncated point is computed exactly, the expected values of the entire policy can be computed exactly, making this algorithm especially suitable for integration with Column Generation.

We demonstrate with experiments on a personalized recommendation domain that both new algorithms are capable of computing capacity-aware recommendation policies of high quality. Constrained PSRL is the most tractable of the two algorithms, and it is especially effective in models where all actions are equally informative. Bounded-regret planning computes nearly optimal learning policies in bounded time relative to the length of the horizon. We show that this algorithm is especially effective in settings where some actions provide valuable information about the agents' models.

As a result, this work presents a practical implementation of optimal learning for an important real-world inspired problem. The proposed expected regret condition may also prove useful in other learning algorithms. We expect that it may be possible to incorporate it in on-line algorithms for optimal learning problems, such as Monte Carlo tree search for Bayes-adaptive MDPs (Guez, Silver, and Dayan, 2013).

5.8.1 Discussion

We show that the proposed algorithms are effective for our problems. This indicates that many natural extensions of our algorithms to cover a broader set of domains may be equally successful. We identify and discuss two practical issues which would require the algorithms and ideas in this chapter to be extended in order to address them: (i) the prior over how likely each type is may not be correct, (ii) the reward function of each type may not be completely known.

Handling incorrect priors

For the satisfaction of the constraints, the Column Generation algorithm requires the reported consumption values to be achieved in expectation. The reported consumption values in turn rely on the prior ϕ to be the true probability distribution over types. In practice, the system operator may only be able to provide a rough estimate of ϕ. For example, in our tourist domain the prior may be subject to seasonal influences, with families being much more likely to visit during school holidays. If we know that our prior may not be accurate, the resource allocation needs to be evolved alongside the posterior belief, necessitating re-planning. However, even re-planning only ensures constraint satisfaction once the posterior has converged to the true distribution. A stricter constraint-handling paradigm is required if the learning stage must be safe as well.

We have seen in Chapter 3 that constraint satisfaction can be guaranteed by computing deterministic resource preallocations, at the cost of computational complexity and potentially conservative policies. If individual agent policies are computed subject to a deterministic preallocation, their behavior would be guaranteed to be safe, independently of the correctness of the prior. Unlike for MDP models, as far as we know no deterministic preallocation algorithms have been suggested for POMDPs. The CMDP-based LP model of Poupart et al. (2015) could be modified into the deterministic preallocation MILP of Wu and Durfee (2010), but this would further increase the complexity of the method. Conversely, *given* a resource preallocation, the worst-case reachable belief space of the Bounded-regret algorithm actually decreases, as the use of a preallocation prunes the set of admissible actions. This suggests that Constrained Factored Policy Iteration (Algorithm 3) may also be effective when applied to compute a deterministic resource preallocation for our hidden-model MDPs.

Unknown or sparse reward function

In designing our tourist recommendation domain, we could make use of an extensive dataset of earlier visits to estimate the reward values of all recommendable locations. However, when the number of recommendable items is large, or when new items are added with regularity, the reward function may be unknown or only partially known. In this case, the optimal learning problem is undefined, and model-based approaches cannot be applied. Instead, model-free reinforcement learning paradigms should be used. Incorporating constraints in a model-free reinforcement learning problem results in a safe reinforcement learning problem.

Safe reinforcement learning has attracted significant attention recently (García and Fernández, 2015), as safety is an important consideration for adoption of these methods in practice. Unfortunately, the few existing approaches to tackle model-free safe reinforcement learning cannot be tractably applied to the multi-agent version: Hans et al. (2008) propose an algorithm using a learned safety function and safe return policy to estimate the risk of taking an exploratory action, although the safety function of one agent would then be dependent on the actions taken by the other agents, coupling their policies. Similarly, in the Diverse Exploration algorithm (Cohen, Yu, and Wright, 2018), a policy which was safe in an iteration i may become unsafe in a later iteration due to other agents changing their behavior, breaking the safe improvement assumption. As such, an important future work will be to develop safe multi-agent model-free reinforcement learning algorithms. One idea which may prove useful in such an algorithm is Collaborative Filtering (Schafer et al., 2007), which amounts to using the experienced reward of one agent to infer the expected reward for another agent having similar parameters.

Chapter 6

Conclusions

In almost any setting where individuals cooperate, they need to coordinate their actions around the use of resources. Therefore, an important challenge for Artificial Intelligence is the creation of agents which can autonomously coordinate their actions subject to resource constraints. Coordination on resources is made even more challenging by the presence of uncertainty, because it threatens to reduce the reliability of plans made in advance. Therefore, at the start of this thesis, we asked the question:

> *How can agents optimize their behavior under uncertainty, to maximize their collective utility while jointly respecting the global resource constraints?*

In this chapter we conclude the thesis by examining how the contributions presented within combine to form an answer to this question, as well as discuss the main limitations of this thesis, which provide opportunities for future work.

6.1 Answers to the research questions

Based on an analysis of the strengths and weaknesses of existing work on optimizing multi-agent systems under constraints and uncertainty, we discovered three open challenges in Section 1.2.3. This thesis investigated each of the challenges in turn, through chapters 3, 4, and 5 respectively; here we briefly summarize the key results.

1. How can safe resource preallocations for a decentralized setting be computed efficiently?

Existing work has proposed the use of static resource preallocations to decouple agents, either using deterministic, worst-case allocations, or through stochastic, expected-value based allocations. We advanced this line of work in Chapter 3, by proposing new algorithms that tackle the weaknesses of each of these approaches: (i) we propose CPI, a heuristic algorithm to compute high-quality worst-case allocations efficiently, and (ii) we propose an algorithm to bound the probability of constraint violations when using expected-value based allocations. In addition, we show how preallocations can be used even when the resource constraint itself is uncertain and behaves according to a stochastic prediction. The resulting algorithms are empirically shown to be efficiently computable while resulting in high-quality solutions, outperforming the state of the art in terms of scalability *and* safety.

2. How can we compute (near-)optimal policies exploiting communication between agents to perform dynamic resource allocation?

Preallocations are effective at decoupling agents, and the resulting policies have the advantage that they require no further interaction. However, the downside is that the individual agent policies cannot adapt to joint realizations of uncertainty. Therefore, in Chapter 4 we investigated the possibility of allocating resources to agents dynamically, based on their realized state. We developed an arbiter, responsible for intervening when the collective demand exceeds the capacity, and show that agents can plan policies which take the effect of the arbiter into account, using one of two strategies: (i) a best-response to the decisions of the arbiter, by aggregating the probability that an action is permitted into the transition function, and (ii) an average utility cost of the resource, computed over fictitious plays, simulated trajectories of previously computed policies. Both strategies result in dynamic policies that react dynamically to the state transitions, yet are efficiently computable. While we cannot give guarantees on the distance to optimality, the proposed algorithms can perform arbitrarily better than naive open-loop re-planning approaches based on the preallocation algorithms studied in Chapter 3.

3. How can we compute policies to learn a model of agents' dynamics, when they operate in resource-constrained environments?

Successful application of planning requires that we have an accurate model of the true dynamics of the world, which may not be possible to obtain a priori. For example, when a new recommender system is introduced, there would not be any data available to predict what users will do when they receive a recommendation. Instead, the agent would need to learn its dynamics on-line; we investigated algorithms which allow the agents to learn under global constraints in Chapter 5. We show that by mapping the learning problem to a partially observable planning model, preallocation algorithms can be used to perform *optimal* (in terms of the value of information) learning under constraints. As planning under partial observability has PSPACE complexity, we proposed bounded belief state space planning, an approximation algorithm based on truncating the state space in places where the agent dynamics are (almost) completely known. This algorithm is shown to outperform state-of-the-art planning algorithms for general partially observable planning problems in scalability, while still computing essentially optimal learning policies.

Conclusions on the main research goal

In conclusion, we have proposed novel algorithms addressing each of the three challenges identified in response to the central research question. Taken together, these algorithms show how agents can coordinate their actions when their dynamics are subject to uncertainty and shared resource constraints under a broad range of conditions.

Because the proposed approaches cover different domain conditions, the results are able to complement each other: in a setting where agents have a communication channel but are nevertheless required to function decentrally, the resource preallocation algorithms can be used to provide a back-up policy to the dynamic arbiter approach. And because the mechanism underlying decoupled coordination algorithms like Column Generation and Fictitious Play operate on summary statistics, there is no restriction on mixing learning agents with agents having a fully specified model.

Nevertheless, there are also several limitations to the approaches we propose here. In the first place there are limitations which occur as a consequence of choosing Markov decision processes as the modeling paradigm; these include: the requirement of discrete state and action spaces for the agent models, the requirement of synchronous decision-making across all the agents, and the curse of dimensionality which appears when modeling objects which have a factored structure.

In the second place, we put several assumptions in place ourselves, which limit the scope of the work: agents are required to be *independent*, in terms of transition and reward functions, and agent utility is expected to be fully *interchangeable*, allowing their utility functions to be added together. Both assumptions are necessary in order for their subproblems to be separable, enabling the effective decoupling into sub-problems.

6.2 Future work directions

Although we identified several limitations in the previous section, we are convinced that the methods proposed in this thesis are sufficiently general to serve as a starting point for future work towards lifting these limitations. In this section we provide an analysis of several promising directions, pointing to related literature wherever possible. We divide our discussion into two sections: technical extensions which broaden the scope of applicability of our algorithms, and longer-term perspectives which place our work in a broader context.

6.2.1 Technical extensions

While there are many possible technical improvements to the algorithms, we discuss in the following sections the future work directions which we expect will have the largest impact: (i) relaxing the fully cooperative assumption, and (ii) relaxing the independence assumption.

Relaxing the fully cooperative assumption

Throughout this thesis, we have assumed that agents are fully cooperative, meaning that the utility obtained by one agent is interchangeable with that of other agents. In practice, how utility is distributed across agents may have significant consequences for the agents, to the point that we may be willing to sacrifice the 'global optimum' for a more 'social optimum'. We treated the problem of computing *fair* resource allocations in the discussions on Chapter 3 (Section 3.6.1), which we briefly summarize here.

One of the challenges in relaxing the fully cooperative assumption is that it is not straightforward to define a 'fair' optimality criterion; several types of distributive norms have been proposed in literature (Forsyth, 2018). Perhaps the most straightforward implementation is to maximize the utility of the least-performing agent, which can be expressed using linear inequalities (Zhang and Shah, 2014). Nevertheless, this model of fairness can only be added directly to the Constrained MDP and MILP models, which proved to be the least scalable of the preallocation algorithms we evaluated.

For the other algorithms, the use of *multi-objective* planning may be a promising route. Multi-objective planners compute multiple policies which together cover any possible weighting of the objectives (Roijers et al., 2013). By treating the resource usage on each constraint as a separate objective, we can obtain the set of all useful policies per agent. Then, we can perform policy selection in a master routine, using the fairness objective to arrive at a fair mix of policies for all agents.

In computing fair policies using the methods sketched above, we are assuming a weaker form of cooperation, where the agents are still jointly optimizing their policies with the 'stronger' agents foregoing some of their utility to improve the utility of the 'weaker' agents. We can also imagine attempting control in settings where agents are more self-interested; as this perspective introduces an aspect of *competition* over the resources between the agents, we should look at algorithms for integrated cooperation and competition. Recently, Wray, Kumar, and Zilberstein (2018) proposed the cooperative-competitive process, a model which may be adapted to include constraints, to study more competitive settings.

Relaxing the independence assumption

Another assumption underlying the model and algorithms in this thesis is the complete independence of agents from one another, outside of the shared constraints, meaning that one agent can not influence the transitions or rewards of another. This allows for efficient algorithms based on decoupling the agents, but it comes at the cost of restricting the expressiveness of the models. For example, we cannot model the box-pushing problem, where two robots are required to work together in order to push a large box (Kube and Zhang, 1997).

Of course, when all agents interact with all other agents, there is no potential to decouple them, in which case the problem simply becomes one of solving very large

MDPs approximately, for example through the use of Monte Carlo Tree Search (Browne et al., 2012). The more interesting scenario is when agents have only *sparse* additional interactions. Taking the recurring example of controlling an aggregation of heat pumps, we can imagine that two houses which do not share a wall are indeed independent; but neighboring houses may transfer heat amongst each other. Thus, in this domain, an agent can have sparse interactions with its neighboring agents, while at the same time participating in the global constraint.

Several approaches have been proposed in literature to exploit such sparse agent interactions (Guestrin et al., 2003; Oliehoek, Witwicki, and Kaelbling, 2012; Oliehoek, Spaan, and Witwicki, 2015). The most promising strategy to include sparse interactions between agents in our global constraint framework seems to be the influence-based abstractions of Oliehoek, Witwicki, and Kaelbling (2012), which transforms the multi-agent problem into a collection of influence-augmented single-agent models. Because an influence-augmented model constitutes a best-response to an earlier policy of the influencing agents (i.e., the neighbors of the house under consideration), we expect that this approach may integrate well with the dynamic resource allocation algorithms proposed in Chapter 4.

6.2.2 Perspectives and opportunities

Besides the technical improvements suggested in the previous section, we also propose future work which is more speculative in nature; these directions of future work should be seen as research projects in their own right. As this thesis has focused on the interaction between autonomous agents, it should perhaps come as no surprise that these perspectives relate primarily to the interaction between the human user and the autonomous agent.

Hierarchical planning to cover multiple timescales

Throughout this thesis we have considered operational planning domains, where decisions are taken with decision intervals on the order of several minutes. While there are no a priori restrictions on the length of the decision interval, the problem becomes more difficult when we want to optimize for *multiple* timescales. For example, the distribution network operator controlling household thermostats may want to optimize its yearly grid investments over time under a limited budget, by considering of the consequences of its investments on the thermostat planning problem. Treating this as one large minute-scale planning problem is clearly intractable, as a 10 year problem would have over a 100.000 constrained decision points.

As a first solution, we may consider creating a decision support tool, using the thermostat planning problem to simulate the consequences of the human planner's decisions. However, if we want to optimize multi-timescale decisions automatically, we should instead look at abstracting the thermostat planning problem under different investments as 'macro actions' (Hauskrecht et al., 1998). A specific challenge in the constrained investment case is that the macro-action space may be too large to compute new policies for each investment option. One option may be to use learning algorithms to automatically explore promising macro actions (Newton et al., 2007). Alternatively, we may try to exploit the structure of the constrained problem, by re-using the thermostat control policies computed for one constraint setting to bootstrap a relaxed constraint.

Preventing users from misrepresenting their preferences

Whether we optimize for a global optimum or with another objective function, a shared risk is that the user will misrepresent its reward function in an effort to allow the agent obtain more resources than it would achieve in case the user reported its interests honestly. To prevent users from acting strategically, we would need to integrate results from mechanism design into our allocation algorithms. Gerding et al. (2011) demonstrate that the allocation of power to multiple competing electric vehicles can be made strategy-proof, but without considering uncertainty or anticipatory (planning) control. Scott and Thiébaux (2017) investigate a more complicated receding horizon setting of allocating energy to households which operate under uncertainty. They expect that the *potential* for manipulation cannot be avoided in light of uncertainty about reported values, but show that manipulation can be *detected*, allowing for post-hoc penalties to be applied to encourage truthfulness. It is currently an open question whether it is possible to enforce truthfulness a priori in a constrained multi-agent planning under uncertainty setting. Solving this problem is essential to guarantee that an optimized solution actually achieves the social optimum.

Effective interaction between autonomous agent and human user

Several of the use-cases of multi-agent autonomous systems studied in this thesis see the agent acting 'on behalf' of a user, whether that user is a household in the thermostatically controlled loads domain, or a tourist in the recommender system domain. For users to put trust in the system and remain engaged it is important for the interface between agent and user to be sufficiently expressive; it may be necessary for the agent to be able to *explain* its decisions, especially considering that decisions may appear sub-optimal as a consequence of the agent yielding resources in favor of another agent. One way to avoid this problem is to plan 'human-aware' policies, which see the agent maintaining a predictive model of the human operator (Chakraborti et al., 2016), to find actions which result in minimum conflicts between human and agent plans. However, in our setting where the agent acts on behalf of the user, it may be more practical to let the agent optimize the user's emotional state towards contentment. We expect this requires the integration of automatic sentiment recognition, which remains an unsolved question itself (Tian, Lai, and Moore, 2018), with an adaptive plan explanation (Fox, Long, and Magazzeni, 2017), based on the current mood and expertise of the user with the system.

Bibliography

Adelman, Daniel and Adam J. Mersereau (2008). “Relaxations of weakly coupled stochastic dynamic programs”. In: *Operations Research* 56.3, pp. 712–727. ISSN: 0030-364X.

Agrawal, Pritee, Pradeep Varakantham, and William Yeoh (2016). “Scalable greedy algorithms for task/resource constrained multi-agent stochastic planning”. In: *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pp. 10–16.

Akbar, Md Mostofa, M. Sohel Rahman, M. Kaykobad, E. G. Manning, and G. C. Shoja (2006). “Solving the multidimensional multiple-choice knapsack problem by constructing convex hulls”. In: *Computers & Operations Research* 33.5, pp. 1259–1273. ISSN: 0305-0548.

Altman, Eitan (1999). *Constrained Markov decision processes*. Ed. by Marco Scarsini, Moshe Shaked, and Shaler Stidham Jr. Stochastic Modeling. Chapman & Hall/CRC. ISBN: 0849303826.

Araya-López, Mauricio, Vincent Thomas, Olivier Buffet, and François Charpillet (2010). “A Closer Look at MOMDPs”. In: *Proceedings of the 22nd IEEE International Conference on Tools with Artificial Intelligence*. Vol. 2, pp. 197–204.

Bai, Haoyu, David Hsu, Wee Sun Lee, and Vien A. Ngo (2011). “Monte Carlo value iteration for continuous-state POMDPs”. In: *Algorithmic Foundations of Robotics IX*. Ed. by David Hsu, Volkan Isler, Jean-Claude Latombe, and Ming C. Lin. Vol. 68. STAR. Heidelberg: Springer, pp. 175–191.

Becker, Raphen, Shlomo Zilberstein, Victor Lesser, and Claudia V. Goldman (2004). “Solving transition independent decentralized Markov decision processes”. In: *Journal of Artificial Intelligence Research* 22, pp. 423–455. ISSN: 10769757.

Bell, Robert, Yehuda Koren, and Chris Volinsky (Dec. 2007). “Chasing $1,000,000: how we won the Netflix progress prize”. In: *Statistical Computer & Graphics* 12.2, pp. 4–12.

Bellemare, Marc G., Will Dabney, and Rémi Munos (2017). “A distributional perspective on reinforcement learning”. In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. Proceedings of Machine Learning Research. PMLR, pp. 449–458.

Bellman, Richard E. (1957a). “A Markovian Decision Process”. In: *Journal of Mathematics and Mechanics* 6.5, pp. 679–684.

— (1957b). *Dynamic programming*. Princeton NJ, Princeton, University Press.

Berger, Ulrich (2007). “Brown’s original fictitious play”. In: *Journal of Economic Theory* 135.1, pp. 572–578.

Bernstein, Daniel S., Robert Givan, Neil Immerman, and Shlomo Zilberstein (2002). “The Complexity of Decentralized Control of Markov Decision Processes”. In: *Mathematics of Operations Research* 27.4, pp. 819–840.

Bertsekas, Dimitri P. and John N. Tsitsiklis (2008). *Introduction to Probability*. 2nd ed. Athena Scientific, Belmont, MA.

Beutler, Frederick J. and Keith W. Ross (1985). “Optimal policies for controlled Markov chains with a constraint”. In: *Journal of Mathematical Analysis and Applications* 112.1, pp. 236–252. ISSN: 0022-247X.

Blau, Roger A. (1974). “Stochastic Programming and Decision Analysis: An Apparent Dilemma”. In: *Management Science* 21.3, pp. 271–276.

Boutilier, Craig (1996). “Planning, Learning and Coordination in Multiagent Decision Processes”. In: *Proceedings of the 6th Conference on Theoretical Aspects of Rationality and Knowledge*, pp. 195–210.

Boutilier, Craig, Thomas Dean, and Steve Hanks (1999). “Decision-theoretic planning: structural assumptions and computational leverage”. In: *Journal of Artificial Intelligence Research* 11, pp. 1–94.

Boutilier, Craig and Tyler Lu (2016). “Budget Allocation using Weakly Coupled, Constrained Markov Decision Processes”. In: *Proceedings of the 32nd Conference on Uncertainty in Artificial Intelligence*, pp. 52–61.

Brafman, Ronen I. and Moshe Tennenholtz (2002). “R-MAX – A general polynomial time algorithm for near-optimal reinforcement learning”. In: *Journal of Machine Learning Research* 3, pp. 953–958.

Browne, Cameron B., Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton (Mar. 2012). “A survey of Monte Carlo tree search methods”.

In: *IEEE Transactions on Computational Intelligence and AI in Games* 4.1, pp. 1–43. ISSN: 1943-068X.

Castañon, David A. (1997). "Approximate dynamic programming for sensor management". In: *Proceedings of the 36th IEEE Conference on Decision and Control*. Vol. 5, pp. 1202–1207. ISBN: 0780339708197.

Cevik, Mucahit, Turgay Ayer, Oguzhan Alagoz, and Brian L. Sprague (2018). "Analysis of mammography screening policies under resource constraints". In: *Production and Operations Management*. ISSN: 10591478.

Chadès, Iadine, Josie Carwardine, Tara G. Martin, Samuel Nicol, Régis Sabbadin, and Olivier Buffet (2012). "MOMDPs: A solution for modelling adaptive management problems". In: *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pp. 267–273.

Chakraborti, Tathagata, Yu Zhang, David E. Smith, and Subbarao Kambhampati (2016). "Planning with resource conflicts in human-robot cohabitation". In: *Proceedings of the 2016 International Conference on Autonomous Agents and Multiagent Systems*. IFAAMAS, pp. 1069–1077.

Charnes, A. and W. W. Cooper (1959). "Chance-Constrained Programming". In: *Management Science* 6.1, pp. 73–79.

Chatterjee, Krishnendu, Petr Novotný, Guillermo A. Pérez, Jean-François Raskin, and Đorđe Žikelić (2017). "Optimizing expectation with guarantees in POMDPs". In: *Proceedings of the 31st AAAI Conference of Artificial Intelligence*, pp. 3725–3732.

Chen, Shaofei, Feng Wu, Lincheng Shen, Jing Chen, and Sarvapali D. Ramchurn (Dec. 2016). "Decentralized patrolling under constraints in dynamic environments". In: *IEEE Transactions on Cybernetics* 46.12, pp. 3364–3376. ISSN: 21682275.

Cheng, Shih-Fen, Larry Lin, Jiali Du, Hoong Chuin Lau, and Pradeep Varakantham (Dec. 2013). "An agent-based simulation approach to experience management in theme parks". In: *Winter Simulation Conference*, pp. 1527–1538.

Chernoff, Herman (1952). "A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the Sum of Observations". In: *The Annals of Mathematical Statistics* 23.4, pp. 493–507.

Cimatti, Alessandro, Enrico Giunchiglia, and Paolo Traverso (1997). "Planning via model checking: A decision procedure for AR". In: *Recent Advances in AI Planning*. Ed. by Sam Steel and Rachid Alami. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 130–142. ISBN: 978-3-540-69665-0.

Cohen, Andrew, Lei Yu, and Robert Wright (2018). “Diverse Exploration for Fast and Safe Policy Improvement”. In: *Proceedings of the 32nd AAAI conference on Artificial Intelligence*.

Dearden, Richard, Nir Friedman, and David Andre (1999). “Model Based Bayesian Exploration”. In: *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*, pp. 150–159.

Dolgov, Dmitri A. and Edmund H. Durfee (2006a). “Resource allocation among agents with MDP-induced preferences”. In: *Journal of Artificial Intelligence Research* 27, pp. 505–549.

— (2006b). “Resource allocation among agents with preferences induced by factored MDPs”. In: *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 297–304.

Duff, Michael O’Gordon (2002). “Optimal learning: Computational procedures for Bayes-adaptive Markov decision processes”. PhD thesis. University of Massachusetts Amherst.

Durfee, Ed and Shlomo Zilberstein (2013). “Multiagent Planning, Control, and Execution”. In: *Multiagent Systems*. Ed. by Gerhard Weiss. 2nd ed. Cambridge, MA: The MIT Press. Chap. 11, pp. 485–546. ISBN: 9780262018890.

Fink, Eugene, P. Matthew Jennings, Ulas Bardak, Jean Oh, Stephen F. Smith, and Jaime G. Carbonell (2006). “Scheduling with Uncertain Resources: Search for a Near-Optimal Solution”. In: *Proc. of the 19th IEEE Intl. Conf. on Systems, Man and Cybernetics*, pp. 137–144. ISBN: 1-4244-0099-6.

Forsyth, Donelson R. (2018). “Conflict”. In: *Group dynamics*. 7th ed. Cengage Learning. Chap. 13, pp. 409–443.

Fox, Maria, Derek Long, and Daniele Magazzeni (2017). “Explainable planning”. In: *Proceedings of the IJCAI-17 workshop on Explainable AI*, pp. 24–30.

García, Javier and Fernando Fernández (2015). “A comprehensive survey on safe reinforcement learning”. In: *Journal of Machine Learning Research* 16, pp. 1437–1480.

Gerding, Enrico H, Valentin Robu, Sebastian Stein, David C Parkes, Alex Rogers, and Nicholas R Jennings (2011). “Online mechanism design for electric vehicle charging”. In: *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*. IFAAMAS, pp. 811–818.

Gilmore, P. C. and R. E. Gomory (1961). “A linear programming approach to the cutting-stock problem”. In: *Operations Research* 9.6, pp. 849–859.

Gopalan, Aditya and Shie Mannor (July 2015). “Thompson Sampling for Learning Parameterized Markov Decision Processes”. In: *Proceedings of The 28th Conference on Learning Theory*. Ed. by Peter Grünwald, Elad Hazan, and Satyen Kale. Vol. 40. Paris, France: PMLR, pp. 861–898.

Gordon, Geoffrey J., Pradeep Varakantham, William Yeoh, Hoong Chuin Lau, Ajay S. Aravamudhan, and Shih-Fen Cheng (2012). “Lagrangian Relaxation for Large-Scale Multi-agent Planning”. In: *WI-IAT*, pp. 494–501. ISBN: 978-1-4673-6057-9.

Guestrin, Carlos, Daphne Koller, Ronald Parr, and Shobha Venkataraman (2003). “Efficient solution algorithms for factored MDPs”. In: *Journal of Artificial Intelligence Research* 19, pp. 399–468.

Guez, Arthur, David Silver, and Peter Dayan (2013). “Scalable and efficient Bayes-adaptive reinforcement learning based on Monte-Carlo tree search.” In: *Journal of Artificial Intelligence Research* 48, pp. 841–883.

Gupta, Tarun, Akshat Kumar, and Praveen Paruchuri (2018). “Planning and Learning For Decentralized MDPs With Event Driven Rewards”. In: *Proceedings of the 32nd AAAI Conference on Artificial Intelligence.*

Hans, Alexander, Daniel Schneegaß, Anton Maximilian Schäfer, and Steffen Udluft (2008). “Safe exploration for reinforcement learning”. In: *Proceedings of the 16th European Symposium on Artificial Neural Networks* (Bruges, Belgium, Apr. 23, 2008–Apr. 25, 2008), pp. 143–148. ISBN: 2-930307-08-0.

Haskell, William B. and Rahul Jain (2015). “A Convex Analytic Approach to Risk-Aware Markov Decision Processes”. In: *SIAM Journal on Control and Optimization* 53.3, pp. 1569–1598.

Hauskrecht, Milos and Hamish Fraser (Mar. 2000). “Planning treatment of ischemic heart disease with partially observable Markov decision processes”. In: *Artificial Intelligence in Medicine* 18.3, pp. 221–244. ISSN: 09333657.

Hauskrecht, Milos, Nicolas Meuleau, Leslie Pack Kaelbling, Thomas Dean, and Craig Boutilier (1998). “Hierarchical solution of Markov decision processes using macro-actions”. In: *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*. AUAI Press, pp. 220–229.

He, Shan, Mark Wallace, Graeme Gange, Ariel Liebman, and Campbell Wilson (2018). “A fast and scalable algorithm for scheduling large numbers of devices under real-time pricing”. In: *Proceedings of the 24th International Conference on Principles and Practice of Constraint Programming*. Ed. by John Hooker. Lecture Notes in Computer Science, vol 11008. Springer, Cham, pp. 649–666.

Hoeffding, Wassily (1963). “Probability inequalities for sums of bounded random variables”. In: *Journal of the American Statistical Association* 58.301, pp. 13–30.

Howard, Ronald A. (1960). *Dynamic programming and Markov processes*. MIT Press.

— (Aug. 1966). “Information value theory”. In: *IEEE Transactions on Systems Science and Cybernetics* 2.1, pp. 22–26.

Hoy, Matthew B. (Jan. 2018). “Alexa, Siri, Cortana, and more: An introduction to voice assistants”. In: *Medical Reference Services Quarterly* 37.1, pp. 81–88. ISSN: 0276-3869.

Isom, Joshua D., Sean P. Meyn, and Richard D. Braatz (2008). “Piecewise linear dynamic programming for constrained POMDPs”. In: *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pp. 291–296.

Jain, Manish, Erim Kardes, Christopher Kiekintveld, Fernando Ordónez, and Milind Tambe (2010). “Security Games with Arbitrary Schedules: A Branch and Price Approach”. In: *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, pp. 792–797.

Kaelbling, Leslie Pack, Michael L. Littman, and Anthony R. Cassandra (May 1998). “Planning and acting in partially observable stochastic domains”. In: *Artificial Intelligence* 101.1-2, pp. 99–134. ISSN: 00043702.

Kakade, Sham Machandranath (2003). “On the Sample Complexity of Reinforcement Learning”. PhD thesis. University College London.

Kallenberg, L. C. M. (1983). *Linear programming and finite Markovian control problems*. Mathematical Centre Tracts 148. Amsterdam: Mathematisch Centrum. ISBN: 90-6196-236-6.

Kearns, Michael and Satinder Singh (2002). “Near-optimal reinforcement learning in polynomial time”. In: *Machine Learning* 49.2, pp. 209–232.

Kim, Dongho, Kee-Eung Kim, and Pascal Poupart (2012). “Cost-sensitive exploration in Bayesian reinforcement learning”. In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger. Curran Associates, Inc., pp. 3068–3076.

Kim, Dongho, Jaesong Lee, Kee-Eung Kim, and Pascal Poupart (2011). “Point-based value iteration for constrained POMDPs”. In: *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pp. 1968–1974. ISBN: 9781577355120.

Kochenderfer, Mykel J. (2015). *Decision Making Under Uncertainty*. The MIT Press. ISBN: 978-0-262-02925-4.

Kube, C. Ronald and Hong Zhang (1997). “Task modelling in collective robotics”. In: *Autonomous Robots* 4.1, pp. 53–72.

Kumar, Rajiv Ranjan, Pradeep Varakantham, and Akshat Kumar (2017). “Decentralized planning in stochastic environments with submodular rewards”. In: *Proceedings of the 31st AAAI Conference on Artificial Intelligence*. AAAI Press, pp. 3021–3028.

Kurniawati, Hanna, David Hsu, and Wee Sun Lee (2008). “SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces”. In: *Robotics: Science and Systems*. Zurich, Switzerland.

Lee, Jongmin, Youngsoo Jang, Pascal Poupart, and Kee-Eung Kim (2017). “Constrained Bayesian reinforcement learning via approximate linear programming”. In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pp. 2088–2095.

Littman, Michael L. (1994). “Markov games as a framework for multi-agent reinforcement learning”. In: *Proceedings of the 11th International Conference on Machine Learning*. Elsevier, pp. 157–163.

Littman, Michael L., Thomas L. Dean, and Leslie Pack Kaelbling (1995). “On the complexity of solving Markov decision problems”. In: *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*, pp. 394–402.

Luedtke, James and Shabbir Ahmed (2008). “A Sample Approximation Approach for Optimization with Probabilistic Constraints”. In: *SIAM Journal on Optimization* 19.2, pp. 674–699.

Martin, James John (1967). *Bayesian decision problems and Markov chains*. Publications in operations research. New York: Wiley.

Martin, Péron, Kai Helge Becker, Peter Bartlett, and Iadine Chadès (2017). “Fast-Tracking Stationary MOMDPs for Adaptive Management Problems”. In: *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pp. 4531–4537.

Mausam, Emmanuel Benazera, Ronen Brafman, Nicolas Meuleau, and Eric A. Hansen (2005). “Planning with continuous resources in stochastic domains”. In: *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pp. 1244–1251.

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

Möller, Judith, Damian Trilling, Natali Helberger, and Bram van Es (July 2018). “Do not blame it on the algorithm: an empirical assessment of multiple recommender systems and their impact on content diversity”. In: *Information, Communication & Society* 21.7, pp. 959–977. ISSN: 1369-118X.

Mortensen, R. E. and K. P. Haggerty (1988). “A stochastic computer model for heating and cooling loads”. In: *IEEE Transactions on Power Systems* 3.3, pp. 1213–1219.

Moser, Martin, Dusan P. Jokanovic, and Norio Shiratori (1997). “An algorithm for the multidimensional multiple-choice knapsack problem”. In: *IEICE transactions on fundamentals of electronics, communications and computer sciences* E80-A.3, pp. 582–589.

Nair, Ranjit, Milind Tambe, Maayan Roth, and Makoto Yokoo (2004). “Communication for Improving Policy Computation in Distributed POMDPs”. In: *Proc. of the 3rd Intl. Conf. on Autonomous Agents and Multi Agent Systems*, pp. 1098–1105.

Newton, Muhammad Abdul Hakim, John Levine, Maria Fox, and Derek Long (2007). “Learning macro-actions for arbitrary planners and domains”. In: *Proceedings of the 17th International Conference on Automated Planning and Scheduling*. AAAI Press, pp. 256–263.

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

De Nijs, Frits, Matthijs T. J. Spaan, and Mathijs M. de Weerdt (2018). “Preallocation and planning under stochastic resource constraints”. In: *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*. AAAI Press, pp. 4662–4669.

— (2015). “Best-response planning of thermostatically controlled loads under power constraints”. In: *Proceedings of the 29th AAAI Conference on Artificial Intelligence*. AAAI Press, pp. 615–621.

— (2016). “Decoupling a resource constraint through fictitious play in multi-agent sequential decision making”. In: *Proceedings of the 22nd European Conference on Artificial Intelligence*. Vol. 285. Frontiers in Artificial Intelligence and Applications. IOS Press, pp. 1724–1725.

De Nijs, Frits, Georgios Theocharous, Nikos Vlassis, Mathijs M. de Weerdt, and Matthijs T. J. Spaan (2018). “Capacity-aware sequential recommendations”. In: *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems*. Stockholm, Sweden, July 10–15: IFAAMAS, pp. 416–424.

Oliehoek, Frans A. and Matthijs T. J. Spaan (2012). “Tree-based Solution Methods for Multiagent POMDPs with Delayed Communication”. In: *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pp. 1415–1421.

Oliehoek, Frans A., Matthijs T. J. Spaan, and Stefan J. Witwicki (2015). “Factored upper bounds for multiagent planning problems under uncertainty with non-factored value functions”. In: *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pp. 1645–1651.

Oliehoek, Frans A., Stefan J. Witwicki, and Leslie P. Kaelbling (2012). “Influence-based Abstraction for Multiagent Systems”. In: *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pp. 1422–1428.

Ong, Sylvie C. W., Shao Wei Png, David Hsu, and Wee Sun Lee (2010). “Planning under Uncertainty for Robotic Tasks with Mixed Observability”. In: *The International Journal of Robotics Research* 29.8, pp. 1053–1068. ISSN: 0278-3649.

Ono, Masahiro, Marco Pavone, Yoshiaki Kuwata, and J. Balaram (2015). “Chance-constrained dynamic programming with application to risk-aware robotic space exploration”. In: *Autonomous Robots* 39.4, pp. 555–571. ISSN: 15737527.

Osband, Ian and Benjamin Van Roy (2014). “Near-optimal Reinforcement Learning in Factored MDPs”. In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger. Curran Associates, Inc., pp. 604–612.

Palensky, Peter and Dietmar Dietrich (2011). “Demand side management: Demand response, intelligent energy systems, and smart loads”. In: *IEEE Transactions on Industrial Informatics* 7.3, pp. 381–388. ISSN: 1551-3203.

Papadimitriou, Christos H. and John N. Tsitsiklis (1987). “The Complexity of Markov Decision Processes”. In: *Mathematics of Operations Research* 12.3, pp. 441–450.

Pinson, Pierre, Henrik Madsen, Henrik Aa. Nielsen, George Papaefthymiou, and Bernd Klöckl (Jan. 2009). “From probabilistic forecasts to statistical scenarios of short-term wind power production”. In: *Wind Energy* 12.1, pp. 51–62. ISSN: 10954244.

Porta, Josep M., Nikos Vlassis, Matthijs T. J. Spaan, and Pascal Poupart (Nov. 2006). “Point-based value iteration for continuous POMDPs”. In: *Journal of Machine Learning Research* 7, pp. 2329–2367.

Poupart, Pascal, Aarti Malhotra, Pei Pei, Kee-Eung Kim, Bongseok Goh, and Michael Bowling (2015). “Approximate Linear Programming for Constrained Partially Observable Markov Decision Processes”. In: *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pp. 3342–3348.

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

Resnick, Paul and Hal R. Varian (Mar. 1997). “Recommender Systems”. In: *Communications of the ACM* 40.3, pp. 56–58. ISSN: 0001-0782.

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

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

Ron, Dana, Yoram Singer, and Naftali Tishby (1996). "The Power of Amnesia: Learning Probabilistic Automata with Variable Memory Length". In: *Machine Learning* 25, pp. 117–149.

Rossman, Lewis A. (1977). "Reliability-constrained dynamic programing and randomized release rules in reservoir management". In: *Water Resources Research* 13.2, pp. 247–255. ISSN: 19447973.

Ruelens, Frederik, Sandro Iacovella, Bert Claessens, and Ronnie Belmans (Aug. 2015). "Learning agent for a heat-pump thermostat with a set-back strategy using model-free reinforcement learning". In: *Energies* 8.8, pp. 8300–8318. ISSN: 1996-1073.

Russell, Stuart J. and Peter Norvig (2016). "Making complex decisions". In: *Artificial Intelligence: A Modern Approach*. Pearson Education Limited. Chap. 17, pp. 645–692.

Santana, Pedro, Sylvie Thiébaux, and Brian Williams (2016). "RAO*: An algorithm for chance-constrained POMDP's". In: *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pp. 3308–3314.

Santana, Pedro, Tiago Vaquero, Catharine L. McGhan, Claudio Toledo, Eric Timmons, Brian Williams, and Richard Murray (2016). "Risk-aware planning in hybrid domains: an application to autonomous planetary rovers". In: *AIAA SPACE 2016*.

Schafer, J. Ben, Dan Frankowski, Jon Herlocker, and Shilad Sen (2007). "Collaborative filtering recommender systems". In: *The Adaptive Web, LNCS 4321*. Ed. by P. Brusilovsky, A. Kobsa, and W. Nejdl. Springer-Verlag Berlin Heidelberg, pp. 291–324.

Schaffer, Steve R., Bradley J. Clement, and Steve A. Chien (2005). "Probabilistic reasoning for plan robustness". In: *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pp. 1266–1271.

Scott, Paul (2016). "Distributed Coordination and Optimisation of Network-Aware Electricity Prosumers". PhD thesis. The Australian National University.

Scott, Paul and Sylvie Thiébaux (2017). "Identification of manipulation in receding horizon electricity markets". In: *IEEE Transactions on Smart Grid*. ISSN: 1949-3053.

Scott, Paul, Sylvie Thiébaux, Menkes van den Briel, and Pascal Van Hentenreyck (2013). "Residential demand response under uncertainty". In: *Proceedings of the 19th Inter-*

national Conference on Principles and Practice of Constraint Programming, pp. 645–660.

Shani, Guy, David Heckerman, and Ronen I. Brafman (2005). “An MDP-Based Recommender System”. In: *Journal of Machine Learning Research* 6. Ed. by Craig Boutilier, pp. 1265–1295.

Silver, David, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis (Oct. 2017). “Mastering the game of Go without human knowledge”. In: *Nature* 550, pp. 354–359. ISSN: 0028-0836.

Silver, Edward Allan (Aug. 1963). “Markovian decision processes with uncertain transition probabilities or rewards”. PhD thesis. Massachusetts Institute of Technology.

Sinha, Prabhakant and Andris A. Zoltners (1979). “The multiple-choice knapsack problem”. In: *Operations Research* 27.3, pp. 503–515.

Song, Wen, Donghun Kang, Jie Zhang, and Hui Xi (2018). “Risk-aware Proactive Scheduling via Conditional Value-at-Risk”. In: *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*.

Spaan, Matthijs T. J., Frans A. Oliehoek, and Nikos Vlassis (2008). “Multiagent Planning under Uncertainty with Stochastic Communication Delays”. In: *Proc. of the 8th Intl. Conf. on Automated Planning and Scheduling*, pp. 338–345.

Staid, Andrea, Jean-Paul Watson, Roger J.-B. Wets, and David L. Woodruff (2017). “Generating short-term probabilistic wind power scenarios via nonparametric forecast error density estimators”. In: *Wind Energy* 20.12, pp. 1911–1925.

Steck, Harald (2013). “Evaluation of Recommendations: Rating-prediction and Ranking”. In: *Proceedings of the 7th ACM Conference on Recommender Systems*. RecSys ’13. New York, NY, USA: ACM, pp. 213–220. ISBN: 978-1-4503-2409-0.

Strens, Malcolm J. A. (2000). “A Bayesian Framework for Reinforcement Learning”. In: *Proceedings of the 17th International Conference on Machine Learning*, pp. 943–950.

Sutton, Richard S. and Andrew G. Barto (2018). *Reinforcement Learning: An introduction*. 2nd ed. The MIT Press.

Tehrani, Pouya, Keqin Liu, and Qing Zhao (2012). “Opportunistic spectrum access in unslotted primary systems”. In: *Journal of The Franklin Institute* 349.3, pp. 985–1010. ISSN: 00160032.

Theocharous, Georgios, Nikos Vlassis, and Zheng Wen (2017). “An Interactive Points of Interest Guidance System”. In: *Proceedings of the 22nd International Conference* *on Intelligent User Interfaces Companion*. IUI ’17 Companion. New York, NY, USA: ACM, pp. 49–52. ISBN: 978-1-4503-4893-5.

Thomee, Bart, David A. Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li (2016). “YFCC100M: the new data in multimedia research”. In: *Communications of the ACM* 59.2, pp. 64–73.

Thompson, William R. (1933). “On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples”. In: *Biometrika* 25.3/4, pp. 285–294. ISSN: 00063444.

Tian, Leimin, Catherine Lai, and Johanna D. Moore (2018). “Polarity and intensity: the two aspects of sentiment analysis”. In: *Proceedings of the 1st Grand Challenge and Workshop on Human Multimodal Language*, pp. 40–47.

Tusch, Guenter (2000). “Optimal sequential decisions in liver transplantation based on a POMDP model”. In: *Proceedings of the 14th European Conference on Artificial Intelligence*. Ed. by Werner Horn. IOS Press, pp. 186–190.

Undurti, Aditya and Jonathan P. How (2010). “An online algorithm for constrained POMDPs”. In: *Proceedings of the 2010 IEEE International Conference on Robotics and Automation*, pp. 3966–3973. ISBN: 9781424450381.

Varakantham, Pradeep, Shih-Fen Cheng, Geoff Gordon, and Asrar Ahmed (2012). “Decision support for agent populations in uncertain and congested environments”. In: *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pp. 1471–1477.

Vrancx, Peter, Katja Verbeeck, and Ann Nowé (Aug. 2008). “Decentralized learning in Markov games”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 38.4, pp. 976–981. ISSN: 1083-4419.

Walraven, Erwin and Matthijs T. J. Spaan (2018). “Column Generation Algorithms for Constrained POMDPs”. In: *Journal of Artificial Intelligence Research*.

De Weerdt, Mathijs M., Sebastian Stein, Enrico H. Gerding, Valentin Robu, and Nicholas R. Jennings (2015). “Intention-Aware Routing of Electric Vehicles”. In: *IEEE Transactions on Intelligent Transportation Systems* 17.5, pp. 1472–1482. ISSN: 15249050.

Witwicki, Stefan J. and Edmund H. Durfee (2010). “Influence-Based Policy Abstraction for Weakly-Coupled Dec-POMDPs”. In: *ICAPS*, pp. 185–192.

Witwicki, Stefan J., Frans A. Oliehoek, and Leslie P. Kaelbling (2012). “Heuristic search of multiagent influence space”. In: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*. Vol. 2. IFAAMAS, pp. 973–980.

Wray, Kyle Hollins, Akshat Kumar, and Shlomo Zilberstein (2018). “Integrated cooperation and competition in multi-agent decision making”. In: *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*.

Wu, Feng, Nicholas R. Jennings, and Xiaoping Chen (2012). “Sample-based policy iteration for constrained DEC-POMDPs”. In: *Proceedings of the 20th European Conference on Artificial Intelligence*. Vol. 242. Frontiers in Artificial Intelligence and Applications. IOS Press, pp. 858–863.

Wu, Jianhui and Edmund H. Durfee (2010). “Resource-driven mission-phasing techniques for constrained agents in stochastic environments”. In: *Journal of Artificial Intelligence Research* 38, pp. 415–473.

Yost, Kirk A. and Alan R. Washburn (2000). “The LP/POMDP marriage: optimization with imperfect information”. In: *Naval Research Logistics* 47.8, pp. 607–619. ISSN: 0894069X.

Zemel, Eitan (1984). “An $O(n)$ algorithm for the linear multiple choice knapsack problem and related problems”. In: *Information Processing Letters* 18.3, pp. 123–128. ISSN: 0020-0190.

Zhang, Chongjie and Julie A. Shah (2014). “Fairness in multi-agent sequential decision-making”. In: *Advances in Neural Information Processing Systems*, pp. 2636–2644.

Zhang, Qi, Edmund Durfee, and Satinder P. Singh (2017). “Minimizing Maximum Regret in Commitment Constrained Sequential Decision Making”. In: *Proceedings of the 27th International Conference on Automated Planning and Scheduling*. Ed. by Laura Barbulescu, Jeremy Frank, Mausam, and Stephen F. Smith.

Curriculum Vitæ

Frits de Nijs was born in 's-Gravenhage on Thursday, 13 March 1986. He followed secondary education at the 'Sint-Maartenscollege' in Voorburg from 1998 to 2004, obtaining his VWO-gymnasium diploma in the category Nature and Technology.

Starting in September 2004, he studied at the Delft University of Technology, first obtaining his B.Sc. degree in Computer Science and Engineering in 2010, followed by his M.Sc degree in Computer Science in 2013, specializing in Algorithmics. His B.Sc. project was performed as an internship for SecureSpace in Auckland, New Zealand, from April–August of 2007. As a master student, he published three conference papers, two of which as first author; his paper on his M.Sc. thesis research on scheduling resource-constrained projects received an honourable mention for best paper at the 2014 ICAPS conference.

Subsequently, Frits joined the Algorithmics group at the Delft University of Technology as a Ph.D. candidate, from November of 2013 until July of 2018. His doctoral research resulted in five peer-reviewed international conference publications as first author, as well as several workshop presentations; twice his work received the runner-up award at the Erasmus energy forum science day. During his time as a Ph.D. candidate, Frits was involved in teaching, including joint supervision of one M.Sc. thesis project.

His Ph.D. research was performed in collaboration with its funding company, Alliander, including disseminating the research to stakeholders and connecting the algorithms with a demonstrator system. He spent May–August of 2017 as a data science intern at the Adobe Research division of Adobe Systems, in San Jose, California. Since July of 2018, Frits is employed as an assistant lecturer at Monash University in Melbourne, Australia.

In his spare time, Frits likes to practice the martial art Ryu Kyu Kobujutsu, having achieved black-belt grade in 2015. Prior to that, he became Dutch champion Kobujutsu in the junior category in 2013.

List of Publications

8. Frits de Nijs, Georgios Theocharous, Nikos Vlassis, Mathijs M. de Weerdt, and Matthijs T. J. Spaan (2018). “Capacity-aware sequential recommendations”. In: *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems*. Stockholm, Sweden, July 10–15: IFAAMAS, pp. 416–424

7. Frits de Nijs, Matthijs T. J. Spaan, and Mathijs M. de Weerdt (2018). “Preallocation and planning under stochastic resource constraints”. In: *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*. AAAI Press, pp. 4662–4669

6. Frits de Nijs, Erwin Walraven, Mathijs M. de Weerdt, and Matthijs T. J. Spaan (2017). “Bounding the probability of resource constraint violations in multi-agent MDPs”. In: *Proceedings of the 31st AAAI Conference on Artificial Intelligence*. AAAI Press, pp. 3562–3568

5. Frits de Nijs, Matthijs T. J. Spaan, and Mathijs M. de Weerdt (2016). “Decoupling a resource constraint through fictitious play in multi-agent sequential decision making”. In: *Proceedings of the 22nd European Conference on Artificial Intelligence*. Vol. 285. Frontiers in Artificial Intelligence and Applications. IOS Press, pp. 1724–1725

4. Frits de Nijs, Matthijs T. J. Spaan, and Mathijs M. de Weerdt (2015). “Best-response planning of thermostatically controlled loads under power constraints”. In: *Proceedings of the 29th AAAI Conference on Artificial Intelligence*. AAAI Press, pp. 615–621

3. Frits de Nijs and Tomas Klos (2014). “A novel priority rule heuristic: learning from justification”. In: *Proceedings of the 24th International Conference on Automated Planning and Scheduling*. AAAI Press, pp. 92–100

2. Adriaan ter Mors, Cees Witteveen, Charlotte Ipema, Frits de Nijs, and Theodor Tsiourakis (2012). “Empirical evaluation of multi-agent routing approaches”. In: *Proceedings of the 2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*. IEEE, pp. 305–309

1. Frits de Nijs, Daan Wilmer, and Tomas Klos (2012). “Evaluation and improvement of Laruelle-Widgrén inverse Banzhaf approximation”. In: *Proceedings of the 24th Benelux Conference on Artificial Intelligence*, pp. 194–201

SIKS Dissertation Series

2011

01 Botond Cseke (RUN), *Variational Algorithms for Bayesian Inference in Latent Gaussian Models*

02 Nick Tinnemeier (UU), *Organizing Agent Organizations. Syntax and Operational Semantics of an Organization-Oriented Programming Language*

03 Jan Martijn van der Werf (TU/e), *Compositional Design and Verification of Component-Based Information Systems*

04 Hado van Hasselt (UU), *Insights in Reinforcement Learning; Formal analysis and empirical evaluation of temporal-difference*

05 Bas van der Raadt (VU), *Enterprise Architecture Coming of Age - Increasing the Performance of an Emerging Discipline.*

06 Yiwen Wang (TU/e), *Semantically-Enhanced Recommendations in Cultural Heritage*

07 Yujia Cao (UT), *Multimodal Information Presentation for High Load Human Computer Interaction*

08 Nieske Vergunst (UU), *BDI-based Generation of Robust Task-Oriented Dialogues*

09 Tim de Jong (OU), *Contextualised Mobile Media for Learning*

10 Bart Bogaert (UvT), *Cloud Content Contention*

11 Dhaval Vyas (UT), *Designing for Awareness: An Experience-focused HCI Perspective*

12 Carmen Bratosin (TU/e), *Grid Architecture for Distributed Process Mining*

13 Xiaoyu Mao (UvT), *Airport under Control. Multiagent Scheduling for Airport Ground Handling*

14 Milan Lovric (EUR), *Behavioral Finance and Agent-Based Artificial Markets*

15 Marijn Koolen (UvA), *The Meaning of Structure: the Value of Link Evidence for Information Retrieval*

16 Maarten Schadd (UM), *Selective Search in Games of Different Complexity*

17 Jiyin He (UvA), *Exploring Topic Structure: Coherence, Diversity and Relatedness*

18 Mark Ponsen (UM), *Strategic Decision-Making in complex games*

19 Ellen Rusman (OU), *The Mind's Eye on Personal Profiles*

20 Qing Gu (VU), *Guiding service-oriented software engineering - A view-based approach*

21 Linda Terlouw (TUD), *Modularization and Specification of Service-Oriented Systems*

22 Junte Zhang (UvA), *System Evaluation of Archival Description and Access*

23 Wouter Weerkamp (UvA), *Finding People and their Utterances in Social Media*

24 Herwin van Welbergen (UT), *Behavior Generation for Interpersonal Coordination with Virtual Humans On Specifying, Scheduling and Realizing Multimodal Virtual Human Behavior*

25 Syed Waqar ul Qounain Jaffry (VU), *Analysis and Validation of Models for Trust Dynamics*

26 Matthijs Aart Pontier (VU), *Virtual Agents for Human Communication - Emotion Regulation and Involvement-Distance Trade-Offs in Embodied Conversational Agents and Robots*

27 Aniel Bhulai (VU), *Dynamic website optimization through autonomous management of design patterns*

28 Rianne Kaptein (UvA), *Effective Focused Retrieval by Exploiting Query Context and Document Structure*

29 Faisal Kamiran (TU/e), *Discrimination-aware Classification*

30 Egon van den Broek (UT), *Affective Signal Processing (ASP): Unraveling the mystery of emotions*

31 Ludo Waltman (EUR), *Computational and Game-Theoretic Approaches for Modeling Bounded Rationality*

32 Nees-Jan van Eck (EUR), *Methodological Advances in Bibliometric Mapping of Science*

33 Tom van der Weide (UU), *Arguing to Motivate Decisions*

34 Paolo Turrini (UU), *Strategic Reasoning in Interdependence: Logical and Game-theoretical Investigations*

35 Maaike Harbers (UU), *Explaining Agent Behavior in Virtual Training*

36 Erik van der Spek (UU), *Experiments in serious game design: a cognitive approach*

37 Adriana Burlutiu (RUN), *Machine Learning for Pairwise Data, Applications for Preference Learning and Supervised Network Inference*

38 Nyree Lemmens (UM), *Bee-inspired Distributed Optimization*

39 Joost Westra (UU), *Organizing Adaptation using Agents in Serious Games*

40 Viktor Clerc (VU), *Architectural Knowledge Management in Global Software Development*

41 Luan Ibraimi (UT), *Cryptographically Enforced Distributed Data Access Control*

42 Michal Sindlar (UU), *Explaining Behavior through Mental State Attribution*

43 Henk van der Schuur (UU), *Process Improvement through Software Operation Knowledge*

44 Boris Reuderink (UT), *Robust Brain-Computer Interfaces*

45 Herman Stehouwer (UvT), *Statistical Language Models for Alternative Sequence Selection*

46 Beibei Hu (TUD), *Towards Contextualized Information Delivery: A Rule-based Architecture for the Domain of Mobile Police Work*

47 Azizi Bin Ab Aziz (VU), *Exploring Computational Models for Intelligent Support of Persons with Depression*

48 Mark Ter Maat (UT), *Response Selection and Turn-taking for a Sensitive Artificial Listening Agent*

49 Andreea Niculescu (UT), *Conversational interfaces for task-oriented spoken dialogues: design aspects influencing interaction quality*

2012

01 Terry Kakeeto (UvT), *Relationship Marketing for SMEs in Uganda*

02 Muhammad Umair (VU), *Adaptivity, emotion, and Rationality in Human and Ambient Agent Models*

03 Adam Vanya (VU), *Supporting Architecture Evolution by Mining Software Repositories*

04 Jurriaan Souer (UU), *Development of Content Management System-based Web Applications*

05 Marijn Plomp (UU), *Maturing Interorganisational Information Systems*

06 Wolfgang Reinhardt (OU), *Awareness Support for Knowledge Workers in Research Networks*

07 Rianne van Lambalgen (VU), *When the Going Gets Tough: Exploring Agent-based Models of Human Performance under Demanding Conditions*

08 Gerben de Vries (UvA), *Kernel Methods for Vessel Trajectories*

09 Ricardo Neisse (UT), *Trust and Privacy Management Support for Context-Aware Service Platforms*

10 David Smits (TU/e), *Towards a Generic Distributed Adaptive Hypermedia Environment*

11 J.C.B. Rantham Prabhakara (TU/e), *Process Mining in the Large: Preprocessing, Discovery, and Diagnostics*

12 Kees van der Sluijs (TU/e), *Model Driven Design and Data Integration in Semantic Web Information Systems*

13 Suleman Shahid (UvT), *Fun and Face: Exploring non-verbal expressions of emotion during playful interactions*

14 Evgeny Knutov (TU/e), *Generic Adaptation Framework for Unifying Adaptive Web-based Systems*

15 Natalie van der Wal (VU), *Social Agents. Agent-Based Modelling of Integrated Internal and Social Dynamics of Cognitive and Affective Processes.*

16 Fiemke Both (VU), *Helping people by understanding them - Ambient Agents supporting task execution and depression treatment*

17 Amal Elgammal (UvT), *Towards a Comprehensive Framework for Business Process Compliance*

18 Eltjo Poort (VU), *Improving Solution Architecting Practices*

19 Helen Schonenberg (TU/e), *What's Next? Operational Support for Business Process Execution*

20 Ali Bahramisharif (RUN), *Covert Visual Spatial Attention, a Robust Paradigm for Brain-Computer Interfacing*

21 Roberto Cornacchia (TUD), *Querying Sparse Matrices for Information Retrieval*

22 Thijs Vis (UvT), *Intelligence, politie en veiligheidsdienst: verenigbare grootheden?*

23 Christian Muehl (UT), *Toward Affective Brain-Computer Interfaces: Exploring the Neurophysiology of Affect during Human Media Interaction*

24 Laurens van der Werff (UT), *Evaluation of Noisy Transcripts for Spoken Document Retrieval*

25 Silja Eckartz (UT), *Managing the Business Case Development in Inter-Organizational IT Projects: A Methodology and its Application*

26 Emile de Maat (UvA), *Making Sense of Legal Text*

27 Hayrettin Gurkok (UT), *Mind the Sheep! User Experience Evaluation & Brain-Computer Interface Games*

28 Nancy Pascall (UvT), *Engendering Technology Empowering Women*

29 Almer Tigelaar (UT), *Peer-to-Peer Information Retrieval*

30 Alina Pommeranz (TUD), *Designing Human-Centered Systems for Reflective Decision Making*
31 Emily Bagarukayo (RUN), *A Learning by Construction Approach for Higher Order Cognitive Skills Improvement, Building Capacity and Infrastructure*
32 Wietske Visser (TUD), *Qualitative multi-criteria preference representation and reasoning*
33 Rory Sie (OU), *Coalitions in Cooperation Networks (COCOON)*
34 Pavol Jancura (RUN), *Evolutionary analysis in PPI networks and applications*
35 Evert Haasdijk (VU), *Never Too Old To Learn – On-line Evolution of Controllers in Swarm- and Modular Robotics*
36 Denis Ssebugwawo (RUN), *Analysis and Evaluation of Collaborative Modeling Processes*
37 Agnes Nakakawa (RUN), *A Collaboration Process for Enterprise Architecture Creation*
38 Selmar Smit (VU), *Parameter Tuning and Scientific Testing in Evolutionary Algorithms*
39 Hassan Fatemi (UT), *Risk-aware design of value and coordination networks*
40 Agus Gunawan (UvT), *Information Access for SMEs in Indonesia*
41 Sebastian Kelle (OU), *Game Design Patterns for Learning*
42 Dominique Verpoorten (OU), *Reflection Amplifiers in self-regulated Learning*
43 *Withdrawn*
44 Anna Tordai (VU), *On Combining Alignment Techniques*
45 Benedikt Kratz (UvT), *A Model and Language for Business-aware Transactions*
46 Simon Carter (UvA), *Exploration and Exploitation of Multilingual Data for Statistical Machine Translation*
47 Manos Tsagkias (UvA), *Mining Social Media: Tracking Content and Predicting Behavior*
48 Jorn Bakker (TU/e), *Handling Abrupt Changes in Evolving Time-series Data*
49 Michael Kaisers (UM), *Learning against Learning - Evolutionary dynamics of reinforcement learning algorithms in strategic interactions*
50 Steven van Kervel (TUD), *Ontologogy driven Enterprise Information Systems Engineering*
51 Jeroen de Jong (TUD), *Heuristics in Dynamic Sceduling; a practical framework with a case study in elevator dispatching*

2013

01 Viorel Milea (EUR), *News Analytics for Financial Decision Support*
02 Erietta Liarou (CWI), *MonetDB/DataCell: Leveraging the Column-store Database Technology for Efficient and Scalable Stream Processing*
03 Szymon Klarman (VU), *Reasoning with Contexts in Description Logics*
04 Chetan Yadati (TUD), *Coordinating autonomous planning and scheduling*
05 Dulce Pumareja (UT), *Groupware Requirements Evolutions Patterns*
06 Romulo Goncalves (CWI), *The Data Cyclotron: Juggling Data and Queries for a Data Warehouse Audience*
07 Giel van Lankveld (UvT), *Quantifying Individual Player Differences*
08 Robbert-Jan Merk (VU), *Making enemies: cognitive modeling for opponent agents in fighter pilot simulators*
09 Fabio Gori (RUN), *Metagenomic Data Analysis: Computational Methods and Applications*
10 Jeewanie Jayasinghe Arachchige (UvT), *A Unified Modeling Framework for Service Design.*
11 Evangelos Pournaras (TUD), *Multi-level Reconfigurable Self-organization in Overlay Services*
12 Marian Razavian (VU), *Knowledge-driven Migration to Services*
13 Mohammad Safiri (UT), *Service Tailoring: User-centric creation of integrated IT-based homecare services to support independent living of elderly*
14 Jafar Tanha (UvA), *Ensemble Approaches to Semi-Supervised Learning Learning*
15 Daniel Hennes (UM), *Multiagent Learning - Dynamic Games and Applications*
16 Eric Kok (UU), *Exploring the practical benefits of argumentation in multi-agent deliberation*
17 Koen Kok (VU), *The PowerMatcher: Smart Coordination for the Smart Electricity Grid*
18 Jeroen Janssens (UvT), *Outlier Selection and One-Class Classification*
19 Renze Steenhuizen (TUD), *Coordinated Multi-Agent Planning and Scheduling*
20 Katja Hofmann (UvA), *Fast and Reliable Online Learning to Rank for Information Retrieval*
21 Sander Wubben (UvT), *Text-to-text generation by monolingual machine translation*
22 Tom Claassen (RUN), *Causal Discovery and Logic*
23 Patricio de Alencar Silva (UvT), *Value Activity Monitoring*
24 Haitham Bou Ammar (UM), *Automated Transfer in Reinforcement Learning*

25 Agnieszka Anna Latoszek-Berendsen (UM), *Intention-based Decision Support. A new way of representing and implementing clinical guidelines in a Decision Support System*
26 Alireza Zarghami (UT), *Architectural Support for Dynamic Homecare Service Provisioning*
27 Mohammad Huq (UT), *Inference-based Framework Managing Data Provenance*
28 Frans van der Sluis (UT), *When Complexity becomes Interesting: An Inquiry into the Information eXperience*
29 Iwan de Kok (UT), *Listening Heads*
30 Joyce Nakatumba (TU/e), *Resource-Aware Business Process Management: Analysis and Support*
31 Dinh Khoa Nguyen (UvT), *Blueprint Model and Language for Engineering Cloud Applications*
32 Kamakshi Rajagopal (OU), *Networking For Learning; The role of Networking in a Lifelong Learner's Professional Development*
33 Qi Gao (TUD), *User Modeling and Personalization in the Microblogging Sphere*
34 Kien Tjin-Kam-Jet (UT), *Distributed Deep Web Search*
35 Abdallah El Ali (UvA), *Minimal Mobile Human Computer Interaction*
36 Than Lam Hoang (TU/e), *Pattern Mining in Data Streams*
37 Dirk Börner (OU), *Ambient Learning Displays*
38 Eelco den Heijer (VU), *Autonomous Evolutionary Art*
39 Joop de Jong (TUD), *A Method for Enterprise Ontology based Design of Enterprise Information Systems*
40 Pim Nijssen (UM), *Monte-Carlo Tree Search for Multi-Player Games*
41 Jochem Liem (UvA), *Supporting the Conceptual Modelling of Dynamic Systems: A Knowledge Engineering Perspective on Qualitative Reasoning*
42 Léon Planken (TUD), *Algorithms for Simple Temporal Reasoning*
43 Marc Bron (UvA), *Exploration and Contextualization through Interaction and Concepts*

2014

01 Nicola Barile (UU), *Studies in Learning Monotone Models from Data*
02 Fiona Tuliyano (RUN), *Combining System Dynamics with a Domain Modeling Method*
03 Sergio Raul Duarte Torres (UT), *Information Retrieval for Children: Search Behavior and Solutions*
04 Hanna Jochmann-Mannak (UT), *Websites for children: search strategies and interface design - Three studies on children's search performance and evaluation*
05 Jurriaan van Reijsen (UU), *Knowledge Perspectives on Advancing Dynamic Capability*
06 Damian Tamburri (VU), *Supporting Networked Software Development*
07 Arya Adriansyah (TU/e), *Aligning Observed and Modeled Behavior*
08 Samur Araujo (TUD), *Data Integration over Distributed and Heterogeneous Data Endpoints*
09 Philip Jackson (UvT), *Toward Human-Level Artificial Intelligence: Representation and Computation of Meaning in Natural Language*
10 Ivan Salvador Razo Zapata (VU), *Service Value Networks*
11 Janneke van der Zwaan (TUD), *An Empathic Virtual Buddy for Social Support*
12 Willem van Willigen (VU), *Look Ma, No Hands: Aspects of Autonomous Vehicle Control*
13 Arlette van Wissen (VU), *Agent-Based Support for Behavior Change: Models and Applications in Health and Safety Domains*
14 Yangyang Shi (TUD), *Language Models With Meta-information*
15 Natalya Mogles (VU), *Agent-Based Analysis and Support of Human Functioning in Complex Socio-Technical Systems: Applications in Safety and Healthcare*
16 Krystyna Milian (VU), *Supporting trial recruitment and design by automatically interpreting eligibility criteria*
17 Kathrin Dentler (VU), *Computing healthcare quality indicators automatically: Secondary Use of Patient Data and Semantic Interoperability*
18 Mattijs Ghijsen (UvA), *Methods and Models for the Design and Study of Dynamic Agent Organizations*
19 Vinicius Ramos (TU/e), *Adaptive Hypermedia Courses: Qualitative and Quantitative Evaluation and Tool Support*
20 Mena Habib (UT), *Named Entity Extraction and Disambiguation for Informal Text: The Missing Link*
21 Kassidy Clark (TUD), *Negotiation and Monitoring in Open Environments*
22 Marieke Peeters (UU), *Personalized Educational Games - Developing agent-supported scenario-based training*
23 Eleftherios Sidirourgos (UvA/CWI), *Space Efficient Indexes for the Big Data Era*
24 Davide Ceolin (VU), *Trusting Semi-structured Web Data*

25 Martijn Lappenschaar (RUN), *New network models for the analysis of disease interaction*
26 Tim Baarslag (TUD), *What to Bid and When to Stop*
27 Rui Jorge Almeida (EUR), *Conditional Density Models Integrating Fuzzy and Probabilistic Representations of Uncertainty*
28 Anna Chmielowiec (VU), *Decentralized k-Clique Matching*
29 Jaap Kabbedijk (UU), *Variability in Multi-Tenant Enterprise Software*
30 Peter de Cock (UvT), *Anticipating Criminal Behaviour*
31 Leo van Moergestel (UU), *Agent Technology in Agile Multiparallel Manufacturing and Product Support*
32 Naser Ayat (UvA), *On Entity Resolution in Probabilistic Data*
33 Tesfa Tegegne (RUN), *Service Discovery in eHealth*
34 Christina Manteli (VU), *The Effect of Governance in Global Software Development: Analyzing Transactive Memory Systems.*
35 Joost van Ooijen (UU), *Cognitive Agents in Virtual Worlds: A Middleware Design Approach*
36 Joos Buijs (TU/e), *Flexible Evolutionary Algorithms for Mining Structured Process Models*
37 Maral Dadvar (UT), *Experts and Machines United Against Cyberbullying*
38 Danny Plass-Oude Bos (UT), *Making brain-computer interfaces better: improving usability through post-processing.*
39 Jasmina Maric (UvT), *Web Communities, Immigration, and Social Capital*
40 Walter Omona (RUN), *A Framework for Knowledge Management Using ICT in Higher Education*
41 Frederic Hogenboom (EUR), *Automated Detection of Financial Events in News Text*
42 Carsten Eijckhof (CWI/TUD), *Contextual Multidimensional Relevance Models*
43 Kevin Vlaanderen (UU), *Supporting Process Improvement using Method Increments*
44 Paulien Meesters (UvT), *Intelligent Blauw. Met als ondertitel: Intelligence-gestuurde politiezorg in gebiedsgebonden eenheden.*
45 Birgit Schmitz (OU), *Mobile Games for Learning: A Pattern-Based Approach*
46 Ke Tao (TUD), *Social Web Data Analytics: Relevance, Redundancy, Diversity*
47 Shangsong Liang (UvA), *Fusion and Diversification in Information Retrieval*

2015

01 Niels Netten (UvA), *Machine Learning for Relevance of Information in Crisis Response*
02 Faiza Bukhsh (UvT), *Smart auditing: Innovative Compliance Checking in Customs Controls*
03 Twan van Laarhoven (RUN), *Machine learning for network data*
04 Howard Spoelstra (OU), *Collaborations in Open Learning Environments*
05 Christoph Bösch (UT), *Cryptographically Enforced Search Pattern Hiding*
06 Farideh Heidari (TUD), *Business Process Quality Computation - Computing Non-Functional Requirements to Improve Business Processes*
07 Maria-Hendrike Peetz (UvA), *Time-Aware Online Reputation Analysis*
08 Jie Jiang (TUD), *Organizational Compliance: An agent-based model for designing and evaluating organizational interactions*
09 Randy Klaassen (UT), *HCI Perspectives on Behavior Change Support Systems*
10 Henry Hermans (OU), *OpenU: design of an integrated system to support lifelong learning*
11 Yongming Luo (TU/e), *Designing algorithms for big graph datasets: A study of computing bisimulation and joins*
12 Julie M. Birkholz (VU), *Modi Operandi of Social Network Dynamics: The Effect of Context on Scientific Collaboration Networks*
13 Giuseppe Procaccianti (VU), *Energy-Efficient Software*
14 Bart van Straalen (UT), *A cognitive approach to modeling bad news conversations*
15 Klaas Andries de Graaf (VU), *Ontology-based Software Architecture Documentation*
16 Changyun Wei (UT), *Cognitive Coordination for Cooperative Multi-Robot Teamwork*
17 André van Cleeff (UT), *Physical and Digital Security Mechanisms: Properties, Combinations and Trade-offs*
18 Holger Pirk (CWI), *Waste Not, Want Not! - Managing Relational Data in Asymmetric Memories*
19 Bernardo Tabuenca (OU), *Ubiquitous Technology for Lifelong Learners*
20 Lois Vanhée (UU), *Using Culture and Values to Support Flexible Coordination*
21 Sibren Fetter (OU), *Using Peer-Support to Expand and Stabilize Online Learning*
22 Zhemin Zhu (UT), *Co-occurrence Rate Networks*

23 Luit Gazendam (VU), *Cataloguer Support in Cultural Heritage*
24 Richard Berendsen (UvA), *Finding People, Papers, and Posts: Vertical Search Algorithms and Evaluation*
25 Steven Woudenberg (UU), *Bayesian Tools for Early Disease Detection*
26 Alexander Hogenboom (EUR), *Sentiment Analysis of Text Guided by Semantics and Structure*
27 Sándor Héman (CWI), *Updating compressed colomn stores*
28 Janet Bagorogoza (TiU), *Knowledge Management and High Performance; The Uganda Financial Institutions Model for HPO*
29 Hendrik Baier (UM), *Monte-Carlo Tree Search Enhancements for One-Player and Two-Player Domains*
30 Kiavash Bahreini (OU), *Real-time Multimodal Emotion Recognition in E-Learning*
31 Yakup Koç (TUD), *On the robustness of Power Grids*
32 Jerome Gard (UL), *Corporate Venture Management in SMEs*
33 Frederik Schadd (TUD), *Ontology Mapping with Auxiliary Resources*
34 Victor de Graaf (UT), *Gesocial Recommender Systems*
35 Jungxao Xu (TUD), *Affective Body Language of Humanoid Robots: Perception and Effects in Human Robot Interaction*

2016

01 Syed Saiden Abbas (RUN), *Recognition of Shapes by Humans and Machines*
02 Michiel Christiaan Meulendijk (UU), *Optimizing medication reviews through decision support: prescribing a better pill to swallow*
03 Maya Sappelli (RUN), *Knowledge Work in Context: User Centered Knowledge Worker Support*
04 Laurens Rietveld (VU), *Publishing and Consuming Linked Data*
05 Evgeny Sherkhonov (UvA), *Expanded Acyclic Queries: Containment and an Application in Explaining Missing Answers*
06 Michel Wilson (TUD), *Robust scheduling in an uncertain environment*
07 Jeroen de Man (VU), *Measuring and modeling negative emotions for virtual training*
08 Matje van de Camp (TiU), *A Link to the Past: Constructing Historical Social Networks from Unstructured Data*
09 Archana Nottamkandath (VU), *Trusting Crowdsourced Information on Cultural Artefacts*
10 George Karafotias (VU), *Parameter Control for Evolutionary Algorithms*
11 Anne Schuth (UvA), *Search Engines that Learn from Their Users*
12 Max Knobbout (UU), *Logics for Modelling and Verifying Normative Multi-Agent Systems*
13 Nana Baah Gyan (VU), *The Web, Speech Technologies and Rural Development in West Africa - An ICT4D Approach*
14 Ravi Khadka (UU), *Revisiting Legacy Software System Modernization*
15 Steffen Michels (RUN), *Hybrid Probabilistic Logics - Theoretical Aspects, Algorithms and Experiments*
16 Guangliang Li (UvA), *Socially Intelligent Autonomous Agents that Learn from Human Reward*
17 Berend Weel (VU), *Towards Embodied Evolution of Robot Organisms*
18 Albert Meroño Peñuela (VU), *Refining Statistical Data on the Web*
19 Julia Efremova (Tu/e), *Mining Social Structures from Genealogical Data*
20 Daan Odijk (UvA), *Context & Semantics in News & Web Search*
21 Alejandro Moreno Célleri (UT), *From Traditional to Interactive Playspaces: Automatic Analysis of Player Behavior in the Interactive Tag Playground*
22 Grace Lewis (VU), *Software Architecture Strategies for Cyber-Foraging Systems*
23 Fei Cai (UvA), *Query Auto Completion in Information Retrieval*
24 Brend Wanders (UT), *Repurposing and Probabilistic Integration of Data; An Iterative and data model independent approach*
25 Julia Kiseleva (TU/e), *Using Contextual Information to Understand Searching and Browsing Behavior*
26 Dilhan Thilakarathne (VU), *In or Out of Control: Exploring Computational Models to Study the Role of Human Awareness and Control in Behavioural Choices, with Applications in Aviation and Energy Management Domains*
27 Wen Li (TUD), *Understanding Geo-spatial Information on Social Media*
28 Mingxin Zhang (TUD), *Large-scale Agent-based Social Simulation - A study on epidemic prediction and control*
29 Nicolas Höning (TUD), *Peak reduction in decentralised electricity systems - Markets and prices for flexible planning*
30 Ruud Mattheij (UvT), *The Eyes Have It*
31 Mohammad Khelghati (UT), *Deep web content monitoring*
32 Eelco Vriezekolk (UT), *Assessing Telecommunication Service Availability Risks for Crisis Organisations*

33 Peter Bloem (UvA), *Single Sample Statistics, exercises in learning from just one example*
34 Dennis Schunselaar (TU/e), *Configurable Process Trees: Elicitation, Analysis, and Enactment*
35 Zhaochun Ren (UvA), *Monitoring Social Media: Summarization, Classification and Recommendation*
36 Daphne Karreman (UT), *Beyond R2D2: The design of nonverbal interaction behavior optimized for robot-specific morphologies*
37 Giovanni Sileno (UvA), *Aligning Law and Action - a conceptual and computational inquiry*
38 Andrea Minuto (UT), *Materials that Matter - Smart Materials meet Art & Interaction Design*
39 Merijn Bruijnes (UT), *Believable Suspect Agents; Response and Interpersonal Style Selection for an Artificial Suspect*
40 Christian Detweiler (TUD), *Accounting for Values in Design*
41 Thomas King (TUD), *Governing Governance: A Formal Framework for Analysing Institutional Design and Enactment Governance*
42 Spyros Martzoukos (UvA), *Combinatorial and Compositional Aspects of Bilingual Aligned Corpora*
43 Saskia Koldijk (RUN), *Context-Aware Support for Stress Self-Management: From Theory to Practice*
44 Thibault Sellam (UvA), *Automatic Assistants for Database Exploration*
45 Bram van de Laar (UT), *Experiencing Brain-Computer Interface Control*
46 Jorge Gallego Perez (UT), *Robots to Make you Happy*
47 Christina Weber (UL), *Real-time foresight - Preparedness for dynamic innovation networks*
48 Tanja Buttler (TUD), *Collecting Lessons Learned*
49 Gleb Polevoy (TUD), *Participation and Interaction in Projects. A Game-Theoretic Analysis*
50 Yan Wang (UvT), *The Bridge of Dreams: Towards a Method for Operational Performance Alignment in IT-enabled Service Supply Chains*

2017

01 Jan-Jaap Oerlemans (UL), *Investigating Cybercrime*
02 Sjoerd Timmer (UU), *Designing and Understanding Forensic Bayesian Networks using Argumentation*
03 Daniël Harold Telgen (UU), *Grid Manufacturing; A Cyber-Physical Approach with Autonomous Products and Reconfigurable Manufacturing Machines*
04 Mrunal Gawade (CWI), *Multi-core Parallelism in a Column-store*
05 Mahdieh Shadi (UvA), *Collaboration Behavior*
06 Damir Vandic (EUR), *Intelligent Information Systems for Web Product Search*
07 Roel Bertens (UU), *Insight in Information: from Abstract to Anomaly*
08 Rob Konijn (VU), *Detecting Interesting Differences:Data Mining in Health Insurance Data using Outlier Detection and Subgroup Discovery*
09 Dong Nguyen (UT), *Text as Social and Cultural Data: A Computational Perspective on Variation in Text*
10 Robby van Delden (UT), *(Steering) Interactive Play Behavior*
11 Florian Kunneman (RUN), *Modelling patterns of time and emotion in Twitter #anticipointment*
12 Sander Leemans (TU/e), *Robust Process Mining with Guarantees*
13 Gijs Huisman (UT), *Social Touch Technology - Extending the reach of social touch through haptic technology*
14 Shoshannah Tekofsky (UvT), *You Are Who You Play You Are: Modelling Player Traits from Video Game Behavior*
15 Peter Berck (RUN), *Memory-Based Text Correction*
16 Aleksandr Chuklin (UvA), *Understanding and Modeling Users of Modern Search Engines*
17 Daniel Dimov (UL), *Crowdsourced Online Dispute Resolution*
18 Ridho Reinanda (UvA), *Entity Associations for Search*
19 Jeroen Vuurens (UT), *Proximity of Terms, Texts and Semantic Vectors in Information Retrieval*
20 Mohammadbashir Sedighi (TUD), *Fostering Engagement in Knowledge Sharing: The Role of Perceived Benefits, Costs and Visibility*
21 Jeroen Linssen (UT), *Meta Matters in Interactive Storytelling and Serious Gaming (A Play on Worlds)*
22 Sara Magliacane (VU), *Logics for causal inference under uncertainty*
23 David Graus (UvA), *Entities of Interest — Discovery in Digital Traces*
24 Chang Wang (TUD), *Use of Affordances for Efficient Robot Learning*
25 Veruska Zamborlini (VU), *Knowledge Representation for Clinical Guidelines, with applications to Multimorbidity Analysis and Literature Search*
26 Merel Jung (UT), *Socially intelligent robots that understand and respond to human touch*

27 Michiel Joosse (UT), *Investigating Positioning and Gaze Behaviors of Social Robots: People's Preferences, Perceptions and Behaviors*

28 John Klein (VU), *Architecture Practices for Complex Contexts*

29 Adel Alhuraibi (UvT), *From IT-Business Strategic Alignment to Performance: A Moderated Mediation Model of Social Innovation, and Enterprise Governance of IT*

30 Wilma Latuny (UvT), *The Power of Facial Expressions*

31 Ben Ruijl (UL), *Advances in computational methods for QFT calculations*

32 Thaer Samar (RUN), *Access to and Retrievability of Content in Web Archives*

33 Brigit van Loggem (OU), *Towards a Design Rationale for Software Documentation: A Model of Computer-Mediated Activity*

34 Maren Scheffel (OU), *The Evaluation Framework for Learning Analytics*

35 Martine de Vos (VU), *Interpreting natural science spreadsheets*

36 Yuanhao Guo (UL), *Shape Analysis for Phenotype Characterisation from High-throughput Imaging*

37 Alejandro Montes Garcia (TU/e), *WiBAF: A Within Browser Adaptation Framework that Enables Control over Privacy*

38 Alex Kayal (TUD), *Normative Social Applications*

39 Sara Ahmadi (RUN), *Exploiting properties of the human auditory system and compressive sensing methods to increase noise robustness in ASR*

40 Altaf Hussain Abro (VU), *Steer your Mind: Computational Exploration of Human Control in Relation to Emotions, Desires and Social Support For applications in human-aware support systems*

41 Adnan Manzoor (VU), *Minding a Healthy Lifestyle: An Exploration of Mental Processes and a Smart Environment to Provide Support for a Healthy Lifestyle*

42 Elena Sokolova (RUN), *Causal discovery from mixed and missing data with applications on ADHD datasets*

43 Maaike de Boer (RUN), *Semantic Mapping in Video Retrieval*

44 Garm Lucassen (UU), *Understanding User Stories - Computational Linguistics in Agile Requirements Engineering*

45 Bas Testerink (UU), *Decentralized Runtime Norm Enforcement*

46 Jan Schneider (OU), *Sensor-based Learning Support*

47 Jie Yang (TUD), *Crowd Knowledge Creation Acceleration*

48 Angel Suarez (OU), *Collaborative inquiry-based learning*

2018

01 Han van der Aa (VU), *Comparing and Aligning Process Representations*

02 Felix Mannhardt (TU/e), *Multi-perspective Process Mining*

03 Steven Bosems (UT), *Causal Models For Well-Being: Knowledge Modeling, Model-Driven Development of Context-Aware Applications, and Behavior Prediction*

04 Jordan Janeiro (TUD), *Flexible Coordination Support for Diagnosis Teams in Data-Centric Engineering Tasks*

05 Hugo Huurdeman (UvA), *Supporting the Complex Dynamics of the Information Seeking Process*

06 Dan Ionita (UT), *Model-Driven Information Security Risk Assessment of Socio-Technical Systems*

07 Jieting Luo (UU), *A formal account of opportunism in multi-agent systems*

08 Rick Smetsers (RUN), *Advances in Model Learning for Software Systems*

09 Xu Xie (TUD), *Data Assimilation in Discrete Event Simulations*

10 Julienka Mollee (VU), *Moving forward: supporting physical activity behavior change through intelligent technology*

11 Mahdi Sargolzaei (UvA), *Enabling Framework for Service-oriented Collaborative Networks*

12 Xixi Lu (TU/e), *Using behavioral context in process mining*

13 Seyed Amin Tabatabaei (VU), *Computing a Sustainable Future*

14 Bart Joosten (UvT), *Detecting Social Signals with Spatiotemporal Gabor Filters*

15 Naser Davarzani (UM), *Biomarker discovery in heart failure*

16 Jaebok Kim (UT), *Automatic recognition of engagement and emotion in a group of children*

17 Jianpeng Zhang (TU/e), *On Graph Sample Clustering*

18 Henriette Nakad (UL), *De Notaris en Private Rechtspraak*

19 Minh Duc Pham (VU), *Emergent relational schemas for RDF*

20 Manxia Liu (RUN), *Time and Bayesian Networks*

21 Aad Slootmaker (OU), *EMERGO: a generic platform for authoring and playing scenario-based serious games*

22 Eric Fernandes de Mello Araujo (VU), *Contagious: Modeling the Spread of Behaviours, Perceptions and Emotions in Social*

Networks

23 Kim Schouten (EUR), *Semantics-driven Aspect-Based Sentiment Analysis*

24 Jered Vroon (UT), *Responsive Social Positioning Behaviour for Semi-Autonomous Telepresence Robots*

25 Riste Gligorov (VU), *Serious Games in Audio-Visual Collections*

26 Roelof Anne Jelle de Vries (UT), *Theory-Based and Tailor-Made: Motivational Messages for Behavior Change Technology*

27 Maikel Leemans (TU/e), *Hierarchical Process Mining for Scalable Software Analysis*

28 Christian Willemse (UT), *Social Touch Technologies: How they feel and how they make you feel*

29 Yu Gu (UvT), *Emotion Recognition from Mandarin Speech*

30 Wouter Beek (VU), *The 'K' in 'semantic web' stands for 'knowledge': scaling semantics to the web*

2019

01 Rob van Eijk (UL), *Comparing and Aligning Process Representations*

02 Emmanuelle Beauxis Aussalet (CWI, UU), *Statistics and Visualizations for Assessing Class Size Uncertainty*

03 Eduardo Gonzalez Lopez de Murillas (TU/e), *Process Mining on Databases: Extracting Event Data from Real Life Data Sources*

04 Ridho Rahmadi (RUN), *Finding stable causal structures from clinical data*

05 Sebastiaan van Zelst (TU/e), *Process Mining with Streaming Data*

06 Chris Dijkshoorn (VU), *Niche sourcing for Improving Access to Linked Cultural Heritage Datasets*

07 Soude Fazeli (OU), *Recommender Systems in Social Learning Platforms*

08 Frits de Nijs (TUD), *Resource-constrained Multi-agent Markov Decision Processes*

Acknowledgments

This thesis would not have been possible without the support of a great many people. In the first place, I am grateful for the daily supervision of Mathijs de Weerdt and Matthijs Spaan, for their insightful comments and continued strong support, even when progress was slow, and for always putting the interests of their students first. I also want to thank Frans Campfens from Alliander, for providing me with an overview of the challenges in the upcoming energy transition, and a pragmatic perspective on what a smart grid should, and should not be. Finally, I want to thank Tomas Klos, whose supportive supervision during my M.Sc. project guided me towards doing a Ph.D. in the first place (thereby also giving me the chance to correct my mistake of not adding acknowledgments to my M.Sc. thesis).

My research was performed in the Algorithmics group, which over the years has become like a second home. Thus, I want to thank Cees Witteveen, for being my promotor initially, and for leading a welcoming and successful research group. When starting out I was very fortunate to share an office with Shruti, as she patiently answered my many questions to get me up to speed that much quicker. During the years I also greatly enjoyed 'sparring' with Erwin, who always offered insightful discussions, resulting in a successful research collaboration and a much sharpened mind. I should also mention Rens here, the group's only certified Ph.D. candidate; I valued your interesting perspectives on the promovendus and optimization situations. A strong tradition of the group is the daily lunch routine, and the lunch discussions have benefited greatly from the interesting opinions and questions of Gleb and Peter (Novák). To Adriaan, Bob, Hans, Joris, Léon, Longjian, Michel, Peter (Bosman), Shémara, and Simon, thank you all, for the gezelligheid, and for making my time in Algorithmics that much more enjoyable. Finally, I am happy to note that the group continues to grow stronger; to the superstars Anna, Canmanie, Greg, Kim, Koos, Lei, Natalia, Neil, and Thiago, who joined Algorithmics during my last year (give or take), best of luck!

Near the end of my term I was fortunate enough to get the cool opportunity to do an internship at Adobe Research. For that chance I want to thank Nikos Vlassis, who also supported my supervision there with sharp questions leading to new insights. The daily supervision of Georgios Theocharous further ensured the success of my internship, by providing me with both a strong vision and practical tips along the way. I also want to thank the other researchers and interns there for the discussions and experiences, in particular Ishan Gupta, for sharing his enthusiasm and warm attitude. I was also lucky to work with and co-supervise Stefan Hugtenburg while at Delft, whose insights on developing code and teaching, and diligent work ethic calibrated my views on what makes a good student.

Because obtaining a doctorate is such a long-term project, one has to pace oneself with regular breaks; in my case, this meant frequent visits to the Lu Gia Jen dojo, where I greatly enjoyed training under the guidance of Ed Abrahams. It was fun pushing myself in the supportive atmosphere, with special thanks to regulars Jeroen, Daniel and René. Oss! Playing the electric guitar was another wish of mine, and fortunately my teacher Nick Papakostas supported me in this while being very understanding of my sometimes overfull schedule. And while I started participating only quite recently, I want to thank Otto Visser for organizing the gaming nights, which helped to blow off some steam while writing up.

Of course, I was also very fortunate to have friends and family who supported me immensely, and who continue to believe in me even when I myself would sometimes not. To my close friends, thanks for all the countless hours we spent together over the many years, whether face-to-face or on-line. Patrick, your enthusiasm and strong spirit continues to inspire me. Wouter, thank you for keeping me down to earth, and looking forward to reading your thesis. I owe a huge debt to my parents, as they made sure through their support that I never needed to worry about anything other than doing my job as best I could. My brothers Peter and Erik, thank you for always being interested and always being around to keep me company. And finally, to the love of my life, Leimin, you give me purpose and meaning, and I am very lucky to have found you. Your unconditional support made it possible for me to finish this thesis.