Capacity-aware Sequential Recommendations

de Nijs, Frits; Theocharous, Georgios; Vlassis, Nikos; de Weerdt, Mathijs M.; Spaan, Matthijs T.J.

Publication date
2018

Document Version
Final published version

Published in
Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems

Capacity-aware Sequential Recommendations

Frits de Nijs
Delft University of Technology
Delft, the Netherlands
f.denijs@tudelft.nl

Georgios Theocharous
Adobe Systems
San Jose, California, United States
theochar@adobe.com

Nikos Vlassis
Netflix
Los Gatos, California, United States
nvlassis@netflix.com

Mathijs M. de Weerdt
Delft University of Technology
Delft, the Netherlands
m.m.deweerdt@tudelft.nl

Matthijs T. J. Spaan
Delft University of Technology
Delft, the Netherlands
m.t.j.spaan@tudelft.nl

ABSTRACT

Personalized recommendations are increasingly important to engage users and guide them through large systems, for example when recommending points of interest to tourists visiting a popular city. To maximize long-term user experience, the system should consider issuing recommendations sequentially, since by observing the user's response to a recommendation, the system can update its estimate of the user's (latent) interests. However, as traditional recommender systems target individuals, their effect on a collective of users can unintentionally overload capacity. Therefore, recommender systems should not only consider the users' interests, but also the effect of recommendations on the available capacity.

The structure in such a constrained, multi-agent, partially observable decision problem can be exploited by a novel belief-space sampling algorithm which bounds the size of the state space by a limit on regret. By exploiting the stationary structure of the problem, our algorithm is significantly more scalable than existing approximate solvers. Moreover, by explicitly considering the information value of actions, this algorithm significantly improves the quality of recommendations over an extension of posterior sampling reinforcement learning to the constrained multi-agent case. We show how to decouple constraint satisfaction from sequential recommendation policies, resulting in algorithms which issue recommendations to thousands of agents while respecting constraints.

ACM Reference Format:
Frits de Nijs, Georgios Theocharous, Nikos Vlassis, Mathijs M. de Weerdt, and Matthijs T. J. Spaan. 2018. Capacity-aware Sequential Recommendations. In *Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), Stockholm, Sweden, July 10–15, 2018,* IFAAMAS, 9 pages.

1 INTRODUCTION

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 [3]. In many cases, recommendations should also take into account relations between objects and the history of the user, which requires the system to consider long-term effects of a recommendation. For example, when recommending news articles to readers, the user's history informs their familiarity with a topic and thereby the value of a contextual article over a latest update. Sequentiality is also important when recommending points-of-interest to tourists, to avoid backtracking over their past route.

One of the primary challenges for a recommender system 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 [33]. Unfortunately, these models cannot anticipate the effect of a *sequence* of recommendations on the user [30]. To plan for long-term gains, we should instead cast the problem as a reinforcement learning problem, where we attempt to learn the dynamics of a Markov Decision Process (MDP) over time [35].

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 [7]. However, capacity constraints on recommended items may also serve an operational purpose: in virtual items such as news articles, limiting recommendations for naturally popular items can promote recommendation diversity.

Sharing resources is especially challenging in a system where multiple *learning* agents interact, because the trade-off between exploration and exploitation couples across agents: should an uncertain agent be awarded the resource in order to learn, or should another agent be allowed to use it to obtain reward with high certainty? However, recommendations provide the potential to steer users around constrained points, motivating the need for capacity-aware sequential recommendations. In this paper we investigate how recommender systems should learn when they are constrained by resource limits restricting their joint actions.

It is critical for a recommender system to identify the true interests of a user in as few recommendations as possible, as mistakes risk losing the user's attention. However, general reinforcement learning algorithms have a high sample complexity, requiring long interaction periods before a good policy is obtained [19]. An optimal learning policy prescribes actions which ensure that the entire learning trajectory is optimal [21]. Computing an optimal learning policy for a general reinforcement learning problem amounts to solving a continuous-state, Partially Observable MDP (POMDP; [12]). Unfortunately, these models can only be practically solved using approximate algorithms [2, 26]. Therefore, in this work we make the simplifying assumption that we can model differences between

Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. André, S. Koenig (eds.), July 10–15, 2018, Stockholm, Sweden.

users through *parametric* MDPs [11] with a finite parameter space, corresponding to a finite number of *user types*.

Work by Guez et al. [15] suggests two approaches to arrive at an optimal learning policy: (i) On-line sparse sampling algorithms such as Posterior Sampling Reinforcement Learning (PSRL; [34]), 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. [6] in casting the parametric MDP to a stationary Mixed-Observable MDP (MOMDP; [23]). 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 [25], limiting its practical scalability.

To address these challenges, we propose two novel algorithms: the first algorithm is an extension of PSRL to the multi-agent, constrained setting, by combining it with a Column Generation technique which has proven effective at decoupling agents from global constraints [10, 40]. The second algorithm exploits the structural properties of the recommendation problem to approximately solve the MOMDP: by computing the worst-case regret of switching from a recommendation strategy over a belief over types to an optimal strategy for a given type, we can bound the size of the state space.

We evaluate our algorithms on a large-scale tourist recommendation domain based on real data from visitors to the city of Melbourne. Both our approaches are significantly more scalable than a state-of-the-art approximate MOMDP solver; constrained PSRL is shown to find high-quality capacity-aware individual recommendations in seconds. Our bounded-regret algorithm finds near-optimal constrained policies even in the more challenging setting of recommending multiple options to users.

2 PROBLEM DESCRIPTION: MULTI-AGENT CONSTRAINED LEARNING

In this section we present the constrained multi-agent learning problem formally. We start with background material on the Parametric MDP model, used to represent the single-agent dynamics, and the Constrained MDP model for modeling the capacity limits. Then, these models are combined in our problem description.

2.1 Parametric MDPs

A finite-horizon MDP [4] is defined by tuple $\langle S, A, T, R, h\rangle$. It consists of the finite sets of states $s \in S$ and actions $a \in A$, a transition function T and reward function R defined over these sets, and finite horizon h. Every time step t, the decision maker chooses an action a, resulting in a stochastic transition from state s to subsequent state s', according to the probability given by transition function $T(s, a, s') = \mathrm{P}(s' \mid s, a)$. The chosen action results in an instantaneous reward given by the reward function $R : S \times A \to \mathbb{R}$.

In a Parametric MDP [11] one or both functions additionally depend on structural parameters. Let Θ stand for a continuous 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 parametrized functions $\bar{R}(\theta, s, a)$, and $\bar{T}(\theta, s, a, s')$. Fixing parameter θ instantiates a parametrized MDP_θ, having $\langle S, A, R_\theta, T_\theta, h\rangle$ with functions $R_\theta(s, a) = \bar{R}(\theta, s, a)$ and $T_\theta(s, a, s') = \bar{T}(\theta, s, a, s')$.

The behavior of a decision maker is prescribed by its policy $\pi(t, s)$, mapping each time $t \in \{1, \ldots, h\}$ and state $s \in S$ to an action a. The value function $V_{\theta,\pi}[t, s]$ gives the expected value of following policy π starting from the given state and time. The objective of a planner is to compute the policy which obtains the maximum expected value over the entire horizon. An optimal *unconstrained* policy π^* can be computed efficiently through an application of dynamic programming: the Bellman equation computes the value maximizing action in each state recursively, by determining the value at time t based on the value function at $t + 1$,

$$V_{\theta,\pi^*}[h, s] = \max_{a \in A} R_\theta(s, a),$$
$$V_{\theta,\pi^*}[t, s] = \max_{a \in A}\Big(R_\theta(s, a) + \sum_{s' \in S}\Big(T_\theta(s, a, s')V_{\theta,\pi^*}[t + 1, s']\Big)\Big). \quad (1)$$

2.2 Constrained MDPs and Column Generation

A Constrained MDP [1] augments the objective function of the MDP planning problem with a number of linear constraints. We consider constraints modeled through a consumption function C and limit function L defined over m resource types. The consumption of resource type r is defined using function $C_r : S \times A \to [0, c_{\max,r}]$, where $c_{\max,r}$ denotes the maximum potential consumption of resource type r. The limit function $L(r)$ gives the maximum permitted instantaneous use of resource r. The optimal constrained policy satisfies the constraints *in expectation*, meaning that it optimizes

$$\max_\pi \mathrm{E}\big[V_{\theta,\pi}\big], \text{ subject to } \mathrm{E}\big[C_{\theta,\pi,t,r}\big] \le L(r) \quad \forall t, \forall r. \quad (2)$$

Computing such a constrained policy involves optimizing a Linear Program (LP). Especially when solving large, factored models such as multi-agent problems, directly optimizing the resulting LP is typically infeasible. For such models Column Generation (CG; [13]) has proven to be an effective algorithm [10, 40]. Column Generation allows for decomposing combinatorial optimization problems, provided the problem has some method to generate new potential solutions efficiently. The technique uses the insight that, when an LP is used to select solutions from an exhaustive set, the simplex algorithm iteratively adds solutions to the selected set which are not 'priced out' by the λ prices computed in the dual solution. A solution is priced out if its contribution to the objective per unit of the constraint is less than λ. If we can generate the optimal solution to be selected on the fly, we avoid having to maintain the exhaustive set of solutions explicitly. Generating the solution comes down to optimizing an ancillary problem subject to the λ costs.

Yost and Washburn [41] identified that this technique can be applied when solving constrained POMDPs, by augmenting the optimality criterion of the planning problem with a term corresponding to the expected resource consumption cost $\mathrm{E}[C_{\pi,r}]$, i.e.,

$$\arg\max_\pi \Big(\mathrm{E}[V_\pi] - \sum_{t,r} \lambda_{t,r}\mathrm{E}[C_{\pi,t,r}]\Big). \quad (3)$$

This routine is used to compute a new policy to be added to the set of potential policies Z, which forms the search space of the LP. The optimal mix of policies subject to constraints is then selected by

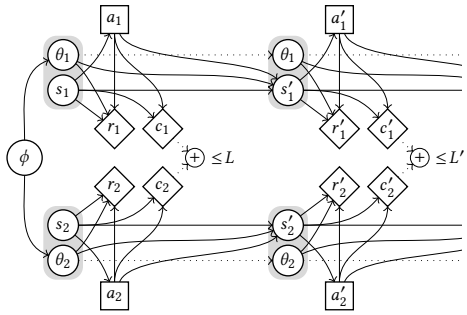

Figure 1: The DBN of a multi-agent constrained learning problem having two agents and one constraint.

solving the following LP:

$$\begin{aligned} \max_{x_j} \quad & \sum_{\pi_j \in Z} x_j \, \mathrm{E}[V_{\pi_j}], \\ \text{s.t.} \quad & \sum_{\pi_j \in Z} x_j \, \mathrm{E}[C_{\pi_j,t,r}] \leq L(r) \quad \forall r, \forall t, \\ & \sum_{\pi_j \in Z} x_j = 1, \text{ and } x_j \geq 0 \qquad \forall j. \end{aligned} \tag{4}$$

2.3 Multi-agent constrained learning problem

Thus far, we have assumed the instantiation parameter θ to be known to the decision maker. However, usually these parameters are hidden; in our model, parameter θ encodes the users' latent interests. We consider a multi-agent recommender system consisting of n agents, their models characterized by a single parametric MDP. Each agent i behaves according to the MDP instantiated from its type θ_i. 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. The result is a constrained, multi-agent parametric MDP having tuple $\langle n, \phi, \Theta, S, A, \bar{R}, \bar{T}, h, C, L \rangle$. Figure 1 presents the interactions between two agents and one constraint graphically through their Dynamic Bayesian Networks (DBN; [5]). Nodes in the figure represent states and observations (circles), decisions (squares), and costs and rewards (diamonds). Solid edges represent stochastic influences, while the dotted edges indicate deterministic influence, capturing the fact that an agent's type θ is stationary.

3 MULTI-AGENT CONSTRAINED PSRL

Column Generation is an effective algorithm for constrained multi-agent MDPs when they are weakly coupled [10]. At the same time, PSRL is an effective heuristic 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.

3.1 Posterior sampling reinforcement learning

The algorithm operates as follows: to identify the true parameters $\hat{\theta}$ of an instantiated MDP, the algorithm iteratively refines a probability density over parameter space Θ, through application of Bayes' Theorem on the likelihood of the observed state. The Thompson sampling heuristic [39] is used to select actions, by optimistically assuming that type θ_j sampled from the current belief over types b is the true type. The optimal policy for the assumed model π_j is used to select actions for an episode of τ steps, during which the belief over Θ is updated with every observed transition to state s'_i.

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 [24], as well as the guarantee of finding the optimal policy in a logarithmic number of time steps with high probability in our on-line (non-episodic) setting [14]. The Thompson sampling heuristic has also proven effective in recommender systems, with applications in ad format selection [36] and contextual recommenders [16].

Algorithm 1 Multi-agent constrained PSRL.

Given prior $\phi = \mathrm{P}(\theta_j)$, epoch length τ, 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 $\langle x, Z \rangle = \textsc{colGen}(\mathrm{MDP}_{\theta_j}, n, \phi)$
2: **for** episode $k = 1 \rightarrow \lceil \frac{h}{\tau} \rceil$ **do**
3: sample $\forall i\colon \theta_i \sim b_i$
4: sample joint $\vec{\pi}$ by $\pi_i \sim \langle x_{\theta_i}, Z_{\theta_i} \rangle$
5: **for** timestep $l = 1 \rightarrow \tau$ **and** $t \leq h$ **do**
6: select joint action $\vec{a} = \vec{\pi}(t, \vec{s})$
7: observe next state $\forall i\colon s'_i \sim \mathrm{P}(\cdot \mid \hat{\theta}_i, s_i, a_i)$ ▷ Agent par. $\hat{\theta}_i$
8: update b_i by Bayes' rule, $\forall i\colon \mathrm{P}(b'_i \mid s_i, a_i, s'_i, b_i)$
9: $\vec{s} \leftarrow \vec{s}\,', \vec{b} \leftarrow \vec{b}', t \leftarrow t + 1$
10: **end for**
11: **end for**

3.2 Combining Column Generation and PSRL

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 1 presents the proposed approach. Column generation is called on line 1 to compute the optimal mix of resource-satisfying policies over the *expected* number of agents of each type. Because our agents behave according to homogeneous types, agents of the same type can be added together [41]. Therefore, the master LP is

$$\begin{aligned} \max_{x_{i,j}} \quad & \sum_{i=1}^{|\Theta|} \sum_{\pi_j \in Z_i} x_{i,j} \, \mathrm{E}[V_{\theta_i, \pi_j}(s_1)], \\ \text{s.t.} \quad & \sum_{i=1}^{|\Theta|} \sum_{\pi_j \in Z_i} x_{i,j} \, \mathrm{E}[C_{\theta_i, \pi_j, r}(t, s_1)] \leq L(r), \quad \forall r, \forall t, \\ & \sum_{\pi_j \in Z_i} x_{i,j} = n\mathrm{P}(\theta_i) \;\; \forall i, \text{ and } x_{i,j} \geq 0, \qquad \forall i, \forall j. \end{aligned} \tag{5}$$

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.

4 A MOMDP APPROACH TO SEQUENTIAL RECOMMENDATIONS

Because PSRL uses the Thompson sampling heuristic to choose policies, the trajectory leading up to convergence may use sub-optimal actions resulting in unexpected resource violations. In order to control the consumption at all times, we need to compute an optimal learning policy, which amounts to solving a constrained Mixed-Observable MDP (MOMDP; [23]). Solving a general MOMDP model to optimality is a hard problem. However our models are built out of a parametric MDP, which enables exploiting its 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 the resulting policy will be used in Column Generation to satisfy the constraints, we need to take special care that the expected values computed by this algorithm remain correct for these approximate solutions, which we address in the following section.

4.1 Optimal learning of Parametric MDPs

Although PSRL eventually converges to the optimal policy, its trajectory leading up to convergence may be sub-optimal as a result of using a heuristic. For example, if there exists an action which is not part of the optimal policy for any MDP_θ, this action will never be chosen by PSRL. This is the case even if this action immediately reveals the true parameters of the MDP. In order to reason about such information gathering actions, a learning algorithm should explicitly consider the decision-theoretic value of information [17].

To our knowledge, Silver [31, Ch. 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. [6] extend the scope to our setting of identifying the true model of a hidden-model MDP, consisting of a set of candidate MDPs each with their own transition and reward function. In order to leverage existing algorithms to compute an optimal policy for hidden-model MDPs, the authors convert the problem to a MOMDP.

The state space of a MOMDP model factors into a fully observable factor $x \in X$ and a partially observable factor $y \in Y$, each with their own transition functions, $T_X(x' \mid x, y, a)$ and $T_Y(y' \mid x, y, a, x')$. As

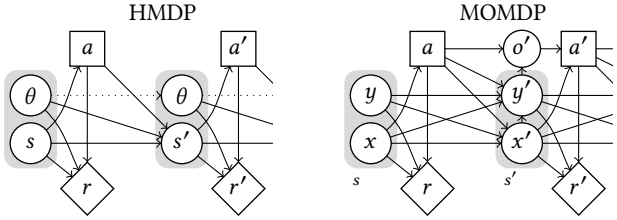

Figure 2: Comparison of HMDP and MOMDP models.

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 policy $\pi(t, x, o)$ on the observable factor x. Given a finite parametric MDP $\langle \Theta, S, A, \bar{R}, \bar{T}, h \rangle$, we derive an equivalent stationary MOMDP $\langle X, Y, A, O, T_X, T_Y, R, \Omega, h \rangle$ having elements

$$\begin{aligned} X &= S, & T_X(s' \mid s, \theta, a) &= T_\theta(s' \mid s, a), \\ Y &= \Theta, & R(s, \theta, a) &= R_\theta(s, a), \\ O &= \{o_{\text{NULL}}\}, & \Omega(o_{\text{NULL}} \mid a, \theta') &= 1, \end{aligned} \tag{6}$$

$$T_Y(\theta' \mid s, \theta, a, s') = \begin{cases} 1 & \text{if } \theta = \theta', \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

Figure 2 presents the two models graphically, through their dynamic Bayesian networks. The dotted edge in the HMDP model captures the notion of stationarity in the type given by equation (7). Although the HMDP appears to be a much less general model, Chadès et al. [6] prove that computing an optimal policy for HMDPs falls in the same PSPACE complexity class as POMDPs [25].

Casting parametric MDPs to MOMDPs has the advantage that existing theory and algorithms can be leveraged. Most algorithms for POMDPs and MOMDPs make use of a celebrated result by Sondik [32] that the optimal value function is piecewise linear convex, and can be represented by a set of α-vectors, each giving expected values associated with taking an action a in belief b. Martin et al. [22] exploit the stationary property of the MOMDP to compute a set of α-vectors which form a lower bound on the optimal value function. They propose to compute the optimal MDP policy for each type, and subsequently apply each optimal policy to all other types to construct an α-vector per policy. Initializing existing solvers with this lower bound speeds up their convergence, by providing tighter bounds for pruning computed vectors.

Unfortunately, existing solvers for MOMDPs typically assume the discounted infinite-horizon case, which 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 complexity of solving a MOMDP necessitates computing approximate solutions, however approximate α-vector based solvers return expected values which do not correspond with the true expected value of the policy. This is problematic because we need true expectations for the integration with Column Generation.

4.2 Computing exact expectations for a reduced belief space

To avoid the drawbacks of existing solvers, we propose a new algorithm for stationary MOMDPs based on explicitly reasoning about reachable belief states. A belief state b records a probability distribution over the possible (unobserved) states S, with $b(s)$ indicating how likely the agent expects to be in state s [18]. 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 (1), although the tractability of this approach is limited by the exponential growth of B in the horizon h. 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 a (suboptimal) policy's 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 values 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 from every reachable 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_{θ_i}, then $V^*[\langle t, s, b\rangle] = V_{\theta_i, \pi_i^*}[t, s]$. 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_{θ_j}, resulting in value $V_{\theta_j, \pi_i^*}[t, s]$. 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

$$Q\big[\langle t, s, b\rangle, \pi_i^*\big] = \sum_{j=1}^{|\Theta|} \Big(b(\theta_j) \cdot V_{\theta_j, \pi_i^*}[t, s] \Big). \tag{8}$$

The optimal value of using a fixed policy in point $\langle t, s, b\rangle$ is then

$$\bar{V}\big[\langle t, s, b\rangle\big] = \max_{\pi} Q\big[\langle t, s, b\rangle, \pi\big]. \tag{9}$$

While the expected value $\bar{V}\big[\langle t, s, b\rangle\big]$ is a lower bound on the optimal 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 $\langle t, s, b\rangle \notin B'$.

Algorithm 2 Bounded belief state space planning.

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

In principle we could compute $\bar{V}\big[\langle t, s, b\rangle\big]$ exactly, 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,

$$\bar{\bar{V}}\big[\langle t, s, b\rangle\big] = \max_{\theta_i \in \Theta} Q\big[\langle t, s, b\rangle, \pi_i^*\big]. \tag{10}$$

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 2 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 for every other type θ_i. 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 12. In case of a missing point b', the best policy π_j^* is selected on line 13, and the expected value of using this policy is computed according to the belief state.

The resulting policy returned on line 24 consists of two stages. For every belief point b in the collection B', the maximally valued action stored in $\pi[b]$ on line 19 is selected. However, in case a $b' \notin B'$ is reached during execution, the policy π_j^* stored on line 13

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 [18], the expected value at any such 'missing' belief state is also exact. Therefore, the values computed for the prior $b_0 = \langle 1, s_1, \phi \rangle$ are the true expectations of the (potentially suboptimal) values obtained by executing the policy computed by Algorithm 2. Therefore, this algorithm avoids all three 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.

4.3 Using expected regret to bound the belief state space

To determine an approximate belief space B' for Algorithm 2, we use the *expected regret* of switching to a fixed MDP policy as a 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_{θ_j, π_i^*}, for which by definition of optimality $V_{\theta_j, \pi_i^*} \leq V_{\theta_j, \pi_j^*}$. Thus, the use of policy π_i^* incurs a regret of

$$\textsc{regret}(\langle t, s, b \rangle, i) = \sum_{j=1}^{|\Theta|} \left(b(\theta_j) \cdot \left(V_{\theta_j, \pi_j^*}[t, s] - V_{\theta_j, \pi_i^*}[t, s] \right) \right). \quad (11)$$

At a given belief point $\langle t, s, b \rangle$, the optimal MDP policy for type i found in (10) minimizes this regret, therefore

$$\textsc{regret}(\langle t, s, b \rangle) = \min_i \left(\textsc{regret}(\langle t, s, b \rangle, i) \right). \quad (12)$$

Because the MDP policies are computed over the entire horizon, regret is also defined for the prior b_0. 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 et al. [18]. 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 $\text{P}(b)$ 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:

$$\textsc{regret}(b) > \left(e^{-\alpha(\text{P}(b)-p)} - e^{-\alpha(1-p)} \right) \cdot \textsc{regret}(b_0). \quad (13)$$

Threshold (13) is based on an exponential decay function over probability $\text{P}(b)$ which attains 0 at $\text{P}(b) = 1$ and approximately $\textsc{regret}(b_0)$ at $\text{P}(b) = p$.

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, derived from a dataset[1] of photograph meta-data from tourists visiting the city [38]. 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 $\text{P}_\theta(l \mid s_k)$.

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, conditional on a variable-length, bounded history of previously observed symbols [29]. 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:

$$\begin{aligned} T_\theta(s_i \mid s_0) &= \text{PST}_\theta(l_i \mid \langle\rangle) && \forall l_i \in P, \\ T_\theta(s_{i,j} \mid s_i) &= \text{PST}_\theta(l_j \mid \langle l_i \rangle) && \forall l_j \in P, \\ T_\theta(s_{j,k} \mid s_{i,j}) &= \text{PST}_\theta(l_k \mid \langle l_i, l_j \rangle) && \forall l_k \in P. \end{aligned} \quad (14)$$

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 (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

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

receiving a recommendation, because no such recommendation system is in place yet to observe the effect of recommendations on users. We follow Theocharous et al. [37] 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

$$
\begin{aligned}
T_\theta(s' \mid s, a_0) &= T_\theta(s' \mid s) \\
T_\theta(s' \mid s, a_i) &= \begin{cases} T_\theta(s' \mid s, a_0)^{\frac{1}{\mu(\theta)}} & \text{if } l_i \text{ selected in } s' \\ T_\theta(s' \mid s, a_0)/z & \text{otherwise} \end{cases} \\
T_\theta(s' \mid s, a_{i,j}) &= \begin{cases} T_\theta(s' \mid s, a_i) & \text{if } R_\theta(l_i) \geq R_\theta(l_j) \\ T_\theta(s' \mid s, a_j) & \text{otherwise} \end{cases}
\end{aligned} \tag{15}
$$

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:

$$
\begin{aligned}
\rho(s_h, a_i) &= \begin{cases} \sigma(0) \max_j R_\theta(l_j) & \text{if } i \in h \\ 0 & \text{otherwise} \end{cases} \\
R_\theta(s_{\ldots,j}, a_0) &= 0 \\
R_\theta(s_{\ldots,j}, a_i) &= \sigma(\mathbb{I}(a_i)) R_\theta(l_i) - \rho(s_{\ldots,j}, a_i) \\
R_\theta(s_{\ldots,j}, a_{i,k}) &= \frac{R_\theta(s_{\ldots,j}, a_i) + R_\theta(s_{\ldots,j}, a_k)}{2}
\end{aligned} \tag{16}
$$

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.

6 EXPERIMENTAL EVALUATION

In this section we empirically evaluate our proposed algorithms on the tourist location recommendation problem. Our objective is to assess the scalability and solution quality of our proposed algorithms. Therefore, we compare our algorithms against state-of-the-art approximate MOMDP planner SARSOP [20]; 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, 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.

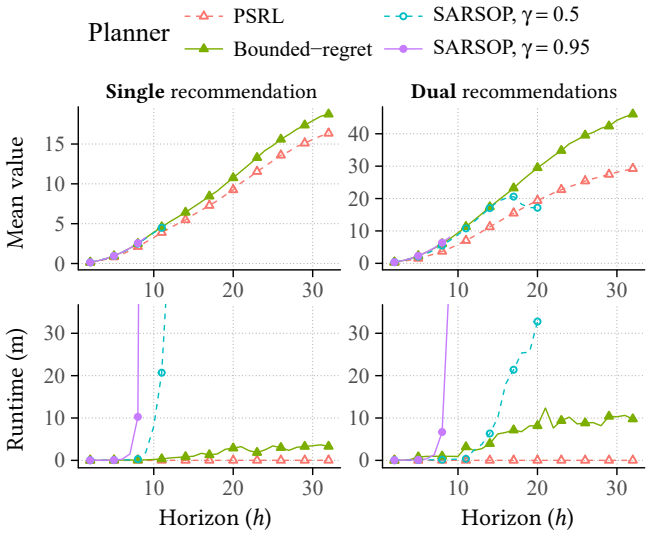

Figure 3: Solution quality and runtime of the capacity-aware recommendation planners, as a function of the horizon.

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 policy as the mean over 1,000 simulations per trial, solving 5 instances per setting. The computation time is measured by mean elapsed wall-clock time per setting, with a 30 minute timeout. Based on preliminary experiments, we set the regret bounding parameters to $\alpha = 500$ and $p = 0.005$, which resulted in a good trade-off between state-space size and eventual bounding of growth.

Figure 3 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 per agent, 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

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

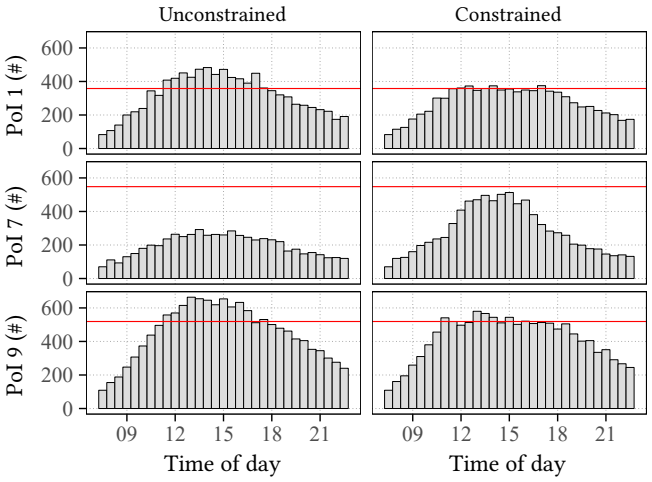

Figure 4: Effect of applying constrained recommendations on number of agents visiting points of interest (PoI).

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 a large-scale problem. Figure 4 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.

7 RELATED WORK

Zhang et al. [42] 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. Their model uncertainty 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 knowledge states, resulting in an exponential complexity in the number of knowledge states.

Our model combining the possible agent types is a constrained POMDP. Poupart et al. [27] propose an algorithm to solve constrained POMDPs directly, by casting the problem as an (approximate) constrained belief state MDP. The framework of Constrained MDPs [1] consists of a single large LP which combines the satisfaction of constraints with computing the policy. Such a solution maintains a variable for each ⟨belief state, action⟩-pair, which quickly grows intractably large. Although their method permits approximate belief spaces, our decoupling into MDP policies could not be applied in their method, because there the satisfaction of constraints is coupled with the computation of the policy. Recently, Walraven and Spaan [40] proposed a novel approximate algorithm for constrained POMDPs on the basis of Column Generation, which solves the expected-value problem by converting α-vector policies to policy graphs. This algorithm is directly applicable to our domain, however 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.

8 CONCLUSIONS AND FUTURE WORK

Recommender systems should use sequential interactions with agents to optimally refine their knowledge about the user, and should plan recommendations which satisfy the user's long-term interests. Many times, the items being recommended are also subject to capacity limitations; in this work we present two novel algorithms for computing capacity-aware sequential recommendations for large-scale recommendation problems, resulting in the following contributions:

(i) We integrate PSRL with Column Generation to obtain an efficient heuristic constrained learning algorithm (Section 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. We show that these solutions can embedded in Column Generation to compute a constrained optimal learning policy for our model (Section 4).
(iii) We use the expected regret to propose an efficient belief space truncating condition, which results in a highly scalable approximation algorithm for stationary MOMDPs (Section 4.3).
(iv) We show how to construct a constrained multi-agent recommender system from passive data, having recommendation actions that allow an alternative (Section 5).

We demonstrate that constrained PSRL finds high-quality policies quickly when the problem considered does not exhibit information gathering actions. However, in case a model does allow information gathering, such when issuing recommendations with alternatives, our results show that we are better off casting the problem as a constrained optimal learning problem. This problem can be solved tractably by planning a MOMDP over a reduced space of beliefs, derived from the regret of switching to an MDP policy.

In future work, we want to investigate methods to ensure fairness between users under constraints. Currently, column generation may decide to structurally give one type of user lower quality recommendations, in order to satisfy the constraints. In this case, we need to consider the recommendation problem as a multi-objective decision problem [28], to compute a set of policies trading off resource consumption with expected value. Another avenue of future work considers how to recommend users when only a subset of them uses the recommender system. In this case the behavior of the uncontrolled users, under stochastic influences such as the weather, impacts the capacity constraint stochastically [9]. This is especially challenging when we can not measure crowds reliably, making the available capacity itself a partially observable quantity. Finally, we intend to evaluate our algorithms on other constrained (multi-agent) learning domains, such as adaptive management problems [22] and smart-grid applications [8].

REFERENCES

[1] Eitan Altman. 1999. *Constrained Markov Decision Processes*. Chapman & Hall/CRC.

[2] Haoyu Bai, David Hsu, Wee Sun Lee, and Vien A. Ngo. 2011. *Monte Carlo value iteration for continuous-state POMDPs*. STAR, Vol. 68. Springer, Heidelberg, 175–191.

[3] Robert Bell, Yehuda Koren, and Chris Volinsky. 2007. Chasing $1,000,000: how we won the Netflix progress prize. *Statistical Computer & Graphics* 12, 2 (Dec. 2007), 4–12.

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

[5] Craig Boutilier, Thomas Dean, and Steve Hanks. 1999. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research* 11 (1999), 1–94.

[6] Iadine Chadès, 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*. 267–273.

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

[8] 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*. 615–621.

[9] 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*.

[10] 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*. 3562–3568.

[11] Richard Dearden, Nir Friedman, and David Andre. 1999. Model Based Bayesian Exploration. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*. 150–159.

[12] Michael O'Gordon Duff. 2002. *Optimal learning: Computational procedures for Bayes-adaptive Markov decision processes*. Ph.D. Dissertation. University of Massachusetts Amherst.

[13] P. C. Gilmore and R. E. Gomory. 1961. A Linear Programming Approach to the Cutting-Stock Problem. *Operations Research* 9, 6 (1961), 849–859.

[14] Aditya Gopalan and Shie Mannor. 2015. Thompson Sampling for Learning Parameterized Markov Decision Processes. In *Proceedings of The 28th Conference on Learning Theory (Proceedings of Machine Learning Research)*, Peter Grünwald, Elad Hazan, and Satyen Kale (Eds.), Vol. 40. PMLR, Paris, France, 861–898.

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

[16] Negar Hariri, Bamshad Mobasher, and Robin Burke. 2014. Context adaptation in interactive recommender systems. In *Proceedings of the 8th ACM Conference on Recommender systems*. 41–48.

[17] Ronald A. Howard. 1966. Information value theory. *IEEE Transactions on Systems Science and Cybernetics* 2, 1 (8 1966), 22–26.

[18] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101, 1-2 (May 1998), 99–134.

[19] Sham Machandranath Kakade. 2003. *On the Sample Complexity of Reinforcement Learning*. Ph.D. Dissertation. University College London.

[20] Hanna Kurniawati, 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.

[21] James John Martin. 1967. *Bayesian decision problems and Markov chains*. Wiley, New York.

[22] Péron Martin, 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*. 4531–4537.

[23] Sylvie C. W. Ong, Shao Wei Png, David Hsu, and Wee Sun Lee. 2010. Planning under Uncertainty for Robotic Tasks with Mixed Observability. *The International Journal of Robotics Research* 29, 8 (2010), 1053–1068.

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

[25] Christos H. Papadimitriou and John N. Tsitsiklis. 1987. The Complexity of Markov Decision Processes. *Mathematics of Operations Research* 12, 3 (1987), 441–450.

[26] Josep M. Porta, Nikos Vlassis, Matthijs T. J. Spaan, and Pascal Poupart. 2006. Point-based value iteration for continuous POMDPs. *Journal of Machine Learning Research* 7 (11 2006), 2329–2367.

[27] Pascal Poupart, 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*. 3342–3348.

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

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

[30] Guy Shani, David Heckerman, and Ronen I. Brafman. 2005. An MDP-Based Recommender System. *Journal of Machine Learning Research* 6 (2005), 1265–1295.

[31] Edward Allan Silver. 1963. *Markovian decision processes with uncertain transition probabilities or rewards*. Ph.D. Dissertation. Massachusetts Institute of Technology.

[32] Edward J. Sondik. 1971. *The Optimal Control of Partially Observable Markov Processes*. Ph.D. Dissertation. Stanford University.

[33] Harald Steck. 2013. Evaluation of Recommendations: Rating-prediction and Ranking. In *Proceedings of the 7th ACM Conference on Recommender Systems (RecSys '13)*. ACM, New York, NY, USA, 213–220.

[34] Malcolm J. A. Strens. 2000. A Bayesian Framework for Reinforcement Learning. In *Proceedings of the 17th International Conference on Machine Learning*. 943–950.

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

[36] Liang Tang, Romer Rosales, Ajit Singh, and Deepak Agarwal. 2013. Automatic ad format selection via contextual bandits. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*. 1587–1594.

[37] Georgios Theocharous, 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)*. ACM, New York, NY, USA, 49–52.

[38] Bart Thomee, 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. *Commun. ACM* 59, 2 (2016), 64–73.

[39] William R. Thompson. 1933. On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples. *Biometrika* 25, 3/4 (1933), 285–294.

[40] Erwin Walraven and Matthijs T. J. Spaan. 2018. Column Generation Algorithms for Constrained POMDPs. *Journal of Artificial Intelligence Research* (2018).

[41] Kirk A. Yost and Alan R. Washburn. 2000. The LP/POMDP Marriage: Optimization with Imperfect Information. *Naval Research Logistics* 47, 8 (2000), 607–619.

[42] Qi Zhang, 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*, Laura Barbulescu, Jeremy Frank, Mausam, and Stephen F. Smith (Eds.).