On Querying for Safe Optimality in Factored Markov Decision Processes

Extended Abstract

Shun Zhang
Computer Science and Engineering
University of Michigan
shunzh@umich.edu

Edmund H. Durfee
Computer Science and Engineering
University of Michigan
durfee@umich.edu

Satinder Singh
Computer Science and Engineering
University of Michigan
baveja@umich.edu

ABSTRACT

As it achieves a goal on behalf of its human user, an autonomous agent's actions may have side effects that change features of its environment in ways that negatively surprise its user. An agent that can be trusted to operate safely should thus only change features the user has explicitly permitted. We formalize this problem, and develop a planning algorithm that avoids potentially negative side effects given what the agent knows about (un)changeable features. Further, we formulate a provably minimax-regret querying strategy for the agent to selectively ask the user about features that it hasn't explicitly been told about. We empirically show how much faster it is than a more exhaustive approach and how much better its queries are than those found by the best known heuristic.

KEYWORDS

Human-robot/agent interaction; Single and multi-agent planning and scheduling

ACM Reference Format:
Shun Zhang, Edmund H. Durfee, and Satinder Singh. 2018. On Querying for Safe Optimality in Factored Markov Decision Processes. In *Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), Stockholm, Sweden, July 10–15, 2018,* IFAAMAS, 3 pages.

1 INTRODUCTION

We consider a setting where a human user tasks a computational agent with achieving a goal to change some state features of the world (e.g., a housecleaning agent should change the state of the floors and kitchen sink from dirty to clean). While accomplishing the goal, the agent also changes other features (e.g., its own position and power level, opening doors, moving furniture, scaring the cat). Some of these side-effects might be expected by the user (e.g., moving), but others may be unexpected/unsafe (e.g., leaving doors open lets the cat escape) even though they may speed goal achievement. Although the user tells the agent about some features that can be changed, as well as some to not change (e.g., don't knock over the priceless vase), the user often lacks the time, patience, or foresight to articulate the changeability of every pertinent feature, and may incorrectly assume that the agent has commonsense (e.g., about cat behavior and the value of vases).

How can the agent execute a **safely**-optimal policy in such a setting? We conservatively assume that, to ensure safety, the agent should never side-effect a feature unless changing it is explicitly known to be fine. Hence, the agent could simply execute the best policy that leaves such features unchanged. However, no such policy might exist, and even if it does it might surprise the user as unnecessarily costly/inefficient. Our focus is thus on how the agent can selectively query the user about the acceptability of changing features it hasn't yet been told about. We reject simply querying about every such feature, as this would be unbearably tedious to the user, and instead put the burden on the agent to limit the number and complexity of queries. In fact, in this paper we mostly focus on finding a single query about a few features that maximally improves upon the policy while maintaining safety.

2 PROBLEM DEFINITION

We consider a simulated robot gridworld-navigation domain inspired by [4] and depicted in Figure 1. The user tasks the agent with turning off the switch as quickly as is *safely* possible. The quickest path (π_1) traverses the carpet, but this gets the carpet dirty and the agent doesn't know if that is allowed. The agent could instead enter the room through door d1 and spend time moving box b1 or b2 out of the way (π_2 or π_3 respectively), open door d2, and then go to the switch. However, boxes might contain fragile objects and should not be moved; the user knows each box's contents, but the agent doesn't. There are of course many other more circuitous paths.

We model the domain as a factored Markov Decision Process (MDP) [5]. In our example, the agent knows that the user expects it to change its location and the switch's status. The agent does not know, however, about whether it can safely change other features like the states of boxes, doors, or carpets. Formally, we define the state s to be features' values for a set of features, $\{\phi_1, \phi_2, \ldots, \phi_n\}$. The state space S is the cross-product of the values the features can take. We assume the agent knows the transition dynamics.

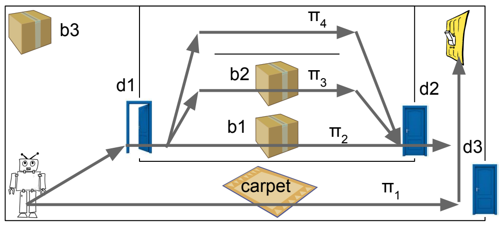

Figure 1: The robot navigation domain. The dominating policies are shown as arrows.

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. © 2018 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

For the purposes of computing safely-optimal policies, the agent partitions the features into the following sets: (1) $\Phi^{\mathbf{A}}_F$: The *free-features* (i.e., freely changeable). The agent knows that these features can be changed freely (for example, its location). (2) $\Phi^{\mathbf{A}}_L$: The *locked-features*. The agent knows it should never change any of these features. (3) $\Phi^{\mathbf{A}}_?$: The *unknown-features*. These are features that the agent doesn't (initially) know whether the user considers freely changeable or locked. The user similarly partitions features, but only into the sets $\Phi^{\mathbf{U}}_L$ and $\Phi^{\mathbf{U}}_F$. We assume that the agent's knowledge, while generally incomplete ($\Phi^{\mathbf{A}}_? \neq \emptyset$), is consistent with that of the user. That is, $\Phi^{\mathbf{A}}_F \subseteq \Phi^{\mathbf{U}}_F$ and $\Phi^{\mathbf{A}}_L \subseteq \Phi^{\mathbf{U}}_L$.
Querying: We assume that the only way for the agent to figure out that an unknown feature is in fact freely changeable is to ask the user a query about that feature. Thus, the focus of the rest of this paper is on how the agent can ask a good query about a small number, k, of features.

3 METHOD

Our solution is to first prune from $\Phi^{\mathbf{A}}_?$ features that are guaranteed to not be relevant to ask, and then to efficiently find the best (minimax-regret) k-sized subset of the relevant features to query.
Querying Relevant Unknown-Features: Intuitively, when is a feature in $\Phi^{\mathbf{A}}_?$ relevant to the agent's planning of a safely-optimal policy? In the navigation domain, if the agent plans to take the quickest path to the switch (π_1 in Figure 1), it will change the state of the carpet (from clean to dirty). The carpet feature is thus *relevant* since the agent would change it if permitted. If the carpet can't be changed but the door d2 can, the agent would follow (in order of preference) policy π_2, π_3, or π_4, so d2, b1, and b2 are relevant. Box b3 and door d3 are *irrelevant*, however, since no matter which (if any) other features are free, an optimal policy would never change them. Thus, an unknown feature is relevant when under some circumstance (some answer to some query) the agent's optimal policy would side-effect that feature. Such policies are *dominating policies*. Formally, a **dominating policy** is a safely-optimal policy for the circumstance where the unknown features $\Phi^{\mathbf{A}}_?$ are partitioned into locked and free subsets. The features changed by any dominating policy are called **relevant features**, denoted by Φ_{rel}.

Following Altman [2], we can use linear programming to find the optimal policy that does not change Φ_L or $\Phi_?$. Instead of finding the dominating policies for all exponentially (in $|\Phi^{\mathbf{A}}_?|$) many subsets $\Phi \subseteq \Phi^{\mathbf{A}}_?$, we contribute an algorithm that finds dominating policies incrementally (and in practice more efficiently) by constructing the sets of relevant features and dominating policies simultaneously. Initially, let $\Phi'_{rel} = \emptyset$. In each iteration, it examines a new subset of Φ'_{rel}, Φ, and finds the safely-optimal policy π with Φ being locked (and $\Phi'_{rel} \setminus \Phi$ being free). It then adds the features changed by π to Φ'_{rel}. It repeats this process until Φ'_{rel} stops growing and all subsets of Φ'_{rel} are examined. When it terminates, Φ'_{rel} is the set of relevant features.
Finding minimax-regret queries: The agent need only query the user about relevant features, but could further reduce the user's burden by being selective about *which* relevant features it asks. We allow the agent to pose a *k-feature query*, Φ_q ($\Phi_q \subseteq \Phi_{rel}$ and $|\Phi_q| = k$), which asks about the changeabilities of features in Φ_q.

We first define the post-response utility when the agent asks query Φ_q and $\Phi_c \subseteq \Phi^{\mathbf{A}}_?$ are actually changeable: $u(\Phi_q, \Phi_c) = \max_{\pi \in \Pi_{\Phi^{\mathbf{A}}_? \setminus (\Phi_q \cap \Phi_c)}} V^{\pi}$. This is the value of the safely-optimal policy after the user's response. We then consider the circumstance where a set of features Φ_c are changeable and under which the difference between the utilities of asking Φ_q and $\Phi_{q'}$ is maximized. We call this difference of utilities the **pairwise maximum regret** of queries Φ_q and $\Phi_{q'}$, defined in a similar way to Regan and Boutilier [15]: $PMR(\Phi_q, \Phi_{q'}) = \max_{\Phi_c \subseteq \Phi^{\mathbf{A}}_?}(u(\Phi_{q'}, \Phi_c) - u(\Phi_q, \Phi_c))$. The **maximum regret** of query Φ_q is determined by the $\Phi_{q'}$ that maximizes $PMR(\Phi_q, \Phi_{q'})$: $MR(\Phi_q) = \max_{\Phi_{q'} \subseteq \Phi_{rel}, |\Phi_{q'}|=k} PMR(\Phi_q, \Phi_{q'})$. The agent wants to ask **minimax-regret (k-feature) query**: $\Phi^{MMR}_q = \arg\min_{\Phi_q \subseteq \Phi_{rel}, |\Phi_q|=k} MR(\Phi_q)$.

Although we could compute the maximum regrets of all k-subsets of relevant features and find the one has the minimax-regret, we can avoid doing so by pruning the query space using dominance relations between queries, and by stopping search for queries when we know we have found the optimal one. More details of the algorithm and the evaluations below can be found in [18].
Empirical evaluations: We compare the following algorithms in a robot navigation domain: (1) Brute force finds all relevant features first and evaluates all k-subsets of the relevant features. (2) Our algorithm. (3) A modified heuristic from the literature (*CoA*) [16]. (4) Other baseline methods: random queries and no queries. As expected, our algorithm always finds a minimax-regret query just as the brute force algorithm does, but more efficiently than the brute force algorithm which quickly becomes computationally intractable with increased k. *CoA* is computationally even more efficient than our algorithm, but it does not always find the minimax-regret query. In particular, our algorithm's benefits increase with more opportunities to be selective (larger $\binom{|\Phi_{rel}|}{k}$). The baseline methods, though the fastest, lead to considerably worse performance than either our algorithm or *CoA* .

4 RELATED WORK & CONCLUSION

Amodei et al. [4] address the problem of avoiding negative side-effects by penalizing all side-effects while optimizing the value. We instead allow the agent to communicate with the user. In other works, safety is formulated as resolving reward uncertainty [3, 7], handling imperfectly-specified instructions [11], learning safe state distributions [9], and safe exploration [1, 8, 12]. Other works can be found in related surveys [4, 6, 10]. Minimax-regret is also used in other model uncertainty settings [13, 15]. Querying is a common approach to resolve such uncertainty [14, 17]. Our query selection strategy differs in that we deal with incomplete knowledge on feature-changeability.

In summary, we addressed the problem of an agent selectively querying a user about what features can be safely side-effected. We borrowed existing ideas from the literature about dominating policies and minimax regret, wove them together in a novel way, and injected tweaks into the resulting algorithms to improve scalability while maintaining safe optimality.
Acknowledgements: Supported in part by the US Air Force Office of Scientific Research, under grant FA9550-15-1-0039.

REFERENCES

[1] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. 2017. Constrained policy optimization. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*. 22–31.

[2] Eitan Altman. 1999. *Constrained Markov Decision Processes*. Vol. 7. CRC Press.

[3] Kareem Amin, Nan Jiang, and Satinder Singh. 2017. Repeated inverse reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*. 1813–1822.

[4] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. 2016. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565* (2016).

[5] Craig Boutilier, Richard Dearden, and Moisés Goldszmidt. 2000. Stochastic dynamic programming with factored representations. *Artificial intelligence* 121, 1 (2000), 49–107.

[6] Javier Garcıa and Fernando Fernández. 2015. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research* 16, 1 (2015), 1437–1480.

[7] Dylan Hadfield-Menell, Smitha Milli, Stuart J Russell, Pieter Abbeel, and Anca Dragan. 2017. Inverse reward design. In *Advances in Neural Information Processing Systems (NIPS)*. 6749–6758.

[8] Alexander Hans, Daniel Schneegaß, Anton Maximilian Schäfer, and Steffen Udluft. 2008. Safe exploration for reinforcement learning. In *European Symposium on Artificial Neural Networks (ESANN)*. 143–148.

[9] Michael Laskey, Sam Staszak, Wesley Yu-Shu Hsieh, Jeffrey Mahler, Florian T Pokorny, Anca D Dragan, and Ken Goldberg. 2016. SHIV: Reducing supervisor burden in DAgger using support vectors for efficient learning from demonstrations in high dimensional state spaces. In *IEEE International Conference on Robotics and Automation (ICRA)*. 462–469.

[10] Jan Leike, Miljan Martic, Victoria Krakovna, Pedro A Ortega, Tom Everitt, Andrew Lefrancq, Laurent Orseau, and Shane Legg. 2017. AI safety gridworlds. *arXiv preprint arXiv:1711.09883* (2017).

[11] Smitha Milli, Dylan Hadfield-Menell, Anca D. Dragan, and Stuart J. Russell. 2017. Should robots be obedient?. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*. 4754–4760.

[12] Teodor M. Moldovan and Pieter Abbeel. 2012. Safe exploration in Markov decision processes. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*. 1711–1718.

[13] Arnab Nilim and Laurent El Ghaoui. 2005. Robust control of Markov decision processes with uncertain transition matrices. *Operations Research* 53, 5 (2005), 780–798.

[14] Kevin Regan and Craig Boutilier. 2009. Regret-based reward elicitation for Markov decision processes. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI)*. 444–451.

[15] Kevin Regan and Craig Boutilier. 2010. Robust policy computation in reward-uncertain MDPs using nondominated policies. In *Association for the Advancement of Artificial Intelligence (AAAI)*. 1127–1133.

[16] Paolo Viappiani and Craig Boutilier. 2009. Regret-based optimal recommendation sets in conversational recommender systems. In *Proceedings of the Third ACM Conference on Recommender Systems*. 101–108.

[17] Shun Zhang, Edmund Durfee, and Satinder Singh. 2017. Approximately-optimal queries for planning in reward-uncertain Markov decision processes. In *Proceedings of the 27th International Conference on Automated Planning and Scheduling (ICAPS)*. 339–347.

[18] Shun Zhang, Edmund Durfee, and Satinder Singh. 2018. Minimax-regret querying on side effects for safe optimality in factored Markov decision processes. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*. To appear.