Comparative Criteria for Partially Observable Contingent Planning

JAAMAS Track

Dorin Shmaryahu
Ben Gurion University of the Negev
Israel

Jörg Hoffmann
Saarland University
Germany

Guy Shani
Ben Gurion University of the Negev
Israel

ABSTRACT

In contingent planning under partial observability with sensing actions, the solution can be represented as a plan tree, branching on various possible observations. Typically, one seeks a satisfying plan leading to a goal state at each leaf. In many applications, however, one may prefer some satisfying plans to others. We focus on the problem of providing valid comparative criteria for contingent plan trees and graphs, allowing us to compare two plans and decide which one is preferable. We suggest a set of such comparison criteria — plan simplicity, dominance, and best and worst plan costs. In some cases certain branches of the plan correspond to an unlikely combination of mishaps, and can be ignored, and we provide methods for pruning such unlikely branches before comparing the plan graphs. We explain these criteria, and discuss their validity, correlations, and application to real world problems. We suggest efficient algorithms for computing the comparative criteria. We provide experimental results, showing that plans computed by existing contingent planners can be compared using the suggested criteria.

KEYWORDS

Contingent Planning; Empirical Evaluation

ACM Reference Format:
Dorin Shmaryahu, Jörg Hoffmann, and Guy Shani. 2019. Comparative Criteria for Partially Observable Contingent Planning. In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019,* IFAAMAS, 3 pages.

1 INTRODUCTION

Agents operating in a partially observable environment gain important information using sensing actions. For example, a robot navigating in a hallway [13] may use its proximity sensors to alert it about nearby walls, and long range cameras to remotely sense obstacles blocking some hallways. When the agent must achieve some goal, it often takes different actions given the different observations it senses. For example, the robot may choose which hallway to traverse based on the output of its long range sensors. Perhaps the most standard model for such problems is the stochastic partially observable Markov decision process (POMDP) [4, 9, 11]. However, POMDPs require the specification of the probabilities of all observations and action outcomes. In many applications it is unclear how these probabilities can be obtained.

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 13–17, 2019, Montreal, Canada. © 2019 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

When the probabilities are unknown, the problem can be modeled as contingent planning under partial observability with sensing actions, specifying only which outcomes and observations are possible [2]. The solution to the contingent problem can be represented as a plan tree, where nodes are labeled by actions, and outgoing edges are labeled by possible observations resulting from the action.

Different solvers may generate a variety of different satisfying plans. In many cases some satisfying plans may be intuitively preferable to others, e.g., plans that incur a lower overall cost. One may compare plan trees by calculating the expected utility from executing each plan, but calculating the expected utility requires a probability distribution over the possible states. However, this distribution may be difficult to obtained in many cases. Thus, researchers that suggest new algorithms for computing plans often report the average number of steps to the goal, making a uniform distribution assumption, which is inapplicable in many cases.

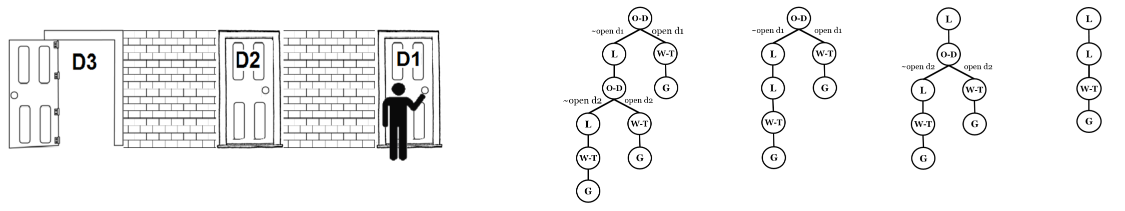
We compare complete plan trees or graphs, suggesting a number of comparison criteria. First, when a plan will be executed by people, it may be desirable to generate simpler plans. One can also consider the plan cost. Given two plan trees τ_1 and τ_2, the cost of reaching the goal for each possible initial state s using τ_1 may be lower than the cost using τ_2. However, this is a very strong criterion, and often τ_1 is better for a state s_1 while τ_2 is better for another state s_2.

Although we assume that the agent does not know the exact stochastic dynamics of the world, it may have some information concerning the likelihood of events. Given this knowledge, we offer weaker variants of dominance, limiting our attention only to the best or worst case over τ_1 and τ_2. We also consider a case where one can make assumption concerning the probability of undesirable observations, which we call *mishaps*. When mishaps are unlikely to happen, the likelihood of observing more than k mishaps may be sufficiently low to ignore. Symmetrically, when mishaps are very likely, sensing k serendipitous non-mishaps is unlikely and can be ignored. We suggest pruning branches in τ_1 and τ_2 where a certain amount of mishaps was observed.

In our journal paper, we provide efficient algorithms, where needed, to compute the various criteria, and experiments, demonstrating that different planners produce different plans that can be ranked using our methods. We also discuss the applicability of the criteria for various applications.

2 ALTERNATIVE COMPARATIVE CRITERIA

We now provide a high level description of our criteria and how they may apply in different applications, domains and scenarios. In the journal paper we provide formal descriptions, as well as algorithms for computing these criteria.



(a) The Doors problem. The agent must pass through a door. Doors 1 and 2 may be locked, Door 3 is always open.

(b) τ_1 (c) τ_2 (d) τ_3 (e) τ_4

Figure 1: The doors problem and all possible plan trees. L **denotes** ***move-left, O-D denotes*** **observe-door*****, and*** **W-T denotes** ***walk-through-door.*** $\tau_1 \underset{best}{\prec} \tau_4$, $\tau_4 \underset{worst}{\prec} \tau_1$, $\tau_4 \underset{dom}{\prec} \tau_3$.

Simplicity: In some cases, a planner produces a plan that must be executed by people. Consider for example the cave diving [1] where several divers collaborate to explore a remote underwater cave. The planner issues a set of directives for each diver, such as to swim to a certain location and leave an air tank, and the diver must then follow the instructions to ensure the success of the mission. In such cases, it may be important that the plan would be simple to understand and follow. For example, we may prefer a plan with less conditions to be checked [7]. In human navigation, a plan that checks at each crossroad which direction has less congestion, may be inferior to a longer route, that can be followed without the person constantly checking the congestion levels.

In some cases a plan is designed to be executed by an autonomous agent, but must be reviewed by a person first. For example, in navigation on remote environments, such as the Mars rover, experts may review plans to verify the correctness of the model and reduce the risk of bugs that cannot be overcome on the deployed system.

We can measure simplicity through the number of actions in a plan, or by the number of branching points in the plan.

Worst Case: The coming criteria that we suggest rely on the assumption that the environment is probabilistic, but the probabilities are unknown. Still, we may assume that the agent has a crude estimate as for the likelihood of certain outcomes. The agent must plan for all outcomes, but optimize for the likely outcomes.

The worst case criterion focuses only on the longest path to the goal, always making pessimistic assumptions about action outcomes and observations [3]. The longest route is generally important in cases where a critical mission must be accomplished, and the plan cannot exceed a given threshold. In the cave diving domain, e.g., when a diver plans his route, she must ensure that under every possible execution there is sufficient air to return to the surface.

It is possible that although the probabilities are unknown, the agent may know that the problem is hard, in that succeeding in accomplishing subtasks is difficult, and may require many retries. For example, in a penetration testing problem, the number of machines with vulnerabilities is relatively low, and identifying a vulnerable machine may require many attempts. In this case, the worst case is highly likely to occur, and optimizing for it can be useful.

Best Case: The best case criterion compares plans based on the shortest (or least costly) path to the goal, when good outcomes are much more likely than bad outcomes.

For example, in the cave diving domain, it might be that the probability for air tanks left for future divers to become faulty is very low. Although we must ensure that the worst case is below the threshold allowing us to return to the surface, we can still prefer plans with better best case behavior. We make the optimistic assumption that the air tanks will remain functional, and given the high likelihood for this, the mission will be accomplished faster.

In a logistics domain, consider two delivery companies, bidding for the same contract, with a given maximal allowed bid. Gathering statistical knowledge about, e.g., road conditions, which is needed for computing a bid requires time and resource. A company may, as a first step, optimize for the best case scenario, seeing if the net gain under the most optimistic conditions is sufficiently profitable to warrant the required resources for collecting statistical data.

Robustness to Mishaps: Motivated by fault tolerance planning [6], we define a mishap to be an undesirable observation, leading to a longer plan. Mishaps are often easy to identify using domain knowledge. For example, in pentesting [8, 12] a mishap for the attacker is when a machine does not contain a vulnerability.

We can assume that a set of events is extremely unlikely. That is, perhaps a set of more than k mishaps in a single run is extremely unlikely, or a set of less than k mishaps is extremely unlikely, depending on the domain. For example, in a recommendation scenario [5, 10], we may assume that finding a good item for a user in less than 3 suggestions is very unlikely. On the other hand, during hallway navigation, we may assume that the likelihood that more than 2 corridors are blocked concurrently is low.

When comparing, we can avoid considering unlikely branches, effectively pruning these branches from the plan tree. Then, we can apply other criteria, such as the best and worst case, on the pruned tree. For example, in the pentesting scenario, we can ignore during plan comparison branches where a vulnerability was found in less than 3 pings, optimizing for the case where the first 3 pings allowed us to discover enough information about the machine to succeed at the next attempt. In hallway navigation, we may ignore the unlikely branches where more than 2 corridors are blocked, comparing plans based only on their behavior when at most 2 corridors are blocked.

Acknowledgments

Supported by the ISF fund, under grant number 1210/18.

REFERENCES

[1] International planning competition 2014. https://helios.hud.ac.uk/scommv/IPC-14/domains_sequential.html. 2014.

[2] Alexandre Albore, Héctor Palacios, and Hector Geffner. A translation-based approach to contingent planning. In *IJCAI*, pages 1623–1628, 2009.

[3] Blai Bonet and Hector Geffner. Planning with incomplete information as heuristic search in belief space. In *AIPS*, pages 52–61, 2000.

[4] Blai Bonet and Héctor Geffner. Solving pomdps: RTDP-Bel vs. point-based algorithms. In *IJCAI*, pages 1641–1646, 2009.

[5] Darius Braziunas and Craig Boutilier. Assessing regret-based preference elicitation with the utpref recommendation system. In *Proceedings of the 11th ACM conference on Electronic commerce*, pages 219–228. ACM, 2010.

[6] Carmel Domshlak. Fault tolerant planning: Complexity and compilation. In *ICAPS*, 2013.

[7] Ellen C Garbarino and Julie A Edell. Cognitive effort, affect, and choice. *Journal of consumer research*, 24(2):147–158, 1997.

[8] Jörg Hoffmann. Simulated penetration testing: From "Dijkstra" to "Turing Test++". In *ICAPS*, pages 364–372, 2015.

[9] Sridhar Mahadevan. Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine learning*, 22(1-3):159–195, 1996.

[10] Guy Shani, David Heckerman, and Ronen I Brafman. An MDP-based recommender system. *Journal of Machine Learning Research*, 6(Sep):1265–1295, 2005.

[11] Guy Shani, Joelle Pineau, and Robert Kaplow. A survey of point-based POMDP solvers. *Autonomous Agents and Multi-Agent Systems*, 27(1):1–51, 2013.

[12] Dorin Shmaryahu, Joerg Hoffmann, Guy Shani, and Marcel Steinmetz. Constructing plan trees for simulated penetration testing. In *Proceedings of the Scheduling and Planning Applications woRKshop (SPARK), ICAPS 2016*, 2016.

[13] Jian Yang, Zhihua Qu, Jing Wang, and Kevin Conrad. Comparison of optimal solutions to real-time path planning for a mobile vehicle. *IEEE SMC-A: Systems and Humans*, 40(4):721–731, 2010.