Open World Planning via Hindsight Optimization

University of New Hampshire
Department of Computer Science
Technical Report 12-03

Scott Kiesel, Ethan Burns, Wheeler Ruml, J. Benton and Frank Kreimendahl
November 27, 2012

Open World Planning via Hindsight Optimization

Scott Kiesel[1] and **Ethan Burns**[1] and **Wheeler Ruml**[1] and **J. Benton**[2] and **Frank Kreimendahl**[1]
[1]Department of Computer Science
University of New Hampshire
skiesel, eaburns, ruml, fri2 at cs.unh.edu

[2]Smart Information Flow Technologies (SIFT), LLC
Minneapolis, MN USA
jbenton@sift.net

Abstract

In classical planning, it is assumed that all relevant aspects of the world are known at planning time. In many domains, however, the existence of certain objects or the states of certain fluents are initially unknown and must be actively discovered. Previous proposals for open world planning either employ complex and expensive knowledge representations or depend on ad hoc assumptions. In this paper, we show how hindsight optimization provides a simple and general approach to planning in open and partially observable worlds. Hindsight optimization samples multiple possible worlds that are consistent with the agent's current knowledge, generates a plan in each, and selects the action that maximizes expected reward over these samples. We demonstrate both in simulation and on a physical robot that this simple approach is more scalable than previous methods on standard benchmarks.

Introduction

Imagine a rescue robot entering a partially-destroyed building to search for survivors of an earthquake. The agent does not know the layout of the building, the locations of potential victims, or even how many there might be. In *open world* planning problems like this, the agent is not given a complete description of the initial state of the world, but it can perform sensing actions to determine the existence of relevant objects and the values of important fluents. To be useful, the planner must be fast enough to not materially delay the actions of the robot. It must be able to take into account newly sensed information, and ideally it would be expressive enough to handle soft goals, durative actions, temporal constraints, and actions with uncertain outcomes.

In this paper, we propose a simple on-line approach to open world planning which meets these criteria, which we call *Optimization in Hindsight with Open Worlds* (OH-wOW). Rather than compute a policy or contingent plan in advance, we estimate on-line at each step which action is best in light of our current knowledge. We do not commit to a particular representation for open world knowledge or goals; instead, our approach can leverage any closed-world deterministic planner appropriate for the underlying domain. Our central assumption is that the agent is not completely ignorant, that high performance in an open world depends on having expectations about that world, e.g. building dimensions are typically tens or hundreds of meters rather than centimeters or kilometers, or that people are usually found in certain densities per square meter, or are more often found in certain areas, such as offices. Such default information can be overridden by direct experience, but ought to play a role in planning. We use these expectations to generate possible states of the world consistent with the agent's knowledge, use the closed-world planner to estimate the future reward achievable in those worlds after taking each currently-applicable action, and then select the action with the highest expected reward.

We will next describe OH-wOW in detail, and then contrast it to previous work. We then report on the method's empirical performance, both in simulated domains and when deployed on a physical mobile robot fully integrated with SLAM and navigation. Our results indicate that the method is surprisingly general and practical, achieving results as good as those of previous systems with lower planning times and fewer ad hoc assumptions. This work showcases the power of Monte-Carlo techniques and adds open world planning to the list of non-classical planning settings in which simple planners can be leveraged to provide state-of-the-art performance.

A Hindsight Optimization Approach

Optimization in hindsight was originally developed for scheduling and networking problems (Chong, Givan, and Chang 2000; Mercier and van Hentenryck 2007; Wu, Chong, and Givan 2002) and has recently been applied to probabilistic planning (Yoon et al. 2008; 2010). In these previous settings, sampling is merely used to resolve uncertainty in the outcome of actions. In the context of open world planning, each sample forms a concrete hypothesis about the world—which objects might exist and which fluents might hold. For example, a rescue robot will generate various plausible floor plans for the building, each with victims in various plausible locations. While these samples are intentionally not exhaustive, they are intended to provide useful relative judgements on the expected value of actions. We apply each action, and for each resulting state find a closed-world a plan using each sampled world. The action with the lowest average plan cost is chosen.

Figure 1 summarizes the algorithm. More formally, define the value of being in a state s_1 as the minimum expected plan cost that extends from s_1. That is, the minimum over

OH-WOW($s = \langle agent, world \rangle, N$)
1. for i from 1 to N do
2. $\quad w_i \leftarrow sample\_world(world)$
3. foreach action a applicable in s
4. $\quad s' \leftarrow a(s)$
5. $\quad c \leftarrow (\sum_{i=1}^{N} solve(s', w_i))/N$
6. $\quad Q(s, a) \leftarrow C(s, a) + c$
7. Return $argmin_a\, Q(s, a)$

Figure 1: The OH-WOW algorithm.

all possible future action sequences of the total cost over all expected future states:

$$V^*(s_1) = \min_{A=\langle a_1,...,a_{|A|}\rangle} \; \underset{\langle s_2,...,s_{|A|}\rangle}{E} \left[\sum_{i=1}^{|A|} C(s_i, a_i) \right]$$

where $C(s, a)$ represents the cost of performing a in s. In open world planning, these future states incorporate the sensed knowledge of the agent (also known as its belief state) and the expectation is over the distribution of sensing outcomes. Given our expectations about sensing outcomes, we would like to find the action sequence $A = \langle a_1, ..., a_{|A|} \rangle$ that minimizes the expected sum of action costs. To compute V^* exactly, we would have to compute the expectation for each of exponentially many plans.

In optimization in hindsight, we approximate the value function by exchanging expectation and minimization, so that we are taking the expected value of minimum-cost plans instead of the minimum over expected-cost plans:

$$\hat{V}(s_1) = \underset{\langle s_2,s_3,...\rangle}{E} \left[\min_{A=\langle a_1,...,a_{|A|}\rangle} \sum_{i=1}^{|A|} C(s_i, a_i) \right]$$

This approximation of $V^*(s)$ uses fixed sensing outcomes in each minimization (sampled on lines 1–2 in Figure 1). For each possible outcome in the expectation, the problem is to minimize cost given a known world, i.e., standard, closed-world, cost-minimizing, deterministic planning (line 5). We define the Q-value to be the cumulative expected cost of taking an action a_1 in state s_1 (line 4–6):

$$Q(s_1, a_1) = C(s_1, a_1) + \underset{\langle s_2,s_3,...\rangle}{E} \left[\min_{A=\langle a_2,...,a_{|A|+1}\rangle} \sum_{i=2}^{|A|+1} C(s_i, a_i) \right]$$

From this, we estimate the best action choice in s_1 as $\min_a Q(s_1, a)$ (line 7).

Related Work

Open world planning is a broad problem that has been attacked from many angles. One issue is how to represent knowledge and goals related to open-ended sets; Etzioni and Weld (1994) and Babaian and Schmolze (2006) have addressed this. We ignore that issue in this paper, except to point out that the underlying planner used in our approach is closed-world and does not require a particularly expressive (and expensive) representation language. We do require that the agent tracks what is known and that the world generator respects this knowledge when sampling possible worlds.

In conformant planning (Cimatti, Roveri, and Bertoli 2004, inter alia), one requires plans that are guaranteed to work without sensing. For most robotics domains, this is overly restrictive and renders problems unsolvable. Contingent planning (Meuleau and Smith 2003, inter alia) allows sensing, but computes a plan before beginning execution. In addition to handling open worlds, we aim to scale to domains in which the number of contingencies may be very large (e.g., the number of possible floor plans), making synthesis of branching plans prohibitively expensive.

In the POMDP literature, computing actions on-line is recognized to provide increased scalability (Ross et al. 2008). However, most POMDP algorithms attempt to compute future belief states of the agent, which can be expensive and cumbersome. Optimization in hindsight represents an extreme approach, disregarding future belief uncertainty and assuming that the agent can achieve the cost accrued by the plans for the fully-observed sampled worlds. Our work is perhaps most closely related to work on sampling techniques for POMDPs (Silver and Veness 2010). Open world planning goes beyond POMDPs because the structure of the world state, such as how many objects exist, is unknown to the agent.

There has been sustained interest from roboticists in open world planning. One way of handling open world planning in practice is to force the robot to move in one direction simply to explore. Such simple ad hoc approaches cannot exploit the agent's expectations about goals (e.g., people are likely in offices) or take sensed information into account (e.g., a hallway implies rooms to explore). Talamadupula et al. (2010) present an approach where the planner assumes objects exist in order to instantiate goals and motivate a search and rescue robot to collect reward (discover victims). As new information arrives about the environment, the planner replans. This can be seen as a degenerate form of our hindsight approach, where the robot operates on a single optimistic "sample." While simpler, it cannot generalize to domains where uncertainty is a major component.

Joshi et al. (2012) use offline symbolic dynamic programming with known goals but unknown numbers or locations of objects, which allows for reusable policies on any instance of the domain. However, the number of possible objects were severely limited to retain feasible computation times (they require 4 hours for their 3 room example), which makes the resulting policies suboptimal. They also do not handle temporal constraints, action costs, or goal rewards.

Evaluation

We evaluate OH-WOW by applying it in two domains: the standard omelette benchmark for planning under uncertainty, and urban search-and-rescue, which we investigate both in simulation and using a physical robot.

Omelettes

In the omelette benchmark, introduced by Levesque (1996), the agent is attempting to make a three-egg omelette with ingredients of unknown freshness. The agent can *break* an egg into a bowl, *pour* the contents of a bowl into another

bowl or the trash, *wash* a bowl, or *sniff* whether the eggs in a bowl are good. All actions are deterministic except sniff. The goal is to have only three good eggs in a specific bowl. To make the domain more challenging, we extended it to have both regular white eggs, which are bad with probability 0.5, and local brown eggs, with bad probability 0.1. The agent can see the color of the next egg without sensing.

We compared OH-wOW to a perfectly omniscient oracle and to a hand-coded controller. The controller puts eggs into the goal bowl, sniffing after each and cleaning bad eggs until it finds a good one, then does the same routine using the extra bowl, pouring good eggs into the goal bowl until the goal is reached. We generated three sets of 100 random instances, each set with a different probability of the next egg being brown. OH-wOW used a domain-dependent deterministic planner based on uniform-cost search. with heuristic $h(n) = 0$.

Figure 2 (left) shows the distribution of the resulting plan costs using box plots. Each box surrounds the middle 50% of the data, with a horizontal line indicating the median and whiskers indicating the range (values beyond $1.5\times$ the interquartile range are shown as circles). The gray vertical stripes show 95% confidence intervals on the mean. The plot shows the costs over the optimal solution found by the oracle of the hindsight planner with 32 and 256 samples, and the hand-coded controller (ctlr), grouped by the probability of an egg being brown (0.0, 0.5, and 1.0). We can see that, when all eggs were white, the hindsight planner with 256 samples had a median cost that was less than the hand-coded controller (significant with $p < 0.05$ via the Wilcoxon signed-rank test). As the probability of a brown egg increased, the hindsight planner performed better, nearly dominating the controller when all eggs were brown. This is likely because the hindsight planner recognized that brown eggs tend to be good, and put multiple into a bowl before bothering to smell.

The average total planning time on a 3.1 GHz Core2 PC for OH-wOW to reach the goal using 256 samples on a problem without brown eggs was 12.9 seconds (standard deviation 8.0 seconds). This compares favorably with the 185 seconds reported for approximate RTDP (CPU unspecified) by Bonet and Geffner (2001). Levesque (2005) generates a plan to solve the three-egg omelette in 1.4 seconds but requires 1,681 seconds if four eggs are needed. With four eggs, OH-wOW's costs relative to optimal were similar to the three-egg case, and total computation time averaged only 76.7 seconds (standard deviation 43.6). The plans found by Levesque's planner also contain strictly more actions than our hand-coded controller (which in turn finds more costly plans than OH-wOW on the median), as Levesque's solution always uses the auxiliary bowl for staging, and requires one extra pour action to move the first egg from the auxiliary bowl into the goal bowl. Finally, both the techniques of Bonet and Geffner (2001) and Levesque (2005) are off-line approaches that must generate complete policies or plans before the agent can begin execution.

Search and Rescue

Finally, we return to the motivating example of rescue robotics. While returning to its starting location by a given deadline, the robot should maximize the number of injured people reported.

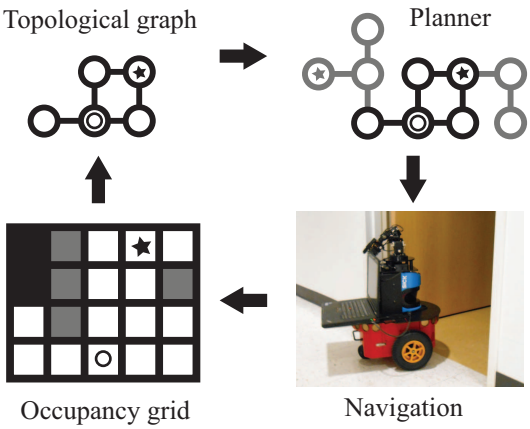

Figure 3: Architecture diagram.

To generate possible worlds for OH-wOW, we need to generate building layouts consistent with the robot's current map and hypothesize the locations of possible injured people. We represent building layouts as topological maps. We assume that undiscovered nodes will lie on a uniform four-connected grid, and that a known node can be extended if it has an adjacent grid cell that can be reached without going through an obstacle or crossing an existing edge in the map. We iteratively choose an extendible node, generate a valid neighbor, and connect them. We use a bias toward extending the most recently added node, and toward generating the neighbor that forms a straight line from the chosen node's parent. This was sufficient to yield plausible layouts with hallways. Victims are generated independently with fixed probability per hypothesized node. The upper right panel of Figure 3 shows a very small example map with hypothesized extensions shown in gray.

The base planner used by OH-wOW precomputes all-pairs shortest-paths among nodes containing people or the start location. It then uses depth-first search, considering at each step to visit an unreported person or return home. The available actions depend on the remaining time; we avoid considering time as a separate state variable by incorporating it into the cost function (Phillips and Likhachev 2011).

Simulation We created 100 random worlds with 100 nodes each. We considered three victim distributions: *unbiased*, uniform probability of 0.1 per node; *south*, nodes south of the start location 0.2 and nodes north 0; and *southwest*, southwest of the start 0.4 and 0 elsewhere. We limit the total number of victims to 10. The cost of a plan is the number of unreported people when the agent returns home and performs a dummy *finish* action. We compared OH-wOW both to an oracle that knows the exact configuration of the building and victims, and to a hand-coded controller that performed a depth-first exploration of the building, reporting people that it encountered and returning to the start location when it had no more time to explore.

To gauge the complexity of these instances, we must consider the number of possible maps and victim configurations.

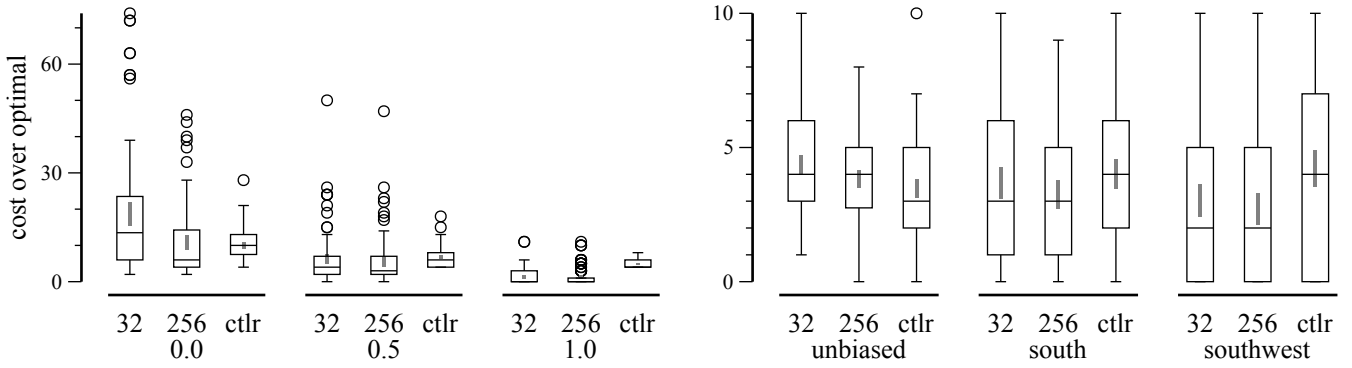

Figure 2: Plan cost in the three-egg omelette domain (left), and the search and rescue domain (right).

Considering only $n \times n$ grids, there are $2(n-1)n$ possible places for edges; our generator is limited to trees, so it must pick $n^2 - 1$. For $n = 10$, this is $\binom{180}{99} \approx 10^{52}$. For each possible map, we must choose locations for victims; for 10 victims, $\binom{100}{10} \approx 10^{13}$. Maintaining a belief over so many possible worlds would be challenging. Thankfully, it also seems unnecessary if we merely wish to estimate the expected value of actions.

Figure 2 (right) shows results, grouped by victim distribution. For the unbiased case, the hand-coded controller gave the best performance, although OH-wOW was quite competitive. With a biased distribution, OH-wOW was superior as it easily leveraged prior knowledge about the possible worlds. The average maximum per-action planning time for OH-wOW with 256 samples was 2.7 seconds (standard deviation 0.85). To compare with Joshi et al. (2012), we also ran smaller instances with at most three victims. The average maximum per-action time for 256 samples was 0.18 seconds (standard deviation 0.035), which is negligible compared to typical mobile robot latencies.

Physical Robot We also integrated OH-wOW with the Robot Operating System (ROS, `www.ros.org`) on a 3.7 GHz quad-core i7 laptop on-board a Pioneer 3dx equipped with a SICK LIDAR (and Cyton arm, not used in this work). We use the ROS Gmapping SLAM stack to generate an occupancy grid, from which we extract a topological map with edge lengths of 1 meter to pass to OH-wOW. The ROS Navigation stack is used to execute the selected action, which specifies the topological node to visit next. Experiments were performed in a hallway of approximately 20 meters with between 1 and 5 open doors to offices and 3 victims. We simulated detection of a victim when the robot occupied the same topological node as certain pre-selected locations (that were unknown to the planner).

We used deadlines of 10 minutes (6 trials found all three victims, one found two), 5 minutes (1 found 2, 2 found 1), and 1 minute (1 found 1, 2 found none). The robot always returned within the hard deadline. This demonstrates that generating possible worlds consistent with experience is feasible in practice, even as the robot's knowledge is updated. It also shows that under realistic conditions, OH-wOW correctly trades off soft goals under temporal constraints, but without the ad hoc goal handling of Talamadupula et al. (2010) or the hours of preprocessing required by Joshi et al. (2012).

Discussion

OH-wOW requires a generative model of plausible worlds. We assume such expectations can be developed either manually or through experience. When the world contradicts the agent's expectations, this can be interpreted as surprise, which might naturally lead to increased learning. The fundamental vulnerability of sampling-based planners is when unlikely worlds play a large role in determining action value; importance sampling may help here. For example, in the "Bombs in Toilets" domain (McDermott 1987; Smith and Weld 1998), OH-wOW may never sample a world in which a certain undunked package contains the bomb. The probability of this, however, is small (10^{-11} for 6 packages and 128 samples). Optimal behavior is unattainable, in any case, if one insists on fast response times in dynamic domains.

While faster than many POMDP algorithms, OH-wOW is much slower than a classical planner, as it must plan in each sampled world (see Yoon et al. (2010) for optimizations). OH-wOW is more general than standard off-line techniques as it can be used on-line and off-line by simulating the domain to construct a branching plan.

In this paper, we assume that the world remains static as we explore it and that non-sensing actions are deterministic, however OH-wOW is very general and immediately applies to dynamic worlds, stochastic actions, and on-line goal arrival; this remains an exciting area for future work.

Conclusion

Open world planning is essential for many real-world agents. We have shown how optimization in hindsight yields a simple and general approach to open-world planning with temporal constraints, decision-theoretic reasoning, and soft goals. While the technique is approximate, it is easy to implement and our results suggest that it can be successful in practice.

Acknowledgments

This work was supported in part by NSF (grant 0812141) and the DARPA CSSG program (grant D11AP00242).

References

Babaian, T., and Schmolze, J. G. 2006. Efficient open world reasoning for planning. *Logical Methods in Computer Science* 2(3).

Bonet, B., and Geffner, H. 2001. GPT: a tool for planning with uncertainty and partial information. In *Proc. IJCAI-01 Workshop on Planning with Uncertainty and Partial Information*, 82–87.

Chong, E.; Givan, R.; and Chang, H. 2000. A framework for simulation-based network control via hindsight optimization. In *IEEE Conference on Decision and Control*.

Cimatti, A.; Roveri, M.; and Bertoli, P. 2004. Conformant planning via symbolic model checking and heuristic search. *Artificial Intelligence* 159(1–2):127–206.

Etzioni, O., and Weld, D. S. 1994. A softbot-based interface to the internet. *Communications of the ACM* 37(7):72–76.

Joshi, S.; Schermerhorn, P. W.; Khardon, R.; and Scheutz, M. 2012. Abstract planning for reactive robots. In *Proceedings of IEEE ICRA*, 4379–4384.

Levesque, H. 1996. What is planning in the presence of sensing? In *Proceedings of AAAI*.

Levesque, H. J. 2005. Planning with loops. In *Proceedings of IJCAI*.

McDermott, D. 1987. A critique of pure reason. *Computational Intelligence* 3(1):151–160.

Mercier, L., and van Hentenryck, P. 2007. Performance analysis of online anticipatory algorithms for large multistage stochastic programs. In *Proceedings of IJCAI*.

Meuleau, N., and Smith, D. E. 2003. Optimal limited contingency planning. In *Proceedings of UAI*.

Phillips, M., and Likhachev, M. 2011. Planning in domains with cost function dependent actions. In *Proceedings of the fourth international symposium on combinatorial search (SoCS-11)*.

Ross, S.; Pineau, J.; Paquet, S.; and Chaib-draa, B. 2008. Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research* 32:663–704.

Silver, D., and Veness, J. 2010. Monte-carlo planning in large POMDPs. In *In Advances in Neural Information Processing Systems 23*, 2164–2172.

Smith, D. E., and Weld, D. S. 1998. Conformant graphplan. In *Proceedings of AAAI*, 889–896.

Talamadupula, K.; Benton, J.; Schermerhorn, P.; Kambhampati, S.; and Scheutz, M. 2010. Integrating a closed world planner with an open world robot: A case study. In *Proceedings of AAAI*.

Wu, G.; Chong, E.; and Givan, R. 2002. Burst-level congestion control using hindsight optimization. *IEEE Transactions on Automatic Control*.

Yoon, S.; Fern, A.; Givan, R.; and Kambhampati, S. 2008. Probabilistic planning via determinization in hindsight. In *Proceedings of Conference on Artificial Intelligence (AAAI)*.

Yoon, S.; Ruml, W.; Benton, J.; and Do, M. B. 2010. Improving determinization in hindsight for on-line probabilistic planning. In *Proceedings of the Tenth International Conference on Automated Planning and Scheduling (ICAPS-10)*.