How Much Does a Household Robot Need To Know In Order To Tidy Up?

Bernhard Nebel and **Christian Dornhege** and **Andreas Hertle**
Department of Computer Science
Georges-Köhler-Allee 52
University of Freiburg, Germany
email: {nebel, dornhege, hertle}@informatik.uni-freiburg.de

Abstract

Although planning for the tasks a household robot has to perform appears to be easy, there exists the problem that the robot is usually uncertain about the state of the household when starting to plan. For example, when getting the order of tidying up the kitchen, the robot does not know what objects it will have to put away and whether there are actually any objects that need to be put away. Furthermore, while sensing operations can provide more information about the environment, things can go wrong when executing an action.

In this paper, we try to identify conditions under which classical planning can be used in a replanning loop in order to solve the planning problem in nondeterministic partially observable open domains. In particular, we will define completeness and soundness of replanning with respect to nondeterministic planning and we will identify a PSPACE-checkable condition that guarantees soundness.

Introduction

Planning for the tasks a household robot has to perform appears to be relatively easy. However, there exists the problem that the robot is usually uncertain about the state of the household when starting to plan. For example, when getting the order of tidying up the kitchen, the robot does not know what objects it will have to put away and whether there are actually any objects that need to be put away. Furthermore, while sensing operations can provide more information about the environment, things can go wrong when executing an action.

This kind of situation calls for techniques to plan in nondeterministic partially observable open domains. Open domains, i.e. domains with an arbitrary, unbounded number of possible objects is, however, undecidable (Erol, Nau, and Subrahmanian 1995). Even if we restrict the number of objects to those that can be found in one particular household (either by knowing about them or by discovering them), the problem is still very difficult. Symbolic planning in nondeterministic, partially observable domains is 2-EXP-complete (Rintanen 2004).

We propose to compile such a planning problem into a classical planning problem preserving soundness and completeness under some reasonable conditions. The task is solved by embedding a classical planner in an observation-monitoring-execution loop commonly referred to as continual planning. We also focus on the problem of checking whether a particular planning task satisfies these conditions and identify some cases for which one can easily check the conditions.

While others have addressed this problem before, we see our contribution in spelling out precisely what it means that a replanning approach is sound and complete with respect to a nondeterministic planning domain, how one can guarantee it, and how expensive it is to check these conditions.

As a proof of concept (which was actually the starting point of our investigation), we describe how the solution was incorporated in the Tidy-Up project.

Scenario

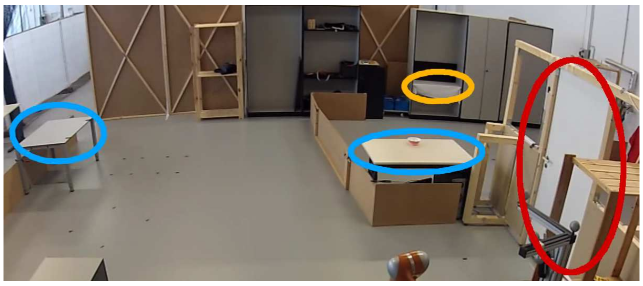

Figure 1: Overview of our experimental household environment. Our test scenario contains two rooms separated by a door (red), tables (blue) and a shelf (yellow) in the back.

In the context of this paper we refer to a household scenario involving multiple different skills within the field of mobile manipulation. The task of tidying up serves as a generic example to demonstrate the problems that a robot faces in a real-world setting. Such a setting contains multiple rooms that are separated by doors that might need to be opened to navigate between rooms. An overview of our test scenario can be seen in Figure 1. An unknown number of objects are placed on tables and shelves in the rooms. The goal is to make sure that all objects are at the location where they belong—their *tidy-location*. In addition, we require that all spots, where objects were found, should be wiped clean.

We only assume that the robot has knowledge about the static parts of the world, i.e. there is a map of the environment and the robot knows where doors and tables are located. There is no prior knowledge about movable objects not even the knowledge that an object exists. This means that the robot will need to acquire that knowledge by searching all possible locations.

As we are dealing with a real-world system any action, including sensing actions, might have multiple nondeterministic effects. Sensing actions might also add additional objects in the task.

Problem Formalization

In order to control the robot described in the scenario above with the help of an automatic planning system, the adequate planner would be one that is able to deal with:

1. open domains (with an unlimited number of objects),
2. with uncertain initial states,
3. nondeterministic (or probabilistic) effects of actions,
4. and sensing operations (in order to acquire knowledge about the environment).

If we ignore the first point for the time being, such planning problems could be, for instance described using the planning language NPPDL (Bertoli et al. 2003), an extension of PDDL that is able to deal with uncertainty, nondeterminism, partial observability and sensing. It should be noted that this language does not support an explicit knowledge or belief modality. Rather all preconditions, effect conditions, and goal specifications are assumed to be implicitly in the scope of a modal belief operator.

One could use the planner MBP (Bertoli et al. 2001), CAltAlt or POND (Bryce, Kambhampati, and Smith 2006) in order to solve planning problems specified in NPDDL. However, given the size of the search space in our domain, such planners will not be able to generate plans in a reasonable time as is evident from the performance data as reported, e.g., in the paper by Bryce, Kambhampati, and Smith (2006).

It seems as a bit of an overkill that a household robot should be able reason about multiple possible situations. A household robot certainly is not supposed to solve Whodunit puzzles or diagnose the failure of the washing machine. Well, at least the cheap models of a household robot are not expected to do that.

Furthermore, it also seems a bit over-cautious to plan for all contingencies in advance, given that the household domain (as many others) is quite forgiving concerning wrong choices or guesses. Also humans plan in most cases without considering all possibilities.

So, as many other have proposed and done, we use a *continual planning* approach, where we plan for one way to solve the planning problem at hand, and replan if anything does not work out according to plan. In this case, it is possible to make use of a traditional, classical planner, in our case, Temporal Fast Downward/Modules (TFD/M) (Dornhege et al. 2009; Eyerich, Mattmüller, and Röger 2009). Such an approach leads to two important questions:

1. How to compile a particular feature in the original planning task description away?
2. Under what conditions can we expect that the approach guarantees that we can reach the goal?

Simplifications and Guarantees

It is clear that we lose optimality, completeness, and perhaps soundness. However, often enough, we might be able to still guarantee that the goal can be reached. Further, we might be able to quantify the computational resources necessary to decide whether this is the case.

Expanding Universes Instead of Open Universes

The first issue we want to address is the mismatch between the *domain closure assumption* in most planning systems and the fact that it seems rather unrealistic to assume that all (relevant) objects are known from the beginning.

Semantically, one would assume an infinite (open) domain, where for all types of objects, we have a countable set of such objects, whereby for most of them we do not know anything about them, e.g., such as where they are located. Such infinite domains lead to the problem that the planning problem may become undecidable and that the technique of transforming a first-order specification of the planning domain into a propositional logic theory does not work any longer.

One way out could be to introduce new objects only when they are detected by the robot when observing its environment. After such an introduction of a new object, one can use replanning to deal with the changed domain. This approach can deal with all the objects we encounter in a household or similar environment and we are supposed to deal with. However, necessary tools (e.g. a hammer) that the robots needs have to be amongst the known objects in advance.

This is indeed the strategy we chose, but it comes, of course, with a price. It leads necessarily to incompleteness since it might be the case that we need a unbounded number of yet unknown tools to achieve our goal. So our planner might conclude that the goal is not reachable although with the right number of tools it would have been. And there is no way around it because first-order STRIPS-planning with the possibility to introduce new objects is already undecidable (Erol, Nau, and Subrahmanian 1995).

Limited Uncertainty Through Kleene's Strong Three-Valued Logic

Uncertainty about the current state is semantically usually represented by a *belief set*, a set of states that are believed to be possible. In NPDDL, uncertainty is introduced by the specification of the initial situation and by nondeterminism in action effects. So, for example, it can be that in a particular situation, "it is believed that cup is broken or the bottle is full". If the robot now observes that the cup is not broken, it will conclude that the bottle must be full.

This kind of puzzle-mode reasoning is, of course, necessary when we want to solve Whodunits or want to diagnose the failure of a machine. However, we do not expect our household robot to be Colombo or a mechanic. So, a simpler way to represent uncertainty might be enough.

Most of the time, we are just uncertain about the values of a fluent and do not consider the connections between different fluents. So, one could extend the value domain (may it be Booleans or many-valued fluents) by a value *unknown*. The evaluation of logical formulae can then be based on *Kleene's strong three-valued logic* (Kleene 1950). This logic does just what you would expect when combining known with unknown truth values. Alternatively, one could view this three-valued logic as a dense representation of what Petrick has called *Cartesian situations* (Petrick 2011).

Of course, such a representation cannot represent that "it is believed that the cup is broken or the bottle is full", or more generally, any disjunctive knowledge. So, if we use such a representation to approximate a given belief set, we will lose completeness. For example, we would over-approximate the above statement about the cup and the bottle by "it is unknown whether the cup is broken" and "it is unknown whether the bottle is full". So finding out that the cup is not broken does not help the robot at all.

In fact, this representation implies that the only way to find out about the truth value of a fluent is to know it either from the beginning, to learn about it by *sensing* its truth value, or by setting the truth value as an effect of an action. Since we do not expect our household robot to be a scientist, we assume that all fluents are observable once the robot is spatially close enough to the location where the fluents can be sensed. So, we do not assume that the robot knows and reasons about hidden variables.

Furthermore, while we assume that the actions of the robot can have nondeterministic effects, we also assume that the robot can only change fluent values of fluents relevant in a spatial area that can also be observed by the robot after the action. In other words, by monitoring the outcomes of the actions, the robot can determine the true effects of its actions after each action.

With these two assumptions, which are admittedly quite strong, the representational move of using Kleene's strong three-valued logic instead of general belief sets does not sacrifice completeness, but of course, optimality. However, we also have now only uncertainty due to the initial state and the degree of uncertainty (as measured by the number of fluents that have the value *unknown*) shrinks monotonically.

It also means that we have reduced the search space from double exponential (belief sets = set of sets of state) to exponential (the set of states).

Continual Replanning Instead of Conditional Planning

While we have reduced the search space of our problem considerably, there is still the problem that after sensing the value of a fluent (be it after an action execution or after a sensing action), we may have to branch according to the sensed value. In fact, what we have done is to simplify the nondeterministic planning problem under partial observability to a nondeterministic planning problem under full observability, where the nondeterminism is created by the possible outcomes of an action followed by a monitoring action or by pure sensing actions. In terms of computational complexity, this is a reduction from 2-EXP (Rintanen 2004) to EXP (Littman 1997). This is, in fact, what is behind Petrick's knowledge-based approach to planning (Petrick and Bacchus 2002; Gaschler et al. 2013).

The classical way to deal with this problem is to generate plans that can branch and perhaps loop. Alternatively and equivalently, a policy (a mapping from states to actions) is created. In this context, one distinguishes *weak plans*, *strong plans*, and *strong cyclic plans*. Weak plans are just sequences of action whereby one has chosen nondeterministic outcomes arbitrarily. Strong plans are winning strategies against the nondeterministic environment. Strong cyclic plans are strategies that guarantee that we never leave the set of states from which the goal is reachable.

For our household robot, strong and strong cyclic planning definitely sounds like overkill. While it would, of course, be nice to create a plan taking into account the state of all doors and the positions of all cups, it does not seem to be worth the effort. In fact, the household domain as a lot of other domains seem to be very forgiving concerning non-optimal or even wrong choices. So, the straight-forward approach would be to develop a plan for the *most likely* outcomes (which have to be marked as such), monitor the success of the plan (which we do anyway in order to reduce uncertainty) and replan if things do not advance as predicted.

It has been shown using a classical planner such as FF (Hoffmann and Nebel 2001) in a replanning loop can easily outperform probabilistic planners on many probabilistic planning problems. The reason was that in most cases considered, the planning domains were *probabilistic uninteresting* (Yoon, Fern, and Givan 2007) meaning that the probabilities had in fact very little impact.

The household domain (as many other domains) probably have a similar property of being *nondeterministically uninteresting*, whereby the nondeterminism as mentioned above comes from the outcomes of sensing actions. So, we propose, as many others before us to replace conditional planning by classical planning with replanning. The main question is, under which conditions this is an approach where we do not lose soundness and completeness.

Before we can answer this question, we have to define what soundness and completeness in a replanning context compared to nondeterministic planning really means. Here we have to take into account all stages of what can happen in a replanning context.

Completeness would mean that if there is strong cyclic plan, then the classical planner is able to generate a successful straight-line plan for each state that could be reached in the strong cyclic plan. This is trivial to achieve provided the classical planner is complete.

Soundness would mean that the classical planner generates only straight-line plans that are traces of one possible execution of a strong cyclic plan. In particular, this implies that if there is no strong cyclic plan, then the replanning agent should not generate a straight-line plan. Moreover, the planner shall never generate a plan that leads to states that are not reachable by a strong cyclic plan. These are, of course, quite severe restrictions that do not seem to be easily satisfiable. In particular, in order to meet them, one has to solve essentially the nondeterministic planning problem!

However, it is possible to specify sufficient conditions on the topology of the search space under which they are satisfied. The main point is that we never want to “paint ourselves into a corner,” or phrasing it differently ending up in a *dead end* of the search space. Here we define a dead end as a state from where no weak plan leads us to the goal. Now, in order to avoid such dead ends, one way would be to assume that they are not reachable by a weak plan from the initial state. In fact, this assumption does the trick.

Note that our condition is a little bit weaker than the condition posed by Kaelbling and Lozano-Pérez (2011), who required to have *reversibility* in their domain, meaning that one has strong connectivity in the search space. Instead, we only assume that we are guaranteed to reach the goal. The difference is that under our assumption, one can for example throw away trash irreversibly, which is not possible under the *reversibility* restriction.

However, the big question is whether this condition can be easily checked. As it turns out, it is easier than nondeterministic planning and only as complex as classical planning.

Theorem 1 *Checking whether there exists a dead end reachable by a weak plan from a given initial state in the search space of a propositional nondeterministic planning task is PSPACE-complete.*

Proof. Membership in PSPACE is demonstrated by the following nondeterministic algorithm for a given initial state s and set of goal states G :

1. Guess a state d.
2. Verify that d is reachable from s by weak plan, which is a problem that is in PSPACE
3. Verify that no state $g \in G$ is reachable from d, which is a problem complementary to the classical planning problem, hence also in PSPACE.

PSPACE-hardness follows from a simple reduction of the problem at hand to classical propositional planning. □

Since PSPACE is closed under complements, it is also PSPACE-complete to check whether a given set of states is dead-end free.

Interestingly, although having the same complexity as classical planning, it is far from obvious how to implement such a check efficiently, the main problem being the question how to identify dead ends.

One interesting question is of whether it would be possible to avoid such dead ends, if they are known in advance. So the problem is to check for a given state, whether the selection of an action can lead to the identified dead ends if all nondeterministic choices are worst-case. However, this is just the question of whether there exists a strong plan for the environment forcing us into the dead-end state. In other words, the decision problem is as hard as nondeterministic planning, i.e., EXP-hard.

So, sticking to our assumption that our domains are dead-end free, it is possible to guarantee that a replanning approach will eventually reach a goal. The reasoning behind it is that by the fact that uncertainty is reduced monotonically, the number of nondeterministic choice points due to sensing actions reduces from one replanning episode to the next. For this reason, eventually, we will have a completely informed state in which a classical plan is sufficient to reach the goal, or we have reached a goal state before this—provided the probability of the desired nondeterministic effect is different from zero.

Proof of Concept

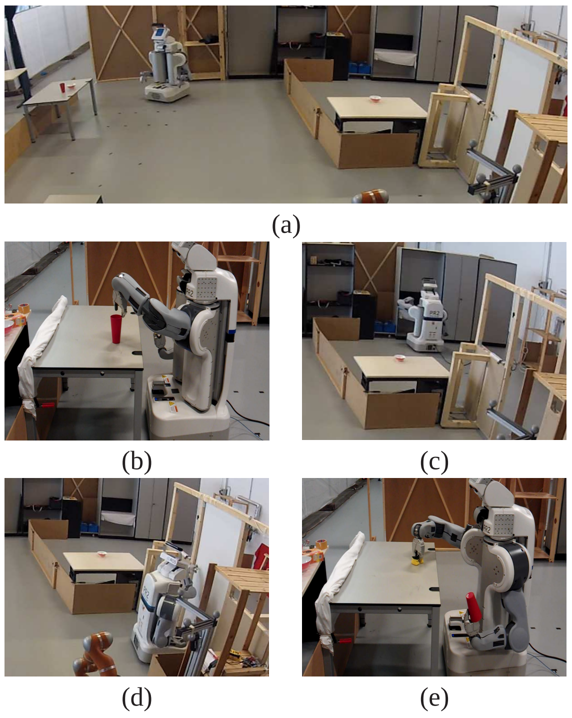
(a) (b) (c) (d) (e)

Figure 2: Example scenes from the PR2 robot operating in the Tidyup-Robot domain. Our test scenario (a) contains two rooms separated by a door, tables and a shelf in the back. Tidying up requires to find objects, pick them up (b) and bring them to the tidy location (c). Navigation might require to open doors (d). Finally, tables, where the objects were found, should be wiped clean (e). A video is available at: `http://www.youtube.com/watch?v=pTSz2RBZ2wA`

Using the techniques described in this paper, we implemented the example scenario on the PR2 mobile manipulation robot as part of the *Tidyup-Robot* project (see Figure 2). We use the Robot Operating System (ROS) to provide robot skills (Quigley et al. 2009). Our TFD/M planner (Dornhege et al. 2009) is employed as a classical planner embedded in a continual planning loop. The planner supports integrated task and motion planning via the concept of semantic attachments. One can compute action costs, truth values of fluents, or numerical effects by *cost modules*, *condition checker* modules, and *effect applicator* modules respectively.

For navigation we use a search-based motion planner with full 3d collision checking that uses the ARA* algorithm (Likhachev, Gordon, and Thrun 2004). The navigation costs are integrated as a *cost module* into the planning process. For manipulation tasks, we employ two modules for put-down actions: A *condition checker* that determines if an object can be put down on a table and an *effect applicator* that provides the numerical state update writing the resulting pose to the state. The implementation is based on a discretization of possible positions on the table. Wiping tables is based on a coverage approach (Hess, Tipaldi, and Burgard 2012).

An explicit sensing action based on ROS' default object recognition is used as a precondition to all manipulation actions. Upon each observation recognized objects are matched on type and distance to previously known objects and new objects are added to the state. In addition we require that all locations are sensed at least once, otherwise, given the expanding universe model, initially all known objects, which are none, will be tidied up and thus the problem will be declared solved. Note that this is the same procedure that humans employ when tasked with tidying a room: The actual locations that need cleaning are identified before stating that everything is done. The actual goal we are solving is thus: "Can you please make sure that the rooms are tidied up?"

Related Work

A number of planners have been developed that deal with nondeterminism, partial observability, and sensing. For instance the planning system MBP, which is based on BDDs (Bertoli et al. 2001) and the planning systems POND and CAltAlt (Bryce, Kambhampati, and Smith 2006; Bryce 2006), a lazy approach to representing belief sets (Hoffmann and Brafman 2005), and a compilation approach to solve nondeterministic planning problems using classical planning (Kuter et al. 2008).

As mentioned above, we do not believe that these approaches can scale up to household domains with a large number of objects. Here we believe, planning approaches that do not try to solve the entire problem offline are more adequate. One approach is, for example, to interleave planning for nondeterministic partially observable domains with planning as proposed by Bertoli et al. (2004). Most such approaches, however, simplify the planning problem considerably. While often there is a simple plan-execute-monitor loop, there are also other approaches more tightly integrating planning and execution (Ambros-Ingerson and Steel 1988; Brenner and Nebel 2009; Knoblock 1995). Nobody has tried to precisely determine under which conditions such approaches are feasible and how expensive the verification of such a condition would be, though.

Work, that is in particular related to our paper is anything that uses automatic symbolic planning to control a robot. In particular, in recent years, robot-control by automatic planning is considered again as a realistic means to achieve autonomy. The reason for that is that automatic planning has become much more efficient, as demonstrated by recent events such as the *International Planning Competitions*.

Notably, Kaelbling and Lozano-Pérez (2011) have developed a robot planning system that integrates task and motion planning. This system uses a hierarchical regression planner and uses the refined prefix of the computed plan to start execution. As mentioned above, they also use a replanning approach which accounts for wrong choices in the planning process (and implicitly for execution failures). As also mentioned, they require the stronger property that a domain should be reversible. Another interesting approach related to our work is the work by Gaschler et al. (2013), which demonstrates how Petrick's belief-space planner can be used in a real robot environment.

Conclusions and Future Work

We have formulated the generic planning problem for a mobile manipulation robot in the real-world as an open partially-observable nondeterministic planning problem. As we are aiming for efficient solutions to this problem, we simplified the generic problem to a classical planning problem running within a continual planning loop. We stated what soundness and completeness means in this case and have given explicit conditions under which we retain those properties. In addition we have shown that the computational complexity to determine if our assumption of no dead-ends holds for a problem is PSPACE-complete. Finally a proof-of-concept implementation has shown the feasibility of the simplifications in a real-world scenario.

In the future we plan to rigorously formalize the notions introduced in this paper and formally prove those claims. We also look for more easily verifiable conditions that guarantee soundness and seek to identify safeguard strategies that avoid getting trapped. Experimentally proving the applicability of our approach will require to compare the performance of our implementation with POND planners.

Acknowledgments

This work was partially supported by DFG as part of the SFB/TR8 project R7 and in the PACMAN project within the HYBRIS research group (NE 623/13-1). We would like to thank all members of the *Tidyup-Robot* project.

References

Ambros-Ingerson, J. A., and Steel, S. 1988. Integrating planning, execution and monitoring. In *Proceedings of the 7th National Conference of the American Association for Artificial Intelligence (AAAI-88)*, 83–88.

Bertoli, P.; Cimatti, A.; Roveri, M.; and Traverso, P. 2001. Planning in nondeterministic domains under partial observability via symbolic model checking. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI)*, 473–478.

Bertoli, P.; Cimatti, A.; Dal Lago, U.; and Pistore, M. 2003. Extending pddl to nondeterminism, limited sensing and iterative conditional plans. In *Proceedings of ICAPS-03 Workshop on PDDL*. AAAI Press.

Bertoli, P.; Cimatti, A.; and Traverso, P. 2004. Interleaving execution and planning in nondeterministic partially observable domains. In *Proceedings of the 16th European Conference in Artificial Intelligence (ECAI 04)*, 657–661. IOS Press.

Brenner, M., and Nebel, B. 2009. Continual planning and acting in dynamic multiagent environments. *Journal of Autonomous Agents and Multiagent Systems* 19(3):297–331.

Bryce, D.; Kambhampati, S.; and Smith, D. E. 2006. Planning graph heuristics for belief space search. *Journal of Artificial Intelligence Research* 26:35–99.

Bryce, D. 2006. POND: the partially-observable and nondeterministic planner. In *Fifth International Planning Competition (IPC)*.

Dornhege, C.; Eyerich, P.; Keller, T.; Trüg, S.; Brenner, M.; and Nebel, B. 2009. Semantic attachments for domain-independent planning systems. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 114–121. AAAI Press.

Erol, K.; Nau, D. S.; and Subrahmanian, V. S. 1995. Complexity, decidability and undecidability results for domain-independent planning. 76(1–2):75–88.

Eyerich, P.; Mattmüller, R.; and Röger, G. 2009. Using the context-enhanced additive heuristic for temporal and numeric planning. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 130–137. AAAI Press.

Gaschler, A.; Petrick, R. P. A.; Kröger, T.; Knoll, A.; and Khatib, O. 2013. Robot task planning with contingencies for run-time sensing. In *Workshop on Combining Task and Motion Planning at ICRA*. to appear.

Hess, J.; Tipaldi, D.; and Burgard, W. 2012. Null space optimization for effective coverage of 3d surfaces using redundant manipulators. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1923–1928.

Hoffmann, J., and Brafman, R. 2005. Contingent planning via heuristic forward search with implicit belief states. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 71–80.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14:253–302.

Kaelbling, L., and Lozano-Perez, T. 2011. Hierarchical task and motion planning in the now. In *IEEE Conference on Robotics and Automation (ICRA)*.

Kleene, S. C. 1950. *Introduction to Metamathematics*. Princeton, NJ: D. Van Nostrand.

Knoblock, C. A. 1995. Planning, executing, sensing, and replanning for information gathering. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, 1686–1693. Montreal, Canada: Morgan Kaufmann.

Kuter, U.; Nau, D. S.; Reisner, E.; and Goldman, R. P. 2008. Using classical planners to solve nondeterministic planning problems. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 190–197.

Likhachev, M.; Gordon, G. J.; and Thrun, S. 2004. ARA*: Anytime A* with provable bounds on sub-optimality. In Thrun, S.; Saul, L.; and Schölkopf, B., eds., *Advances in Neural Information Processing Systems 16*. Cambridge, MA: MIT Press.

Littman, M. L. 1997. Probabilistic propositional planning: Representations and complexity. In *Proceedings of the 14th National Conference of the American Association for Artificial Intelligence (AAAI)*, 748754. Portland, OR: MIT Press.

Petrick, R. P. A., and Bacchus, F. 2002. A knowledge-based approach to planning with incomplete information and sensing. In *Proceedings of the 6th International Conference on Artificial Intelligence Planning Systems (AIPS)*, 212–222.

Petrick, R. P. A. 2011. Knowledge, action, and cartesian situations in the situation claculus. In Lakemeyer, G., and McIlraith, S. A., eds., *Knowing, Reasoning and Acting*. London, UK: College Publications. 379–398.

Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; and Ng, A. 2009. ROS: an open-source robot operating system. In *ICRA Workshop on Open Source Software*.

Rintanen, J. 2004. Complexity of planning with partial observability. In Zilberstein, S.; Koehler, J.; and Koenig, S., eds., *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 345–354. AAAI Press.

Yoon, S. W.; Fern, A.; and Givan, R. 2007. FF-replan: A baseline for probabilistic planning. In *Proceedings of the International Conference on Automated Planning and Scheduling, (ICAPS)*, 352–. AAAI Press.