Continual HTN Robot Task Planning in Open-Ended Domains: A Case Study

Dominik Off and **Jianwei Zhang**
TAMS, Department of Informatics, University of Hamburg
Vogt-Koelln-Strasse 30, 22527 Hamburg, Germany
{off,zhang}@informatik.uni-hamburg.de

Abstract

The fact that many AI planning approaches are still based on too simplifying assumptions makes it often hard to apply these approaches to real-world robotics. In particular, it is in many cases difficult to generate a complete plan in advance, because not all information is available at the beginning of the planning process. We briefly present the continual planning system ACogPlan and a preliminary test case that demonstrates how the planning system can enable mobile robots to continually plan and execute activities in an open-ended domain.

Introduction

High-level reasoning and task planning are essential if we want to enable robots to autonomously perform high-level tasks (e.g., "Bring me a cup of coffee"). Planning algorithms have been developed that in principle are efficient enough to solve complex planning problems in real time. However, many AI planning approaches are still based on too simplifying assumptions. Therefore, it is in many cases hard to apply these approaches to real-world robotics. In particular, the fact that in real-world scenarios often not all necessary information is available at the beginning of the planning process makes it difficult to a priori generate a complete plan—as necessary in the most AI planning approaches. Most of the previous approaches that are able to generate plans in partially known environments generate conditional plans—or policies—for all possible contingencies. Unfortunately, planning approaches that generate conditional plans are computationally hard, scale badly in dynamic unstructured domains and are only applicable if it is possible to foresee all possible outcomes of a knowledge acquisition process (Ghallab, Nau, and Traverso 2004; Brenner and Nebel 2009).

Mobile robots can usually acquire additional information from a multitude of sources. Thus, we believe that *continual planning* (Brenner and Nebel 2009) is a more promising approach for mobile robots, since it enables them to interleave planning and execution such that missing information can be acquired via active information gathering. However, if we want to enable robots to autonomously acquire information by means of active information gathering, then—as already pointed out by (Nau 2007)—we have to enable them to answer the following questions: What information to look for? How to acquire the necessary information? Yet, these questions have not been sufficiently addressed by existing planning approaches (Nau 2007). We believe that this is one major reason why it is still difficult to apply AI planning to real-world robots that inhabit a partially known environment.

Continual Planning in Open-Ended Domains

If we want to enable robots to autonomously extend their knowledge about the environment by means of active information gathering, then the planning system needs to be able to derive which extensions of the current domain model are relevant and possible. Most planning systems are unable to do that, since their underlying domain model is based on the assumption that all information is available at the beginning of the planning process (Nau 2007). In contrast, the proposed continual planning system—called *ACogPlan*—is based on the open-ended domain model *ACogDM* (Off and Zhang 2011). ACogDM enables the planner to reason about relevant extensions of its domain model. It is particularly intended for forward search (i.e., forward decomposition) *Hierarchical Task Network (HTN)* planning approaches like ACogPlan. Forward decomposition HTN planners choose between a set of *relevant* (Ghallab, Nau, and Traverso 2004, Definition 11.4) methods (i.e., STN methods) or planning operators (i.e., actions) that can be in principle applied to the current *task network*. Let σ be a substitution. A relevant method or planning operator can actually be applied if and only if its precondition p holds (i.e., an instance $p\sigma$ is derivable) with respect to the given domain model. Therefore, we define the set of *relevant preconditions* with respect to a given *planning context* (i.e., a domain model and a task list) to be the set of all preconditions of relevant methods or planning operators. A HTN planner cannot continue the planning process in situations where no relevant precondition is derivable with respect to the domain model at hand. The notation of a relevant precondition is a first step to determine relevant extensions of a domain model, since only domain model extensions that make the derivation of an additional instance of a relevant precondition possible constitute an additional way to continue the planning process. All other possible extensions are irrelevant, because they do not

imply additional planning alternatives. In other words, if it were possible to acquire additional information which implies the existence of a new instance of a relevant precondition, then the planning process could be continued in an alternative manner. This is particularly relevant for situations in which it would otherwise be impossible to find any plan at all. In order to formalize this we introduce the following concepts in (Off and Zhang 2011): a *possibly-derivable statement* (e.g., a precondition) and an *open-ended literal*. Let L_x be a set of literals and p be a precondition. p is called *possibly-derivable* iff the existence of a new instance $l\sigma$ for each $l \in L_x$ implies the existence of a new instance $p\sigma$ of p. Obviously this definition is only useful if the existence of an additional instance for each $l \in L_x$ is possible. A literal for which the existence of non-derivable instances is possible is called *open-ended*. Based on that, one can say that a possibly-derivable precondition constitutes the partition of a precondition into a derivable and an open-ended part (i.e., a set of open-ended literals).

The general idea of the underlying planning algorithm of ACogPlan is to behave like a closed-world assumption based planner (i.e., the TFD procedure as specified in (Ghallab, Nau, and Traverso 2004, Figure 11.4)) as long as sufficient information is available. However, if necessary information is missing, then the planner generates and executes a knowledge acquisition plan for the open-ended part of a relevant precondition and continues the initial planning process based on the updated domain model. By this means the planner automatically switches between planning and acting such that missing information is acquired via active information gathering.

Full System Test Case

The proposed planning system is implemented on the mobile service robot platform TASER. We performed a first simple test case in the office environment of our institute in order to demonstrate the system behaviour. The only used external knowledge source in this test case is perception. The robot was instructed to perform the task of delivering a mug (Bob's mug) into the kitchen. In this test run the robot has no information about the state of doors and therefore cannot generate a complete plan in advance.

TASER successfully performed (i.e., executed) the task. The overall execution is composed of six planning and execution phases as illustrated in Figure 1. Actions that are directly executed by a corresponding robot control program are printed blue and marked with the symbol "▶". All other tasks are non-primitive and cannot be directly executed. The fact that only a partial plan exists for a task is illustrated by a subsequent "`[...]`". Furthermore, the result of a sensing acting is shown under the corresponding task.

At the first planning phase the planner generates a complete plan that determines how to pick up Bob's mug. Non-primitive tasks that have no subsequent "`[...]`" and are not further decomposed usually indicate the situation that nothing has to be done to perform the task. For example, in the first phase the task $move\_to(lab)$ is not further decomposed, because the robot initially is in the lab. Due to the fact that the planner had no information about the

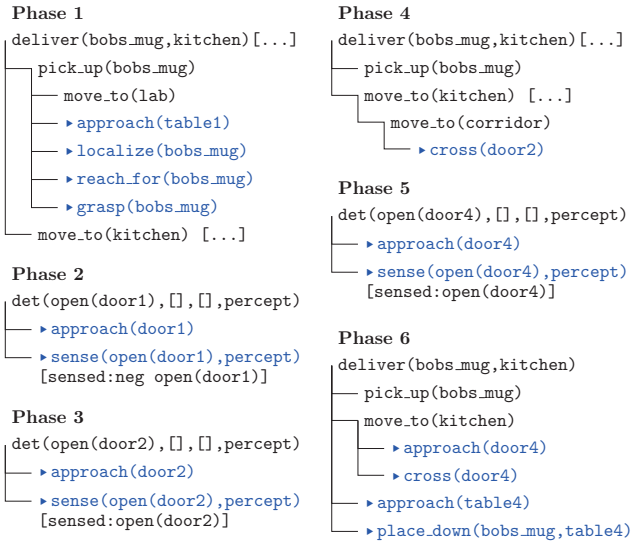

Figure 1: Execution phases of the full system test case

state of the doors it could not generate a plan for the task $move\_to(kitchen)$. The planner decides to execute the plan for $pick\_up(bobs\_mug)$ and then starts the second planning and execution phase in order to determine whether the first lab door is open. During the second execution phase the robot determines that the first lab door is closed. In order to avoid the more expensive door opening procedure the planner decides to determine whether the second lab door is open at the third planning and execution phase. TASER determines that the second lab door is open and can continue to perform the initial task (i.e., bring Bob's mug into the kitchen). In the fifth phase, the robot determines that the kitchen door is open. After the fifth phase all necessary information is available and the robot successfully finishes its task in the last execution phase.

Acknowledgements

This work is founded by the DFG German Research Foundation (grant #1247) – International Research Training Group CINACS (Cross-modal Interactions in Natural and Artificial Cognitive Systems)

References

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

Ghallab, M.; Nau, D.; and Traverso, P. 2004. *Automated Planning Theory and Practice*. Elsevier Science.

Nau, D. S. 2007. Current trends in automated planning. *AI Magazine* 28(4):43–58.

Off, D., and Zhang, J. 2011. Open-ended domain model for continual forward search HTN planning. In *Proceedings of the ICAPS-2011 Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*.