UMOP 1.2 Software Demonstration

Rune M. Jensen and **Manuela M. Veloso**
Computer Science Department,Carnegie Mellon University,
5000 Forbes Avenue, Pittsburgh,PA 15213-3891, USA
{runej,mmv}@cs.cmu.edu

The UMOP Planning System

UMOP (Jensen 2000; Jensen & Veloso 2000; Jensen, Veloso, & Bowling 2001) is a universal planning system for multi-agent and non-deterministic domains. As input UMOP takes a planning problem described in the Non-deterministic Agent Domain Language (NADL). The NADL problem is translated to a search problem in a finite transition system. The transition system is represented by a conjunctive partitioned transition relation encoded symbolically as a set of reduced ordered Binary Decision Diagrams (BDDs) (Bryant 1986).

Version 1.2 of UMOP includes the three original BDD-based universal planning algorithms: weak, strong, and strong cyclic planning where the actions of the uncontrollable environment agents are abstracted (Cimatti, Roveri, & Traverso 1998a; 1998b; Cimatti *et al.* 2001). An execution of a strong universal plan is guaranteed to reach states covered by the plan and terminate in a goal state after a finite number of steps. An execution of a strong cyclic plan is also guaranteed to reach states covered by the plan and terminate in a goal state, if it terminates. However, a goal state may never be reached due to cycles. An execution of a weak plan may reach states not covered by the plan, it only guarantees that some execution exists that reaches the goal from each state covered by the plan.

function NDP(S_0, G)
  $P \leftarrow \emptyset$; $C \leftarrow G$
  while $S_0 \subseteq C$
    $P_c \leftarrow$ PRECOMP(C)
    if $P_c = \emptyset$ **then return** "no solution exists"
    else
      $P \leftarrow P \cup P_c$
      $C \leftarrow C \cup$ STATES(P_c)
  return P

Figure 1: A generic algorithm for synthesizing non-deterministic plans. S_0 is a set of initial states, while G is a set of goal states.

In addition, UMOP 1.2 includes adversarial versions of the weak and strong-cyclic algorithms where explicit reasoning about the actions of the environment leads to solutions of substantial higher quality (Jensen, Veloso, & Bowling 2001). All of these algorithms iteratively compute a BDD representing a universal plan during a breadth-first search backward from the goal states to the initial states. The search algorithms rely on efficient techniques developed in *symbolic model checking* for computing components of the plan and searching the state space. The algorithm is shown in Figure 1. The set of states S_0 is the possible initial states, while G is the goal states. In each iteration, a precomponent of the plan is generated from the current set of covered states C and added to the plan P. The algorithms only differ by the way the precomponent is defined.

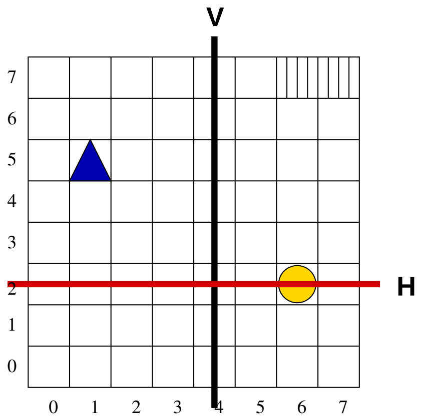

Figure 2: The work cell domain.

Usage

UMOP is implemented in C++/STL under Linux 7.1. The main options of the planner are

```
UMOP -d <dom> -a <alg> -o <plan>
```

where `dom` is an ASCII file containing a planning problem in NADL, `alg` is the name of the planning algorithm the system should apply, and `plan` is the name of a BDD-file UMOP should write the plan to. In addition, UMOP provides options to adjust the BDD package parameters such as

Copyright © 2003, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

cache sizes and dynamic variable ordering. A complete list of options is printed by the -h option.

The software package also includes an executor that can be used to study produced plans. In addition, there is an X-based visual executor for plans generated for the two domains described in this demonstration. The software is open source and can be used for teaching and research purposes. UMOP has been used in a graduate level class on planning, learning and execution for three years at the Computer Science Department of Carnegie Mellon University.

Demonstration

The demonstration will show two applications of the UMOP planning system. The first application is for controlling a factory work cell. The work cell is shown in Figure 2. It consists of two magnetic arms that can slide metal objects from a random position on a grid to a feed belt. The actions of the arms are constrained by rules that guarantee that no collisions happen between objects on the two arms. The purpose of this application is to show that UMOP can compute optimal plans for a multi-agent system by reasoning on the complete space of joint actions. It will first be shown how the problem is described in NADL. The command line options of UMOP will then briefly be described and an example call of the system will be shown. The resulting BDD, representing the universal plan will then be analysed visually via the X-based plan executor.

The second application is a hunter and prey problem on a chess board. The domain is shown in Figure 3. The hunter

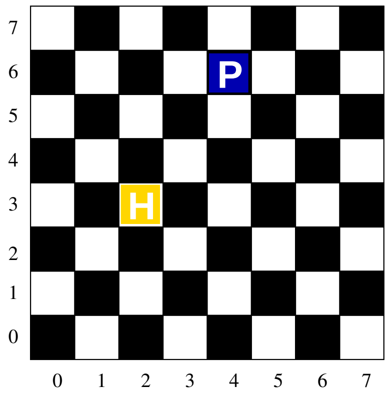

Figure 3: The hunter and prey domain.

and prey move simultaneously in each iteration. The goal is to match the position of the hunter and prey from some random initial position. There are two versions of the domain. In the first, both the hunter and prey can make a Kings move in each iteration. In the second, the hunter can only make a single-step Bishop move. The purpose ofd the domain is to show the difference between the ordinary weak and strong cyclic universal plans and adversarial plans. For the first version of the domain, both a strong cyclic and a strong cyclic adversarial plan exists. The reason is that no matter what strategy the hunter chooses, it will eventually catch the prey. For the second version, only a strong cyclic solution exists. The reason is that the prey has a simple strategy to avoid the hunter. If the hunter starts at a black position, it will stay at black positions. Thus, the prey just needs to move to a white position and stay at white positions. This problem is only realized by the strong cyclic adversarial planning algorithm, not the ordinary strong cyclic algorithm.

The demonstration will present the NADL description of the problem. The two versions of the domains are then visually demonstrated using the X-based executor as above. The demonstration shows a dramatic difference between the quality of ordinary and adversarial universal plans both in terms of goal reachability and expected execution time.

References

Bryant, R. E. 1986. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers* 8:677–691.

Cimatti, A.; Pistore, M.; Roveri, M.; and Traverso, P. 2001. Weak, strong, and strong cyclic planning via symbolic model checking. Technical Report 0104-11, ITC irst, Trento, Italy.

Cimatti, A.; Roveri, M.; and Traverso, P. 1998a. Automatic OBDD-based generation of universal plans in non-deterministic domains. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI'98)*, 875–881. AAAI Press.

Cimatti, A.; Roveri, M.; and Traverso, P. 1998b. Strong planning in non-deterministic domains via model checking. In *Proceedings of the 4th International Conference on Artificial Intelligence Planning System (AIPS'98)*, 36–43. AAAI Press.

Jensen, R. M., and Veloso, M. M. 2000. OBDD-based universal planning for synchronized agents in non-deterministic domains. *Journal of Artificial Intelligence Research* 13:189–226.

Jensen, R. M.; Veloso, M. M.; and Bowling, M. 2001. Optimistic and strong cyclic adversarial planning. In *Pre-proceedings of the 6th European Conference on Planning (ECP'01)*, 265–276.

Jensen, R. M. 2000. The UMOP 1.2 software package. `http://www.cs.cmu.edu/˜runej/umop/umop.html`.