

Controlled Search over Compact State Representations, in Nondeterministic Planning Domains and Beyond

Ugur Kuter and Dana Nau

University of Maryland,
Department of Computer Science and
Institute for Systems Research,
College Park, Maryland 20742, USA

Abstract

Two of the most efficient planners for planning in nondeterministic domains are MBP and ND-SHOP2. MBP achieves its efficiency by using Binary Decision Diagrams (BDDs) to represent sets of states that share some common properties, so it can plan for all of these states simultaneously. ND-SHOP2 achieves its efficiency by using HTN task decomposition to focus the search. In some environments, ND-SHOP2 runs exponentially faster than MBP, and in others the reverse is true. In this paper, we discuss the following:

- We describe how to combine ND-SHOP2's HTNs with MBP's BDDs. Our new planning algorithm, YoYo, performs task decompositions over classes of states that are represented as BDDs. In our experiments, YoYo easily outperformed both MBP and ND-SHOP2, often by several orders of magnitude.
- HTNs are just one of several techniques that are originally developed for classical planning domains and that can be adapted to work in nondeterministic domains. By combining those techniques with a BDD representation, it should be possible to get great speedups just as we did here.
- We discuss how these same ideas can be generalized for use in several other research areas, such as planning with Markov Decision Processes, synthesizing controllers for hybrid systems, and composing Semantic Web Services.

Introduction

An active area of automated-planning research is how to plan in *nondeterministic domains*, in which the actions may have multiple possible outcomes. Planning in nondeterministic domains is much harder than in classical (i.e., deterministic) planning domains: the algorithms must reason about all or most of the possible execution paths, and the sizes of the solution plans may grow exponentially.

Two of the most promising approaches for planning in nondeterministic planning domains are the following:

- *Planning as model checking.* Here, substantial speedups have been achieved (Cimatti *et al.* 2003; Rintanen 2002; Jensen & Veloso 2000) by using Binary Decision Diagrams (BDDs) (Bryant 1992) to compactly represent sets of states that share common properties. Under the right conditions, this approach provides exponential speed-ups

as demonstrated with the MBP planner (Bertoli *et al.* 2001; Cimatti *et al.* 2003).

- *Planning with search control.* Many planning algorithms developed for classical planning domains use search-control techniques that can be generalized to work in nondeterministic domains (Kuter & Nau 2004). These search-control techniques can constrain the planner's search to just a small portion of the search space. For example, the Hierarchical Task Network (HTN) decomposition techniques used in SHOP2 (Nau *et al.* 2003) have been generalized in this fashion, producing a planner called ND-SHOP2 (Kuter & Nau 2004). In some cases, ND-SHOP2 is exponentially faster than MBP, and in some cases the reverse is true.

In this paper, we discuss how to combine the two techniques, and show that the combination of the two works much better than either one alone. Our new planning algorithm, YoYo, does HTN-based task decompositions over BDD-based representations of classes of states. In our experiments, YoYo was never dominated by either MBP or ND-SHOP2, and could easily deal with problem sizes that neither MBP nor ND-SHOP2 could scale up to. Furthermore, YoYo could solve problems about two or three orders of magnitude faster than MBP and ND-SHOP2.

We also discuss several potential generalizations of our ideas. One promising direction is to use other types of search-control mechanisms from classical planning, such as the control rules used in TLplan (Bacchus & Kabanza 2000) and TALplanner (Kvarnström & Doherty 2001). Also, there is good potential for generalizing our ideas to other research areas, such as planning with Markov Decision Processes, synthesizing controllers for hybrid systems, and composing Semantic Web Services.

The YoYo Planner

The YoYo planner (Kuter *et al.* 2005) was developed for planning in nondeterministic domains under the full-observability assumption – that is, the world can be completely observed at runtime. In nondeterministic domains, an action may have multiple outcomes but no probabilities and utilities are associated with them. A planner does not know which outcome will actually occur when an action is executed; so, it generates plans that guarantee to reach the

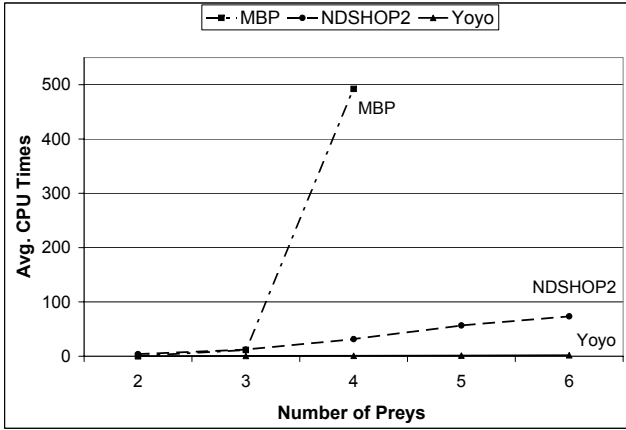


Figure 1: Average running times (in sec.'s) of YoYo, ND-SHOP2, and MBP on hunter-prey problems as a function of the number of prey in a 4×4 grid. MBP could not solve problems with 5 and 6 prey within 40 minutes.

goals, no matter what happens during execution.

YoYo combines task-decomposition techniques as in HTN planning with compact state representations using Binary Decision Diagrams (BDDs) as in planning via Symbolic Model Checking. In YoYo, domain-specific search-control strategies encoded as HTNs constrain the planner’s search to small portions of the state spaces. BDDs, on the other hand, provide a way to compactly represent classes of states as propositional formulas and transform those formulas to perform the search in YoYo more efficiently.

YoYo does HTN decompositions over BDD-based representations of classes of states as follows. In a set S of states represented as a BDD, a possible HTN decomposition of a task t specifies (1) a set of subtasks and (2) a subset S' of S in which the particular decomposition of t is possible. Thus, decomposing a task t in a set S of states represented by a BDD yields two sub-BDDs — one that represents S' and the other that represents the rest of the states $S \setminus S'$ in which other possible decompositions for t must be tried.

YoYo recursively generates BDD-task pairs in the way outlined above. The BDD in each such pair represents a set of states that share some common properties, so that the associated task can be achieved in all of these states simultaneously. If a BDD-task pair is generated such that there is no possible way of achieving the task in the states represented by the BDD then this is a failure point; so, YoYo backtracks and tries other possible HTN decompositions. Otherwise, YoYo continues with planning until the tasks in all of the generated BDD-task pairs correspond to primitive actions. This terminates the planning process successfully and the set of all generated BDD-action pairs is a solution plan.

We compared YoYo with the two state-of-the-art planners in nondeterministic domains, namely ND-SHOP2 and MBP. ND-SHOP2 is an HTN planner based on explicit state representations (Kuter & Nau 2004). MBP implements BDD-based planning algorithms (Bertoli *et al.* 2001) but cannot use search control as in ND-SHOP2 or YoYo.

For all our experiments, we used a 900MHz laptop with

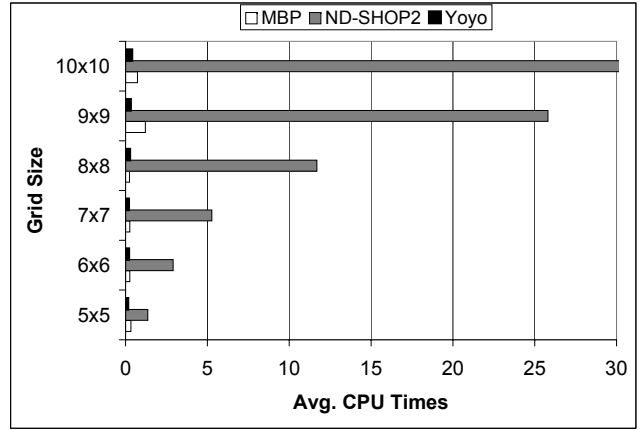


Figure 2: Average running times (in sec.'s) of YoYo, ND-SHOP2, and MBP on hunter-prey problems as a function of the grid size, with one prey. ND-SHOP2 could not solve problems with larger grids due to memory overflows.

256MB memory, running Linux Fedora Core 2. We set the time limit for the planners as 40 minutes. Our experimental testbed was a simple variant of a *pursuit-evasion game* described in (Koenig & Simmons 1995). In this domain, there is a hunter and one or more prey in a grid world. The goal for the hunter is to catch all the prey. The nondeterminism for the hunter is introduced through the actions of the prey: at any time, the prey can take a move in the world independent from the hunter’s move. For a detailed description of our experimental setup, see (Kuter *et al.* 2005).

Figure 1 shows the results on one experimental set, in which we used a 4×4 grid world and varied the number of prey in the world. Each data point is the average of 20 randomly-generated problems. We encoded the following search-control strategy for ND-SHOP2 and YoYo using HTNs: “choose a prey and chase it until it is caught, then choose another prey, and so on, until all of the prey are caught.” This prunes away large portions of the state space during planning since, when the hunter is chasing a prey, it need not reason about the locations of the other prey.

As shown in Figure 1, MBP’s running times grew exponentially faster than those of YoYo since MBP cannot exploit search-control strategies and its BDD-based representations were not useful to prune irrelevant portions of the search space. YoYo also easily outperformed ND-SHOP2 by taking advantage of BDD-based representations. ND-SHOP2, in turn, outperformed MBP using the search-control strategy described above, although it does not exploit BDDs.

In another experimental set (see Figure 2), we had only one prey and varied the size of the grid. Similarly as before, we randomly generated 20 problems and run our algorithms on those problems. This setting does not admit good search-control strategies: all we can do is to “look at the prey and move towards it.” In the absence of good strategies ND-SHOP2 did not perform well in large problems due to its explicit state representations. YoYo, on the other hand, was exponentially better than ND-SHOP2 even without good strategies due to its BDD-based state representa-

tions. It also outperformed MBP since the above strategy helps to prune some portions of the state space.

Generalized Search Control Techniques for Nondeterministic Planning

In (Kuter & Nau 2004), we described a way to take a class of planners that were originally developed for classical (i.e., deterministic) domains and generalize them to work in nondeterministic domains. ND-SHOP2 is a generalization of SHOP2 (Nau *et al.* 2003) we developed based on that work, along with generalizations of TLplan (Bacchus & Kabanza 2000) and TALplanner (Kvarnström & Doherty 2001).

In YoYo, we used HTNs to perform a controlled search over BDD-based compact representations for classes of states in nondeterministic domains. A similar approach can be developed to use search-control rules as in TLplan and TALplanner, specified in modal temporal logics (TL). By combining those techniques with a BDD representation, we should get great speedups just as we got with YoYo. Furthermore, it should be possible to get even greater speedups by developing techniques that use TL formulas and HTNs together for controlled search over BDDs. (Bacchus & Kabanza 2000) points out that the two techniques for specifying search-control strategies (i.e., HTNs and TL formulas) are useful in different situations; thus, this is a promising approach for broadening the range of problems we could solve efficiently in nondeterministic domains.

An important consideration with using domain-specific search control based on HTNs or TL formulas under nondeterminism is that, in complex domains, it may not be possible to compile a control strategy for the entire domain. For example, there may be many possible outcomes of an action, all of which may not be anticipated in advance. However, controlled search over BDD-based state representations will yield significant speed-ups even if we can use them for a part of a planning domain. Furthermore, in those parts of a planning domain for which we do not have any search-control strategies, we can use the existing algorithms to generate a partial solution that compensates the gap in the search-control strategy. As an example, if we have an incomplete set of HTNs for YoYo, we can use MBP for generating the type of a partial solution mentioned above. This will allow us to use our new planning algorithms in complex domains where compiling search-control strategies is difficult.

YoYo'ing Beyond

In this section, we discuss how our ideas in YoYo and some of the generalizations discussed in the previous section can be applied to other research areas; in particular, planning with Markov Decision Processes, synthesizing controllers for hybrid systems, and composing Semantic Web Services.

Planning with Markov Decision Processes

In planning domains where actions have probabilistic outcomes, the primary approach is based on Markov Decision Processes (MDPs); see (Boutilier, Dean, & Hanks 1999) for a survey. The primary difference between nondeterministic planning problems and MDP planning problems is that

the latter has rewards associated with the states, probabilities and costs associated with the state transitions, and the goal is to generate a plan that optimizes a utility function.

Many MDP solution methods have been developed to reason over sets of states during planning. Examples include (Dearden & Boutilier 1997; Feng & Hansen 2002; Lane & Kaelbling 2002). Among others, (Feng & Hansen 2002) describes a way for searching over BDD-based representations in MDPs but no search-control strategies as in YoYo. A form of search control for MDPs has been used in Reinforcement Learning (Parr 1998; Dietterich 2000). This approach is based on hierarchical abstractions that are somewhat similar to HTN planning. The hierarchical abstraction of an MDP is analogous to an instance of the decomposition tree that an HTN planner might generate. However, the abstractions must be supplied in advance by the user, rather than being generated on-the-fly by an HTN planner.

In YoYo, much of the computational machinery was for correctly handling the possible state transitions induced by the nondeterministic actions — a characteristic that nondeterministic planning shares with MDP planning. This suggests that it should be possible to generalize our approach in YoYo for solving MDPs. This requires developing new techniques for handling the probabilities, rewards, and costs in the state-transition operations dictated by the transformations over the BDDs and the search-control strategies.

Synthesizing Controllers for Hybrid Systems

Hybrid systems are dynamical systems that have both continuous- and discrete-valued state variables (Lygeros, Tomlin, & Sastry 1999; Tomlin *et al.* 2003). Examples of such systems include robotic systems, tactical fighter aircrafts, intelligent vehicle/highway systems, and flight control systems. The inherent uncertainties and interactions between the discrete and continuous components make it very hard to synthesize optimal controllers for such systems.

Typical existing approaches for hybrid systems model the discrete and continuous components of a hybrid system independently and generate controllers using techniques that can exploit the interactions between the two separate models. Examples include the use of timed automata (Alur & Dill 1994), game-theoretic approaches (Tomlin, Lygeros, & Sastry 2000), linear hybrid automata (Shakernia, Pappas, & Sastry 2000), and linear programming and simulation techniques (Hauskrecht & Kveton 2004).

Combinations of compact state representations and search-control strategies can also be used in synthesizing controllers for hybrid systems in order to abstract away from the continuous parts of the state space, which is usually infinite due to the continuous-valued state variables in those systems. This approach primarily involves developing algorithms similar to YoYo that decompose the system models into smaller and smaller models until we generate a solution controller. Our work on YoYo suggests that this approach will compare favorably with the existing techniques.

Semantic Web Service Composition

Semantic Web services are functionalities on the Web that are designed to be composed, i.e., combined in workflows

of varying complexity to provide a functionality that none of the component services could provide alone.

In (Kuter *et al.* 2004), we described an HTN planning algorithm for composing Web Services. This algorithm uses HTNs to describe the operational semantics of Web Services and performs the composition process using HTN decompositions over these descriptions. Another promising approach for composing Web Services is to use BDDs to represent service descriptions and generate compositions by reasoning over them as described in (Traverso & Pistore 2004).

It should be possible to combine the above approaches for composing Web Services in a single framework similar to YoYo. For the reasons discussed in this paper, this approach should yield new service-composition algorithms that are significantly more efficient than the existing ones.

Conclusions

We have described how to combine a BDD state representation with HTN search-control to achieve efficient planning in fully observable nondeterministic domains (Kuter *et al.* 2005). Our experiments show that the combination has large advantages in speed, memory usage, and scalability. Promising areas for future research include (1) the use of other search-control techniques such as TLplan's or TALplanner's control rules, and (2) applications of our approach to other areas, including planning with MDPs, synthesizing controllers for hybrid systems, and composing Semantic Web Services.

Acknowledgments. This work was based in part on (Kuter *et al.* 2005). Figures 1 and 2 are © AAI, 2005, and are used with permission. This work was supported in part by ISLE contract 0508268818 (subcontract to DARPA's Transfer Learning program), UC Berkeley contract SA451832441 (subcontract to DARPA's REAL program), NSF grant IIS0412812, and the FIRB-MIUR project RBNE0195k5, "Knowledge Level Automated Software Engineering." The opinions expressed in this paper are those of authors and do not necessarily reflect the opinions of the funders.

References

- Alur, R., and Dill, D. 1994. A Theory of Timed Automata. *Theoretical Computer Science* 126:183–235.
- Bacchus, F., and Kabanza, F. 2000. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence* 116(1-2):123–191.
- Bertoli, P.; Cimatti, A.; Pistore, M.; Roveri, M.; and Traverso, P. 2001. MBP: a model based planner. In *IJCAI-2001 Workshop on Planning under Uncertainty and Incomplete Information*.
- Boutilier, C.; Dean, T. L.; and Hanks, S. 1999. Decision-theoretic planning: Structural assumptions and computational leverage. *JAIR* 11:1–94.
- Bryant, R. E. 1992. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Computing Surveys* 24(3):293–318.
- Cimatti, A.; Pistore, M.; Roveri, M.; and Traverso, P. 2003. Weak, strong, and strong cyclic planning via symbolic model checking. *Artificial Intelligence* 147(1-2):35–84.
- Dearden, R., and Boutilier, C. 1997. Abstraction and approximate decision-theoretic planning. *Artificial Intelligence* 89(1-2):219–283.
- Dietterich, T. G. 2000. Hierarchical reinforcement learning with the MAXQ value function decomposition. *JAIR* 13:227–303.
- Feng, Z., and Hansen, E. 2002. Symbolic Heuristic Search for Factored Markov Decision Processes. In *AAAI-2002*.
- Hauskrecht, M., and Kveton, B. 2004. Linear Program Approximations for Factored Continuous-State MDPs. In *NIPS-2004*.
- Jensen, R., and Veloso, M. M. 2000. OBDD-based universal planning for synchronized agents in non-deterministic domains. *JAIR* 13:189–226.
- Koenig, S., and Simmons, R. G. 1995. Real-time search in non-deterministic domains. In *IJCAI-1995*.
- Kuter, U., and Nau, D. 2004. Forward-chaining planning in nondeterministic domains. In *AAAI-2004*.
- Kuter, U.; Sirin, E.; Nau, D.; Parsia, B.; and Hendler, J. 2004. Information Gathering during Planning for Web Services Composition. In *ISWC-2004*.
- Kuter, U.; Nau, D.; Pistore, M.; and Traverso, P. 2005. A hierarchical task-network planner based on symbolic model checking. In *ICAPS-2005*.
- Kvarnström, J., and Doherty, P. 2001. TALplanner: A temporal logic based forward chaining planner. *Annals of Mathematics and Artificial Intelligence* 30:119–169.
- Lane, T., and Kaelbling, L. 2002. Nearly deterministic abstractions of Markov decision processes. In *AAAI-2002*.
- Lygeros, J.; Tomlin, C.; and Sastry, S. 1999. Controllers for Reachability Specifications for Hybrid Systems. *Automatica* 35(3).
- Nau, D.; Au, T.-C.; Ilghami, O.; Kuter, U.; Murdock, W.; Wu, D.; and Yaman, F. 2003. SHOP2: An HTN planning system. *JAIR* 20:379–404.
- Parr, R. 1998. Hierarchical control and learning for markov decision processes. PhD thesis, UC Berkeley.
- Rintanen, J. 2002. Backward plan construction for planning as search in belief space. In *AIPS-2002*.
- Shakernia, O.; Pappas, G.; and Sastry, S. 2000. Decidable Controller Synthesis for Classes of Linear Systems. In *Hybrid Systems: Computation and Control (LNCS 1790)*.
- Tomlin, C.; Mitchell, I.; Bayen, A.; and Oishi, M. 2003. Computational Techniques for the Verification and Control of Hybrid Systems. *IEEE Proceedings* 91(7).
- Tomlin, C.; Lygeros, J.; and Sastry, S. 2000. A Game Theoretic Approach to Controller Design for Hybrid Systems. *IEEE Proceedings* 88(7).
- Traverso, P., and Pistore, M. 2004. Automated Composition of Semantic Web Services into Executable Processes. In *ISWC-2004*.