Applications of SHOP and SHOP2

Dana Nau
University of Maryland

Tsz-Chiu Au
University of Maryland

Okhtay Ilghami
University of Maryland

Ugur Kuter
University of Maryland

Héctor Muñoz-Avila
Lehigh University

J. William Murdock
IBM Research

Dan Wu
University of Maryland

Fusun Yaman
University of Maryland

CS-TR-4604, UMIACS-TR-2004-46
June 25, 2004

Abstract

SHOP and SHOP2 are HTN planning systems that were designed with two goals in mind: to investigate some research issues in automated planning, and to provide some simple, practical planning tools. They are available as freeware, and have developed an active base of users in government laboratories, industrial R&D projects, and academic settings. This paper summarizes how SHOP and SHOP2 work, describes some of the applications that we and others have developed for them, and discusses directions for future research and enhancements.

1 Introduction

The SHOP and SHOP2 planning systems were designed with two goals in mind: to investigate some research issues in automated planning, and to provide some simple, practical planning tools. They have been successful in both respects.

SHOP and SHOP2 are available as open-source software, and have been downloaded thousands of times. Their practical utility is shown by the emergence of an active set of users, which include government laboratories, industrial R&D projects, and academic settings. As an example of their research impact, SHOP2 received one of the top four awards in the 2002 International Planning Competition.

One reason for the success of SHOP and SHOP2 is their use of Hierarchical Task Networks (HTNs). HTN planning is done by applying *HTN methods*, which basically are forms that describe how to decompose tasks into subtasks. HTN methods can be used to describe the "standard operating procedures" that one would normally use to perform tasks in some domain; thus they often correspond well to the way that users think about problems.

Another reason for the success of SHOP and SHOP2 is their use of a search-control strategy called *ordered task decomposition*, which reduces the complexity of reasoning by eliminating a great deal of uncertainty about the world. Ordered task decomposition makes it easy to incorporate a great deal of expressive power into the planning system: for example, SHOP and SHOP2 can do complex inferential reasoning, mixed symbolic/numeric computations, and call user-supplied subroutines.

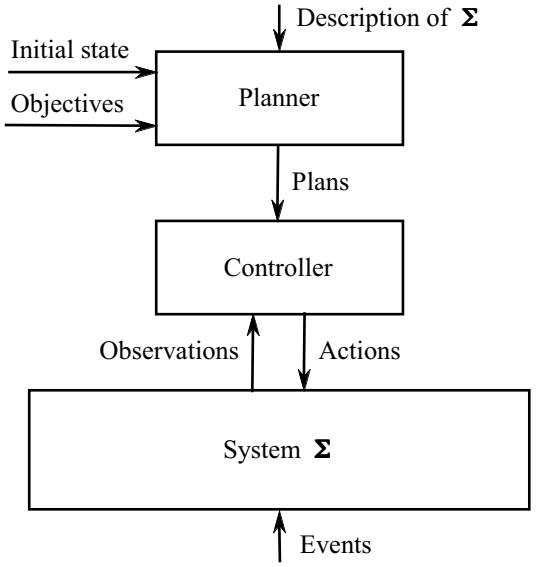

Figure 1: A simple conceptual model for planning.

This paper is organized as follows. Section 2 gives an informal description of HTN planning. Section 3 gives an overview of SHOP and SHOP2, and Section 4 summarizes some of the projects in which they have been used. Section 5 contains concluding remarks, and describes our ongoing and future work.

2 Background

Automated Planning. In general, the purpose of an automated planning system is to generate a plan or policy that a plan executor can execute in order to achieve some set of goals or objectives. Most planning research has focused on *offline* planning (see Figure 1), in which the entire plan is formulated before the plan executor begins executing it. Thus at planning time, no direct information is available to the planner about a plan's success or failure—instead, the planner must reason about whether the plan will (or is likely to) succeed or fail. Automated planning systems can be classified roughly into three types:

- *Domain-specific planning systems*, where the planning domain is known beforehand and the system is designed specifically to reason about plans in that domain. Several of the most successful planning systems are of this type (e.g., [12]).
- *Domain-independent planning systems*, which are designed to work in any domain within some large class of domains (e.g., the well-known *classical* planning domains [3]), provided that the input includes definitions of the basic actions in the domain. Domain-independent systems have been developed that work quite well in abstract domains—but getting them to work well in domains of practical importance has been an elusive goal.
- *Domain-configurable planning systems.* Here, the planning engine is domain-independent, and the domain description includes both the basic actions (like in domain-independent planning) and also some information about how those actions should or may be combined in order to solve planning problems.

Much more work has been done on automated planning than we can describe here, and we refer the reader to [3] for details.

HTN Planning. For domain-specific and domain-configurable planning, one of the best-known approaches is *HTN planning*, in which the planning system formulates a plan by decomposing *tasks* (symbolic representations of activities to be performed) into smaller and smaller subtasks until *primitive* tasks are reached that can be performed directly. The basic idea was developed in the mid-70s [10, 13], and the formal underpinnings were developed in the mid-90s [2].

HTN-planning research has been much more application-oriented than most other AI-planning research. Most of the domain-configurable systems (e.g., O-Plan [14], SIPE-2 [15], SHOP [8], and SHOP2 [7]) have been used in application development, and domain-specific HTN planning systems have been built for several application domains (e.g., [12]).

An HTN planning problem consists of the following: the *initial state* (a symbolic representation of the state of the world at the time that the plan executor will begin executing its plan), the *initial task network* (a set of tasks to be performed, along with some constraints that must be satisfied), and a *domain description* that contains the following:

- A set of *planning operators* that describe various kinds of actions that the plan executor can perform. Each operator may have a set of preconditions that must be true in the state in which the operator is to be executed, and a set of effects that will occur when the operator is executed. Each of the possible actions is an operator instance, produced by assigning values to an operator's parameters.
- A set of *methods* that describe various possible ways of decomposing tasks into subtasks. These are the "standard operating procedures" that one would normally use to perform tasks in the domain. Each method may have a set of constraints that must be satisfied in order to be applicable.
- Optionally, various other information such as definitions of auxiliary functions and definitions of axioms for inferring conditions that are not mentioned explicitly in states of the world.

Planning is done as follows. For each nonprimitive task, the planner chooses an applicable method and instantiates it to decompose the task into subtasks. For each primitive task, the planner chooses an applicable operator and instantiates it to produce an action. If all of the constraints are satisfied, then the planner has found a solution plan; otherwise the planning system will need to backtrack and try other methods or other instantiations.

Example. Figure 2 gives a pseudocode representation of an extremely simple planning domain in which there are two ways to travel from one location to another: by foot and by taxi. These are represented by two methods: travel-by-foot and travel-by-taxi. The travel-by-foot method has one constraint: a precondition saying that the distance from the starting point to the destination must be less than or equal to 2 miles. If the method is applicable, it decomposes the task into a single subtask: walk to the park. The travel-by-taxi method has one constraint, which is also a precondition: the traveler must have enough cash to pay the taxi driver. If the method is applicable, it decomposes the task into three subtasks: call a taxi, ride to the park, and pay the driver. All of the subtasks are primitive, i.e., the traveler is expected to know how to accomplish them directly.

Now, suppose that in the initial state, I am at home, I have $20, and I want to travel to a park that is 8 miles away. To plan how to travel to the park (see Figure 3), first I try to use the travel-by-foot method, but this method is not applicable because the park is more than 2 miles

method travel-by-foot
  precond: $distance(x, y) \leq 2$
  task: travel(a, x, y)
  subtasks: walk(a, x, y)

method travel-by-taxi
  task: travel(a, x, y)
  precond: $cash(a) \geq 1.5 + 0.5 \times distance(x, y)$
  subtasks: call-taxi$(a, x) \longrightarrow$ ride$(a, x, y) \longrightarrow$ pay-driver(a, x, y)

operator walk
  precond: $location(a) = x$
  effects: $location(a) \leftarrow y$

operator call-taxi(a, x)
  effects: $location(taxi) \leftarrow x$

operator ride-taxi(a, x)
  precond: $location(taxi) = x$, $location(a) = x$
  effects: $location(taxi) \leftarrow y$, $location(a) \leftarrow y$

operator pay-driver(a, x, y)
  precond: $cash(a) \geq 1.5 + 0.5 \times distance(x, y)$
  effects: $cash(a) \leftarrow cash(a) - 1.5 + 0.5 \times distance(x, y)$

Figure 2: Pseudocode representation of a simple travel-planning domain. Left-arrows denote assignments of values to state-variables; right-arrows are ordering constraints.

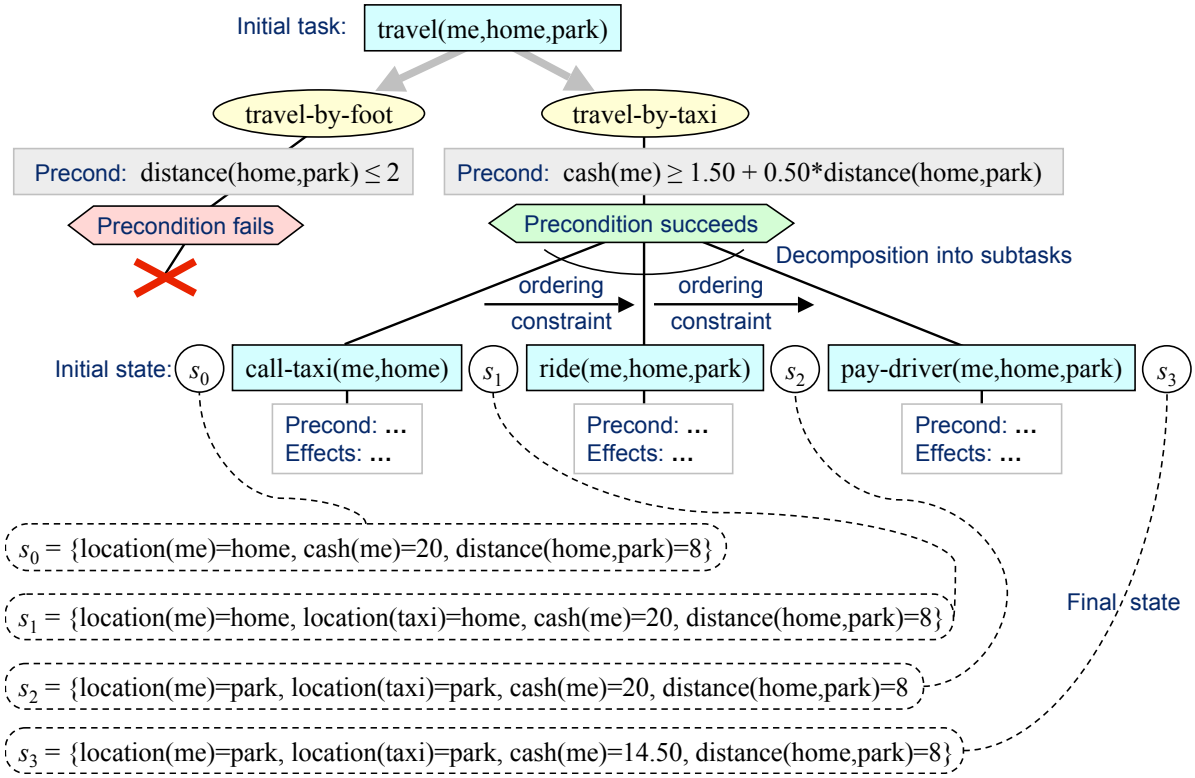

Figure 3: Solving a planning problem in the travel-planning domain.

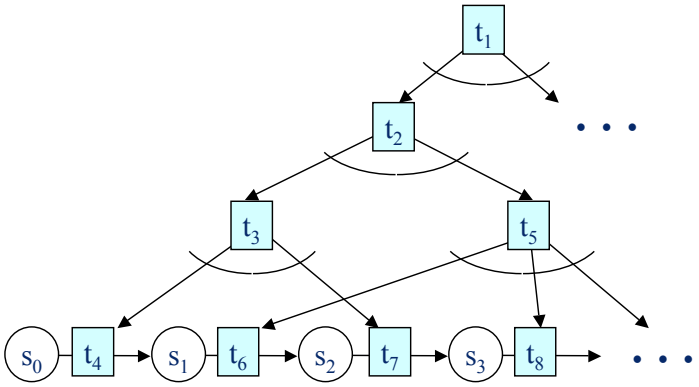

Figure 4: An example of ordered task decomposition. The subscripts show the order in which the tasks are decomposed. In this example, tasks t_4, t_6, t_7, t_8 are primitive, i.e., they correspond to planning operators.

away. Next, I try to use the travel-by-taxi method. Its precondition is satisfied, so the method produces a sequence of three subtasks, with a constraint saying they are to be performed in the following order: (1) call a taxi to my home, (2) ride in it to the park, and (3) pay the driver $5.50. The subtasks all are primitive, i.e., each of them corresponds to an action. The first action has no preconditions, so it is applicable and produces a state s_1 that is identical to the initial state except that $location(taxi) = home$. This state satisfies the preconditions of the second action. The second action produces a state in which the precondition of the third action is satisfied, so I have a solution plan. After execution of this plan, the final state will be as shown in the figure.

3 SHOP and SHOP2

HTN planning is basically a trial-and-error search: the planner may have to try many different possibilities before finding a plan that works. In any trial-and-error search, one of the most important questions is what kind of search-control strategy to use.

SHOP and SHOP2 use a search-control strategy called *ordered task decomposition*: they choose to decompose tasks into subtasks in the same order as the order in which the tasks are supposed to be accomplished. As a consequence, SHOP and SHOP2 generate the steps of each plan in the same order that the plan executor will execute those steps (see Figure 4), so they know the current state at each step of the planning process. This reduces the complexity of reasoning by eliminating a great deal of uncertainty about the world, thereby making it easy to incorporate substantial expressive power into the planning system, such as the auxiliary functions and axioms mentioned earlier.

The primary difference between SHOP and SHOP2 is that SHOP requires a strict linear ordering on subtasks and does not allow them to be interleaved. In contrast, SHOP2 does not impose these requirements. For example, in Figure 4, the subtasks of task t_3 and task t_5 are interleaved; this can occur in SHOP2 but not in SHOP. As a result, some planning domains that would be rather cumbersome to describe in SHOP can be described more easily in SHOP2.

In April 2002, the SHOP2 planning system achieved high visibility because of its performance in the 2002 International Planning Competition, where it received one of the top four awards.[1]

[1]There were two awards for "distinguished performance" and two for "distinguished performance of the first

SHOP2 was one of the three fastest planners in the competition: it was able to solve planning problems many times faster and many times more complicated than those solved by most of the other systems. In addition, SHOP2 solved 899 out of 904 problems, more than any of the other systems.

Both SHOP and SHOP2 are open-source software, and may be downloaded at ⟨`http://www.cs.umd.edu/projects/shop`⟩. SHOP is available in both Common Lisp and Java. SHOP2 is only available in Common Lisp, but it includes an interface for interoperating with programs written in other languages, and we are currently implementing a Java version.

4 Applications of SHOP and SHOP2

SHOP and SHOP2 have been downloaded thousands of times,[2] and have developed a significant user base, including users from government laboratories, industries, and universities. Based on information given to us by some of the users on our mailing list, here are descriptions of a few of the projects in which SHOP and SHOP2 have been used.

4.1 Projects in Government Laboratories

Evacuation Planning (Naval Research Laboratory, Washington, DC)

The HICAP system at the US Naval Research Laboratory (NRL) is a tool for helping experienced human planners to develop evacuation plans, i.e., plans to evacuate humans whose lives are in danger. Evacuation planning must be done by a human expert or under the supervision of a human expert: it is unrealistic to expect that a planning system could produce good plans by itself, and flawed evacuation plans could yield dire consequences. For this reason, the top level of HICAP is a plan editor with which users can edit tasks manually and can use a planning system to interactively refine plans.

Only part of the knowledge necessary for evacuation planning can be formalized sufficiently for automated planning. In general, there will be an incomplete domain description, in the form of standard requirements and operating procedures—but these are not sufficient for deriving detailed plans. For that, knowledge about previous experiences is essential. Thus, HICAP's planning module includes both generative and case-based planning. The generative component is provided by SHOP. The case-based component, NaCoDAE, works by retrieving fragments of plans from previous evacuations. Within HICAP, NaCoDAE and SHOP are integrated quite tightly: each is capable of using the other to decompose tasks into subtasks. For more information about HICAP, see ⟨`http://www.aic.nrl.navy.mil:80/hicap`⟩.

Evaluating Terrorist Threats (Naval Research Laboratory, Washington, DC)

The purpose of NRL's AHEAD project is to help intelligence analysts understand and evaluate hypotheses about terrorist threats. Given a hypothesis, the AHEAD system uses analogical retrieval to obtain a model of the hostile activity most closely related to the hypothesis. This model is encoded as an HTN domain description for SHOP2 in which individual actions are annotated

order;" SHOP2 received one of the former. For more information about the competition, see ⟨`http://planning.cis.strath.ac.uk/competition`⟩.

[2] As of June 24, the log of downloads from our web site shows 1783 downloads, but this does not include the number of times that users have downloaded directly from our ftp server rather than going through our web site. We imagine the total number of downloads is above 2000.

with additional explanatory information about their function. AHEAD invokes SHOP2 using this domain description to produce a plan that is compatible with the hypothesis. As each operator is added to the plan, SHOP2 queries an external evidence database to determine whether the evidence is consistent with that operator. When the evidence is consistent with the operator, AHEAD generates an argument in favor of the hypothesis; when the evidence is inconsistent, AHEAD generates a counterargument. The resulting structured argument is presented in a browsable user interface. HTN planning is particularly well-suited to this process because HTNs organize behavior into meaningful components at multiple levels of abstraction thus enabling coherent, structured argumentation. For more information about AHEAD, see ⟨`http://www.nrl.navy.mil/aic/iss/ida/projects/ahead/AHEAD.php`⟩.

Fighting Forest Fires (LAAS/CNRS, Toulouse, France)

The European Union's COMETS project focuses on the development of unmanned aerial vehicle (UAV) control techniques for detection and monitoring of forest fires. As part of this project, researchers at LAAS, a government research laboratory in Toulouse, France, are developing a distributed architecture in which each UAV will contain a generic "decisional node" consisting of a supervisor and a planner.

Within each decisional node, they are using SHOP2 as the symbolic planner: they exploit SHOP2's forward-chaining capability to integrate its planning activity with specialized software for estimating the costs, time, etc., for basic UAV operations. In order to perform temporal reasoning, they are using the same time-stamping technique we developed for temporal planning with SHOP2 in the 2002 International Planning Competition [7]. The researchers anticipate that they soon will have simulation results and will be able to run experiments using LAAS's blimp, *Karma*. More information about the project is available at ⟨`http://www.comets-uavs.org`⟩.

Software Systems Integration (NIST, Gaithersburg, MD)

NIST (National Institute of Standards and Technology) is using SHOP2 in a project whose goal is to automate tasks of software systems integration. So far, the particular example they have used is based on General Motors' ebXML-based "bulk rental car buying" interfaces. These interfaces allow a buyer to search for cars of a particular make, model and year, and purchase them. But if a buyer wants to get a summary of various cars at various locations (e.g., in order to minimize costs), the interface makes it difficult to do this. NIST's code reads the seller's ebXML BPSS (business process specification schema) and produces a SHOP2 planning problem in which SHOP determines the sequence of transactions against the seller's interfaces to achieve the buyer's objective. For further information about the project, see ⟨`http://www.mel.nist.gov/proj/mee.htm`⟩.

4.2 Industry Projects

Controlling Multiple UAVs (SIFT, Minneapolis, MN)

SIFT, LLC is using a modified version of the SHOP2 planner in a UAV control system in their PVACS project, with funding from DARPA through an SBIR contract. SIFT's "Playbook" control system allows time-pressured users, who are not UAV operators, to request reconnaissance missions using high-level tasking commands, modeled on the way people delegate tasks to human subordinates. The SIFT Playbook supports interactions through both PDA and desktop/laptop interfaces. The Playbook translates users' brief, general commands into very specific control actions suitable for execution. The Playbook's Executive provides high-level closed-loop monitoring and

implementation of the Playbook's plans, controlling multiple UAVs through the Variable Autonomy Control System (VACS) Ground Control Station (GCS), developed by Geneva Aerospace, Inc. The Playbook currently operates these UAVs in a high-fidelity simulation environment, but the interface it uses to control the simulated UAVs through the VACS GCS is the same as the one used to direct VACS UAVs in real flight operations.

The modified SHOP2 planner plays a key role in SIFT's Playbook, translating the user's high level task specifications into a sequence of commands that can be executed by UAVs. SIFT's plan library contains tasks for multiple reconnaissance missions, for both rotorcraft and fixed-wing UAVs. Robert Goldman at SIFT has developed an augmented version of SHOP2 that generates temporal plans including durative actions, and provides more knowledge-engineering and debugging support. For further information, see ⟨`http://www.sift.info/English/projects/PVACS.ppt`⟩.

Evaluation of Enemy Threats (Lockheed Martin ATL, Cherry Hill, NJ)

Lockheed Martin Advanced Technology Laboratories, in collaboration with the Army Research Laboratory, is using SHOP in a project that attempts to evaluate possible enemy threats. They are using SHOP to decompose higher level tasks such as 'attack blue-convoy' into sequences of operations such as 'move red-tank1 to location2, ..., fire red-tank1 at blue-convoy.' Due to their confidentiality restrictions, they were unable to tell us any further details.

Location-Based Services (Sony Electronics, San Jose, CA)

Sony Electronics Incorporated has used SHOP in a project aimed at developing mobile GIS devices to help people plan errands that take them to different geographical locations. Due to Sony's confidentiality requirements, they were unable to tell us any further details.

Material Selection for Manufacturing (Infocraft Ltd., Sri Lanka)

Infocraft Ltd. ⟨`http://www.infocraft.lk`⟩ is developing a system that uses SHOP2 for material selection in continuous-process manufacturing: specifically, the production of activated carbon from charcoal using a discrete set of continuous manufacturing processes. The desired properties of the carbon (specifically its grade size and adsorption level) will vary from one run to another, as will the characteristics of different supplies of charcoal. The objective of the project is to use SHOP2 to select which supplies of charcoal will most reliably produce activated carbon with a desired set of properties. SHOP2's abilities to do numerical and axiomatic reasoning are essential for this project: adsorption levels are represented as real numbers, and grade sizes are represented as normal distributions.

4.3 University Projects

Automated Composition of Web Services (University of Maryland)

Web services are Web accessible, loosely coupled chunks of functionality with an interface described in a machine readable format. Web services are designed to be *composed*, that is, combined in workflows of varying complexity to provide functionality that none of the component services could provide alone.

In the OWL-S (formerly DAML-S) language for semantic markup of web services, services can be described as complex or atomic processes with preconditions and effects. This makes it possible to translate the OWL-S process-model constructs directly to SHOP2 methods and operators, and

we have developed an algorithm to do so. This means that SHOP2 can be used to solve service composition problems, by telling SHOP2 to find a plan for the task that is the translation of a composite process [16, 11].

Project Planning (Lehigh University)

The SHOP/CCBR system is a tool developed at the Lehigh University for investigating the use of HTN planning techniques to support project management. The SHOP/CCBR system is a straightforward extension of SHOP that uses *cases* to decompose tasks. Cases are similar in structure to methods, the main difference being that cases include preference information for use in ranking applicable cases. SHOP/CCBR uses a communication module to interact with Microsoft Project, a commercial tool for project management. This allows displaying the HTN decompositions generated with SHOP's hierarchical planning algorithm in Microsoft Project. The on-going work involves developing algorithms to capture cases automatically from user interactions with Microsoft Project.

Statistical Goal Recognition in Agent Systems (University of Rochester)

For an agent to perform effectively in a multi-agent environment, an important task is *goal recognition*, i.e., inferring the goals of other agents. Researchers at the University of Rochester are developing a statistical approach to goal recognition using machine-learning techniques. To do the learning requires a labeled "plan corpus" of plans and their associated goals. They are using SHOP2 to generate such plan corpora stochastically. For this purpose, they are using a modified version of SHOP2 that makes random choices at every point where more than one possible decision is available to SHOP2. For more information about the project, see ⟨`http://www.cs.rochester.edu/research/cisd/projects/goalrec`⟩.

Additional University Projects

Worldwide, SHOP and SHOP2 have been used in many more college and university projects than we can mention, but here are some notes about a few of them.

- Drexel University regularly uses SHOP and SHOP2 in their Introductory AI class in order to teach planning, and in their Knowledge-Based Agents course to do agent reasoning, service composition, etc.
- At the National University of Colombia in Medellin, Colombia, a system is being developed that uses SHOP2 to automatically create virtual courses from existing educational material.
- At the Technical University of Cluj-Napoca in Romania, SHOP has been used for an e-commerce aplication, to build plans for bidding in a modified version of the Trading Agent Competition.
- At Trinity College Dublin, SHOP2 is being used in a web-service composition project somewhat similar to ours.
- Researchers in the Aerospace Engineering Department at the University of Maryland have just begun a project in which they are using SHOP2 as the planning component in an architecture that combines task planning, real-time scheduling, and motion/trajectory planning.
- At Villanova University, SHOP2 has been used in a mock spacecraft-mission scenario, to study how the density, distribution and overall layout of environment obstacles can be used to compute and predict the best optimization technique to use within SHOP2.

5 Concluding Remarks

We have been pleasantly surprised at the extent to which people have begun using SHOP and SHOP2 in their research and development projects. We believe that this has come about for several reasons:

- SHOP and SHOP2 are based on HTN decomposition. The decomposition of tasks into subtasks seems to correspond well to the way in which users think about how to generate plans.
- Unlike most other automated-planning systems, SHOP and SHOP2 plan for tasks in the same order that the tasks will be executed. This removes a great deal of uncertainty at planning time, which makes it easier to write write complex domain descriptions.
- SHOP and SHOP2 are available as open-source software. This has made it easy for users to find and fix bugs, and to adapt the software for their own purposes.

The success of SHOP and SHOP2 has given us many ideas for improvements and extensions. We now describe some of our ongoing and future work on those topics.

5.1 Automated Learning of Planning Domains

A great challenge in using any planning system to solve real-world problems is the difficulty of acquiring the domain knowledge that the system will need. We are working on ways to address part of this problem by having the planning system *learn* the HTN methods incrementally under supervision of an expert.

We have developed a general formal framework for learning HTN methods, and a supervised learning algorithm, named *CaMeL*, based on this formalism [4]. We have developed theoretical results about CaMeL's soundness, completeness, and convergence properties, and have done experimental studies of its speed of convergence under different conditions. The experimental results suggest that CaMeL may potentially to be useful in real-world applications.

5.2 Compiling Planning Domains

A domain-configurable planner may be viewed as an interpreter of its domain-description language: given a domain description D and a planning problem P, the planner invokes the methods and operators of D interpretively on P. An alternative approach is to write a *compiler* for the domain description language: the input to the compiler is a domain description D, and the output is a domain-specific planning program for D, that can be run directly on any planning problem P in D.

The advantages of such a *planner compilation* approach are analogous to the advantages that compilationhas over interpretation in conventional programming languages. By compiling domain descriptions directly into low-level executable code, we can do implementation-level optimizations that are not otherwise possible and have not been explored in previous research on AI planning. These optimizations can be coupled with other speed-up techniques (e.g., domain analysis and other automated domain information synthesis techniques) in order to obtain additional speedups.

We are developing JSHOP2, a Java implementation of SHOP2 that uses this domain-compilation technique. Our preliminary experimental results suggests that the compilation technique substantially increases the planner's efficiency. A technical report on this topic is available [9], and we intend to make JSHOP2 itself available in a few months.

5.3 Planning Under Uncertainty

In planning research, the "classical" model of actions is that they have deterministic outcomes. However, in many situations where one might want to do planning, it may be useful to assume that some actions have more than one possible outcome. This action model can be useful in situations where the outcome of an action might vary due to random changes in the environment or to the actions of other agents.

We are developing a general technique for taking forward-chaining planners for deterministic domains and adapting them to work in nondeterministic planning domains, i.e., planning domains in which each action may have more than one possible outcome. We have shown both theoretically and experimentally that our approach can produce exponential speedups over previous algorithms for planning in nondeterministic environments [5].

We are currently extending our approach to work for situations in which actions have probabilistic outcomes (e.g., MDP models of actions). We believe that we will be able to obtain similar speedups in these kinds of planning domains.

5.4 Planning with Distributed Information Sources

Planning researchers typically assume that the planning system is *isolated*: it begins with a complete description of the planning problem, and has no need of interacting with the external world during the planning process. In many practical situations, such an assumption is clearly unrealistic: the planner may need to obtain information from external sources during planning.

We have developed a formalism for *wrappers* that may be placed around conventional (isolated) planners to replace some of the planner's memory accesses with queries to external information sources. When appropriate, the wrapper can automatically backtrack the planner to a previous point in its operation. We have done both mathematical and experimental analysis of several different query-management strategies for these wrappers, i.e., strategies for when to issue queries, and when/how to backtrack the planner. Our results [1] show conditions under which different query management strategies are preferable, and suggest that domain-configurable planners such as SHOP2 are likely to be better suited than other planners for planning with volatile information.

Even better performance can be obtained if a planner can make *non-blocking* queries to external information sources, i.e., if the planner can continue exploring other parts of its search space while waiting for the response to a query. We have developed a modified version of SHOP2 that works in this way. We have shown experimentally that this dramatically improves (i) the time needed to find a solution and (ii) in cases where the information source is not guaranteed to respond, the chance of finding a solution at all [6].

Acknowledgments

This work was supported in part by the following grants, contracts, and awards: Air Force Research Laboratory F30602-00-2-0505, Army Research Laboratory DAAL0197K0135, and Naval Research Laboratory N00173021G005. The opinions expressed in this paper are those of authors and do not necessarily reflect the opinions of the funders.

We also wish to acknowledge the following people who have shared information with us about the use of SHOP and SHOP2 in their projects: David Aha at NRL, Jeremi Gancet at LAAS/CNRS, Peter Denno at NIST, Robert Goldman at SIFT LLC, Sergio Gigli and Benjamin Grooters at

Lockheed Martin ATL, Mark Plutowski at Sony Electronics Incorporated, Nuwan Waidyanatha at Infocraft Ltd., Nate Blaylock at the University of Rochester, William Regli at Drexel University, Jaime Guzman at the National University of Colombia, Adrian Groza at the Technical University of Cluj-Napoca, Romania, Ella Atkins at the University of Maryland, and Filip Jagodzinski at Villanova University.

References

[1] Tsz-Chiu Au, Dana Nau, and V.S. Subrahmanian. Utilizing volatile external information during planning. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, August 2004. To appear.

[2] Kutluhan Erol, James Hendler, and Dana S. Nau. Complexity results for hierarchical task-network planning. *Annals of Mathematics and Artificial Intelligence*, 18:69–93, 1996.

[3] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning: Theory and Practice*. Morgan Kaufmann, May 2004.

[4] Okhtay Ilghami, Dana S. Nau, Héctor Muñoz-Avila, and David W. Aha. CaMeL: Learning methods for HTN planning. In *AIPS-2002*, Toulouse, France, 2002.

[5] U. Kuter and D. Nau. Forward-chaining planning in nondeterministic domains. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2004. To appear.

[6] U. Kuter, E. Sirin, D. Nau, B. Parsia, and J. Hendler. Information gathering during planning for web services composition. In *ICAPS-04 Workshop on Planning and Scheduling for Web and Grid Services*, 2004.

[7] Dana Nau, Tsz-Chiu Au, Okhtay Ilghami, Ugur Kuter, J. William Murdock, Dan Wu, and Fusun Yaman. SHOP2: An HTN planning system. *Journal of Artificial Intelligence Research*, 20:379–404, December 2003.

[8] Dana S. Nau, Yue Cao, Amnon Lotem, and Héctor Muñoz-Avila. SHOP: Simple hierarchical ordered planner. In Thomas Dean, editor, *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 968–973. Morgan Kaufmann Publishers, July 31–August 6 1999.

[9] Okhtay Ilghami Okhtay and Dana S. Nau. A general approach to synthesize problem-specific planners. Technical Report CS-TR-4597, UMIACS-TR-2004-40, University of Maryland, October 2003.

[10] E. Sacerdoti. The nonlinear nature of plans. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 206–214, 1975.

[11] Evren Sirin, Bijan Parsia, Dan Wu, James Hendler, and Dana Nau. HTN planning for web service composition using SHOP2. *Web Semantics Journal*, 2004. To appear.

[12] Stephen J. J. Smith, Dana S. Nau, and Thomas Throop. Computer bridge: A big win for AI planning. *AI Magazine*, 19(2):93–105, 1998.

[13] A. Tate. Generating project networks. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 888–893, 1977.

[14] A. Tate, B. Drabble, and R. Kirby. *O-Plan2: An Architecture for Command, Planning and Control*. Morgan-Kaufmann, 1994.

[15] David E. Wilkins. *Practical Planning: Extending the Classical AI Planning Paradigm*. Morgan Kaufmann, San Mateo, CA, 1988.

[16] Dan Wu, Bijan Parsia, Evren Sirin, James Hendler, and Dana Nau. Automating DAML-S web services composition using SHOP2. In *Proceedings of the Second International Semantic Web Conference (ISWC2003)*, 2003.