The SHOP Planning System

Dana Nau, Yue Cao, Amnon Lotem, and Héctor Muñoz-Avila

■ SHOP is a hierarchical task network planning algorithm that is provably sound and complete across a large class of planning domains. It plans for tasks in the same order that they will later be executed, and thus, it knows the current world state at each step of the planning process. SHOP takes advantage of this knowledge by allowing a high degree of expressive power in its knowledge bases. For example, SHOP's preconditions can include logical inferences, complex numeric computations, and calls to external programs. SHOP is powerful enough that an implementation of it is being used as an embedded planner in the Naval Research Laboratory's HICAP system.

SHOP (simple hierarchical ordered planner) is a domain-independent generalization of a planning technique that we originally developed for use in several domain-specific planning systems, including the EDAPS system for manufacturing planning (Smith et al. 1997) and the BRIDGE BARON program for declarer play in the game of bridge (Smith, Nau, and Throop 1998). The SHOP algorithm is a hierarchical task network (HTN) planning algorithm, but it differs from other HTN planning algorithms in that SHOP plans for tasks in the same order that they will be executed. Thus, SHOP always knows the current state of the world at each step of the planning process, and SHOP takes advantage of this knowledge by incorporating a high degree of expressive power into its domain representations. For example, SHOP's preconditions can include Horn-clause inferencing, numeric computations, and calls to external programs.

SHOP's expressive power can be used to create domain representations for complex application domains. For example, an implementation of SHOP is being used as the generative planning module for HICAP (Muñoz-Avila et al. 2001a, 1999), a plan-authoring system for noncombatant evacuation operations (NEOs).

The SHOP Planning Algorithm

Here, we summarize the SHOP algorithm's primary features. For more details, see Nau et al. (1999).[1]

Because SHOP is an HTN planning algorithm, it creates plans by recursively decomposing tasks (activities that need to be performed) into smaller and smaller subtasks, until primitive tasks are reached (tasks that can be accomplished directly). SHOP uses methods and operators as in HTN planning. An *operator* (similar to a STRIPS operator) specifies a way to perform a primitive task, and a *method* specifies a way to decompose a nonprimitive task into a set of subtasks.

Unlike most HTN planners, SHOP requires the decomposition produced by each method to be a totally ordered set of subtasks. Using this restriction, SHOP plans for the tasks in the same order that they will later be executed, which makes SHOP simpler than HTN planners such as NONLIN (Tate 1977), SIPE-2 (Wilkins 1990), O-PLAN (Currie and Tate 1991), and UMCP (Erol, Hendler, and Nau 1994) and makes it easier to prove soundness and completeness results for SHOP.

The SHOP algorithm is shown in figure 1. As an example of how it works, figure 2 shows two methods for traveling from one location to another: (1) traveling by airplane and (2) traveling by taxi. Note that each method produces a totally ordered list of subtasks. Suppose that all these subtasks are primitive except for the travel subtasks. If we asked SHOP to use these methods to find a plan for the task of traveling from the University of Maryland to

```
procedure SHOP (S, T, D)
1. if T = nil then return nil endif
2. t = the first task in T
3. U = the remaining tasks in T
4. if t is primitive (i.e., there is an operator for t) then
5.     nondeterministically choose an operator o for t
6.     P = SHOP (o(S), U, D)
7.     if P = FAIL then return FAIL endif
8.     return cons(p, P)
9. else if there is a method applicable to t whose
       preconditions can all be inferred from S then
10.    nondeterministically let m be such a method
11.    return SHOP(S, append(m(t,S), U), D)
12. else
13.    return FAIL
14. endif
end SHOP
```

Figure 1. The SHOP Planning Algorithm.

S is a state, *T* is a list of tasks, and *D* is the knowledge base (methods, operators, and Horn-clause axioms).

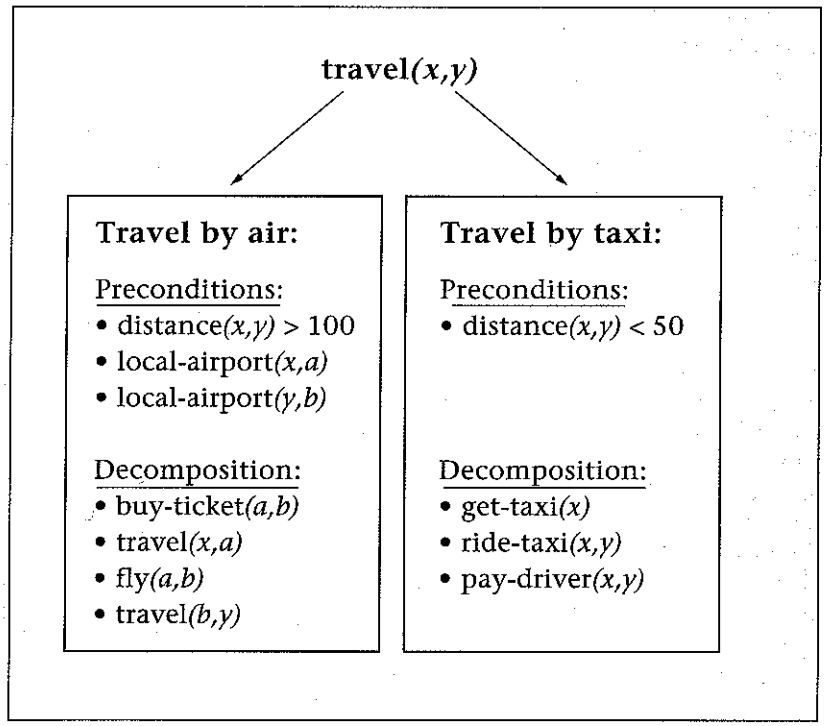

Figure 2. Two Different Methods for Traveling from One Location to Another.

the Massachusetts Institute of Technology, SHOP would expand the tasks and subtasks in the order shown in figure 3.

As long as the procedure for inferring *m*'s preconditions from *S* is a sound and complete inference procedure (such as Horn-clause theorem proving), the SHOP algorithm itself will also be sound and complete.

In implementations of SHOP,[2] the inference procedure is a Horn-clause theorem prover with several extensions. For example, the Horn clauses can include calls to attached procedures for numeric computations (for example, "distance(UofMD,BWI)<50" in the previous example), or (in some of the implementations) any other procedure calls defined by the user.

Experimental Results

In our experiments (Nau et al. 1999), SHOP generated plans several orders of magnitude more quickly than BLACKBOX (Kautz and Selman 1999), IPP (Koehler et al. 1997), and UMCP (Erol, Hendler, and Nau 1994). We believe the primary reason it outperformed these planners was because SHOP's higher level of expressivity made it possible to formulate highly expressive domain algorithms in SHOP.

In our experiments, SHOP also performed several times as fast as TLPLAN (Bacchus and Kabanza 2000). TLPLAN, which does a forward search guided by pruning rules written in modal logic, has expressivity similar to that of SHOP, and in fact, we believe that the big-O complexity of TLPLAN and SHOP was not too different.

In the Fifth International Conference on Artificial Intelligence Planning and Scheduling (AIPS'00) planning competition, SHOP was outperformed by TALPLANNER (Doherty and Kvarnström 1999). TALPLANNER is based on TLPLAN but achieves substantial speedups compared to TLPLAN through the use of preprocessing techniques and fast data structures. We have started to make changes to SHOP's data structures to make them faster; for example, we found that a simple change to the data structure SHOP uses to represent its world states would speed SHOP up by about an order of magnitude on large problems. We intend to make more optimizations in the near future.

An Application of SHOP

One of our implementations of SHOP is used as an embedded planning module in the Naval Research Laboratory's HICAP plan-authoring system for noncombatant evacuation operations (NEOs) (Muñoz-Avila et al. 2001a, 1999). The architecture of HICAP is shown in figure 4. HICAP dynamically elaborates plans, derived from military doctrine on NEOs and represented as HTNs, using interactive case-based inferencing

(Aha and Breslow 1997). HICAP assists users with dynamic plan elaboration by providing the following functions: (1) manual editing of plans represented using HTNs using a hierarchical task editor, (2) interactive plan expansion using a case-based reasoning module called NaCoDAE (Aha and Breslow 1997), (3) automated plan expansion using SHOP, and (4) a lessons delivery module that monitors HICAP's plan to notify the user when lessons become applicable and recommend corresponding plan-elaboration operations.

We are currently extending the capabilities of HICAP and SHOP as part of the Defense Advanced Research Project Agency's Active Templates Program.

Discussion and Conclusions

SHOP illustrates the synergy that can result from the interplay between planning applications and planning theory. The SHOP algorithm is a domain-independent formalization of our previous domain-specific work on domain-specific planners for applications in manufacturing planning and the game of bridge. Conversely, an implementation of the SHOP algorithm is being used as an embedded planning system in the HICAP application program.

Our ongoing and future work on SHOP is as follows:

We have developed an algorithm called SHOP2 (Nau et al. 2001), which still generates the steps of a plan in the same order that they will later be executed but does not require the subtasks of a method to be partially ordered. In some cases, it is much easier to write knowledge bases for SHOP2 than for SHOP.

We are integrating SHOP with the IMPACT (Eiter and Subrahmanian 1999; Eiter et al. 1999) multiagent architecture to provide planning in a multiagent environment. We have developed the theoretical foundations for this integration (Dix, Münoz-Avila, and Nau 2001, 2000) and are developing an implementation.

We are making optimizations to SHOP's data structures, as mentioned earlier. We believe that these optimizations will speed up SHOP by several orders of magnitude.

We are extending SHOP to incorporate ways to reason about time and uncertainty, generate and evaluate contingency plans, and react to new information from external programs. We believe these extensions will be useful in several problem domains, such as the NEO domain mentioned earlier.

Because of the similarity between HTN planning and the work breakdown structures (WBSs) used in commercial project management packages (Muñoz-Avila et al. 2001b), we hope to develop HTN planning techniques to assist human project planners in creating WBSs.

1. travel(U of MD, MIT)
   2. buy-ticket(BWI, Logan)
   3. travel(U of MD, BWI)
      4. get-taxi(U of MD)
      5. ride-taxi(U of MD, BWI)
      6. pay-driver
   7. fly(BWI, Logan)
   8. travel(Logan, MIT)
      9. get-taxi(Logan)
      10. ride-taxi(Logan, MIT)
      11. pay-driver(Logan, MIT)

Figure 3. The Order in Which SHOP Would Decompose Tasks While Planning How to Travel from the University of Maryland to the Massachusetts Institute of Technology.

Tasks 1, 3, and 8 are nonprimitive; all the other tasks are primitive.

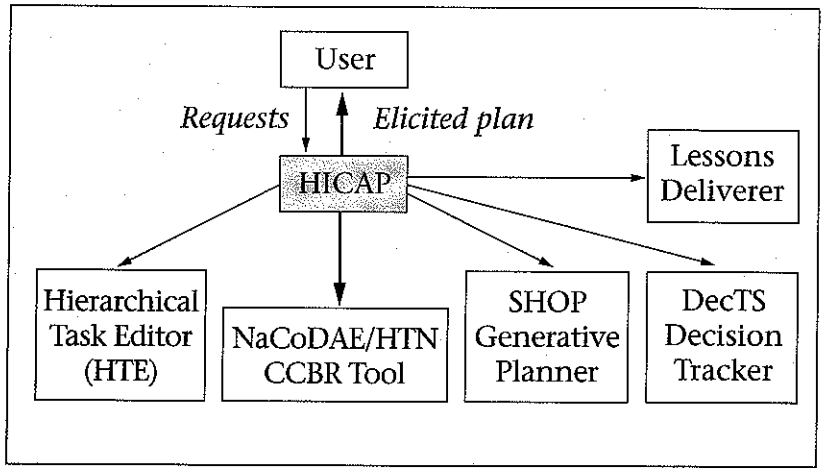

Figure 4. Architecture of HICAP, a System for Authoring Noncombatant Evacuation Operation (NEO) Plans.

Acknowledgments

This work was supported in part by the following funding sources: AFRL F306029910013 and F30602-00-2-0505, ARL DAAL0197K0135, and University of Maryland General Research Board. Opinions expressed here do not necessarily reflect those of the funders.

Note

1. Also see www.cs.umd.edu/projects/shop.

2. Implementations of SHOP are available at www.cs. umd.edu/projects/shop.

References

Aha, D., and Breslow, L. 1997. Refining Conversational Case Libraries. Paper presented at the Second International Conference on Case-Based Reasoning (ICCBR-97), 25–27 July, Providence, Rhode Island.

Bacchus, F., and Kabanza, F. 2000. Using Temporal Logics to Express Search Control Knowledge for Planning. *Artificial Intelligence* 116(1–2): 123–191.

Currie, K., and Tate, A. 1991. O-PLAN: The Open Planning Architecture. *Artificial Intelligence* 52(1): 49–86.

Dix, J.; Muñoz-Avila, H.; and Nau, D. S. 2001. IMPACTING SHOP: Putting an AI Planner into a Multiagent Environment. *Annals of Mathematics and AI.* Forthcoming.

Dix, J.; Muñoz-Avila, H.; and Nau, D. 2000. IMPACTING SHOP: Planning in a Multi-Agent Environment. Paper presented at the CL-2000 Workshop on Computational Logic in Multi-Agent Systems (CLIMA-00), 24–25 July, London, United Kingdom.

Doherty, P., and Kvarnström, J. 1999. TALPLANNER: An Empirical Investigation of a Temporal Logic–Based Forward-Chaining Planner. Paper presented at the Sixth International Workshop on Temporal Representation and Reasoning (TIME'99), 1–2 May, Orlando, Florida.

Eiter, T., and Subrahmanian, V. S. 1999. Heterogeneous Active Agents, II: Algorithms and Complexity. *Artificial Intelligence* 108(1–2): 257–307.

Eiter, T.; Subrahmanian, V. S.; and Pick, G. 1999. Heterogeneous Active Agents, I: Semantics. *Artificial Intelligence* 108(1–2): 179–255.

Erol, K.; Hendler, J.; and Nau, D. 1994. UMCP: A Sound and Complete Procedure for Hierarchical Task Network Planning. Paper presented at the Second International Conference on AI Planning Systems (AIPS-94), 13–15 July, Chicago, Illinois.

Kautz, H., and Selman, B. 1. 1999. Unifying SAT-Based and Graph-Based Planning. In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, 318–325. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Koehler, J.; Nebel, B.; Hoffman, J.; and Dimopoulus, Y. 1997. Extending Planning Graphs to an ADL Subset. Paper presented at the European Conference on Planning (ECP-97), 24–26 September, Toulouse, France.

Muñoz-Avila, H.; Aha, D.; Ballas, J.; Breslow, L.; and Nau, D. 1999. Using Guidelines to Constrain Interactive Case-Based HTN Planning. Paper presented at the International Conference on Case-Based Reasoning (ICCBR-99), 27–30 July, Seeon Monastery, Germany.

Muñoz-Avila, H.; Aha, D.; Nau, D.; Weber, R.; Breslow, L.; and Yaman, F. 2001a. SiN: Integrating Case-Based Reasoning with Task Decomposition. Paper presented at the Seventeenth International Joint Conference on Artificial Intelligence, 4–10 August, Seattle, Washington.

Muñoz-Avila, H.; Gupta, K.; Aha, D.; and Nau, D. 2001b. Knowledge-Based Project Planning. Paper presented at the IJCAI-2001 Workshop on Knowledge Management and Organizational Memories, 6 August, Seattle, Washington.

Nau, D.; Cao, Y.; Lotem, A.; and Muñoz-Avila, H. 1999. SHOP: Simple Hierarchical Ordered Planner. In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, 968–973. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Nau, D.; Muñoz-Avila, H.; Cao, Y.; Lotem, A.; and Mitchell, S. 2001. Total-Order Planning with Partially Ordered Subtasks. Paper presented at the Seventeenth International Joint Conference on Artificial Intelligence, 4–10 August, Seattle, Washington.

Sacerdoti, E. 1977. *A Structure for Plans and Behavior.* New York: American Elsevier.

Smith, S.; Hebbar, K.; Nau, D.; and Minis, I. 1997. Integrating Electrical and Mechanical Design and Process Planning. In *Knowledge Intensive CAD, Volume 2,* eds. M. Mantyla, S. Finger, and T. Tomiyama, 269–288. London: Chapman and Hall.

Smith, S.; Nau, D.; and Throop, T. 1990. Computer Bridge: A Big Win for AI Planning. *AI Magazine* 19(2): 93–105.

Tate, A. 1977. Generating Project Networks. In Proceedings of the Fifth International Joint Conference on Artificial Intelligence, 888–893. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Wilkins, D. 1990. Can AI Planners Solve Practical Problems? *Computational Intelligence* 6(4): 232–246.

Dana Nau is a professor at the University of Maryland in the Department of Computer Science and the Institute for Systems Research. His research interests include AI planning and searching and computer-integrated design and manufacturing. He received his Ph.D. from Duke University in 1979, where he was a National Science Foundation (NSF) graduate fellow. He has received an NSF Presidential Young Investigator award, the ISR Outstanding Faculty award, and several "best paper" awards. He is a fellow of the American Association for Artificial Intelligence. His e-mail address is nau@cs. umd.edu.

Yue (Jason) Cao received a B.S. and an M.S. in computer science from the University of Maryland, where he did some of the programming work on the SHOP planning system. He currently works at Solers, Inc., in Arlington, Virginia. His e-mail address is JasonC@homemail.com.

Amnon Lotem is the algorithm development manager of Estimotion Ltd., Israel. He teaches at Tel Aviv University, Israel. He received his Ph.D. in computer science from the University of Maryland at College Park, where he focused on efficient algorithms for hierarchical task network planning. His e-mail address is a_lotem@yahoo.com.

Hector Muñoz-Avila is an assistant research scientist at the University of Maryland at College Park. Muñoz-Avila received his B.S. and M.S. in computer science and a B.S. in mathematics from the University of the Anden (Colombia). He received his Ph.D. (Dr. rer nat) from the University of Kaiserslautern (Germany), where he was a DAAD graduate fellow. Muñoz-Avila has been on program committees and a reviewer for international conferences, including AAAI and ICCBR. He was recently appointed for a faculty position at Lehigh University. His e-mail address is munoz@cs. umd.edu.