EEE Expert, Intelligent Systems & their Applications, Vol. 11(6), 1996.

AI planning systems in the real world

Planning systems generate partially ordered sequences of actions (or plans) that solve a goal. They start from a specification of the valid actions (also called operators), which includes both the conditions under which an action applies (the preconditions) and the expected outcome of applying that action (the effects). The general problem is quite hard both because of the potentially enormous search space and the difficulty in fully and accurately representing real-world problems. Approaches to planning include operator-based planning, hierarchical task-network planning, case-based planning, reactive planning, and many more. Early planning work focused largely on "toy" problems (for example, the blocks world). More recently, there has been a big push toward applying planning systems to real-world applications. While planning systems have not yet achieved the level of commercial success enjoyed by some other areas of artificial intelligence—neural nets, for example—a number of successful applications of planning technology to real-world problems have recently emerged.

This installment of "Trends & Controversies" highlights five such applications. I have asked the developers of these systems to describe the application domain and the planning technology used to solve the problems. These systems all use some form of hierarchical task-network planning (in some cases combined with other techniques). HTN planning provides a way of specifying, as part of the operator definition, how to hierarchically expand actions into partially ordered sequences (task networks) of actions. This approach succeeds, in part, because it provides a natural way of limiting the possibly very large search spaces. See *Readings in Planning* (Morgan Kaufmann, 1990) or *Artificial Intelligence: A Modern Approach* (Prentice Hall, 1995) for more details on various planning techniques.

In the first article, Stephen Smith, Dana Nau, and Thomas Throop describe their use of planning technology to build a system for declarer play in contract bridge. The system can beat the best commercially available program and is currently being incorporated into a commercial product. Second, John Mark Agosta and David Wilkins describe how the SIPE-2 planner helps evaluate the US Coast Guard's ability to respond to marine oil spills. This system, which automates a problem that is currently done by hand, is undergoing evaluation by the Coast Guard. Third, Austin Tate describes a planning application, in use by the European Space Agency, for the project management of spacecraft assembly, integration, and verification. Fourth, Steve Chien and his colleagues describe their use of a planning system to automate the operations of NASA's Deep Space Network communication antennas. This system is currently being integrated into a new system that will become operational in 1997. Finally, Thomas Lee and David Wilkins describe their use of SIPE-2 in producing military air campaign plans. Their planner is part of a demonstration system that is fully integrated with the other software modules currently used for solving parts of this problem.

—*Craig Knoblock*

AI planning's strong suit

Stephen J.J. Smith, Hood College
Dana Nau, University of Maryland
Thomas Throop, Great Game Products

Although game-tree search techniques work well in perfect-information games such as chess, checkers, and Othello, difficulties arise in adapting them to imperfect-information games such as bridge. In bridge, the game tree's branching factor is very large because no player has complete knowledge about the state of the world, the possible actions, and their effects. Because bridge deals must be played in just a few minutes, a full game-tree search will not search a significant portion of this tree within the time available. Matthew Ginsberg is developing a modified game-tree search procedure to address this problem.[1] However, others have shown some pitfalls in any approach that (like Ginsberg's) treats an incomplete-information problem as a collection of complete-information problems.[2] No evidence yet proves that these pitfalls can be overcome.

Our approach grows out of the observation that bridge is a game of planning. The bridge literature describes a number of tactical and strategic schemes (such as finessing, ruffing, and crossruffing) that people combine into plans in playing bridge games. We have taken advantage of the planning nature of bridge, by adapting and extending some ideas from HTN planning.

Approach

HTN planning is an AI planning methodology that creates plans by *task decomposition*—by decomposing tasks into smaller and smaller subtasks until primitive tasks are found that can be performed directly. HTN planning systems have knowledge bases containing *methods* that tell how to develop plans by such decompositions.[3–5] Given a task to accomplish, the planner chooses an applicable method and instantiates it to decompose the task into subtasks, and chooses and instantiates other methods to decompose the subtasks even further. If the constraints on the subtasks or the interactions among them prevent the plan from being feasible, the planning system will backtrack and try other methods.

To represent the tactical and strategic schemes of card playing in bridge, we use structures similar to the methods described just now, but modified to represent multiagency and uncertainty. For example, Figure 1 shows a portion of our task-network structure for a bridge tactic called *finessing*. To generate game trees, we use a procedure similar to task decomposition to build up a game tree whose branches represent moves generated by these methods.

For a game tree generated in this manner, the number of branches from each state is *not* the number of actions an agent can perform (as in conventional game-tree search procedures), but instead is the number of different tactical and strategic schemes the agent can employ. This results in a smaller branching factor and a much smaller search tree: Tignum 2 generates game trees containing only about 420,000 nodes in the worst case and 26,000 nodes on the average, as compared to 6×10^{44} nodes in the worst case and 10^{29} nodes on the average if we had generated a conventional game tree. Thus, Tignum 2 can search the game tree all the way to the end, to predict the likely results of the various sequences of cards it might play.[6,7]

Comparison with conventional HTN planning

In Tignum 2, we have extended HTN planning to include ways to represent and reason about possible actions by other agents (such as the opponents in a bridge game), as well as uncertainty about their capabilities (for example, lack of knowledge about what cards they have). However, to accomplish this, we needed to restrict how Tignum 2 formulates its plans. Most HTN planners develop plans in which the actions are only partially ordered, postponing some of the decisions about the order in which the actions will be performed. In contrast, Tignum 2 is a total-order planner that expands tasks in left-to-right order.

Because Tignum 2 expands tasks in the same order in which they will be performed when the plan executes, this means that when it plans for each task, Tignum 2 already knows the state of the world (or as much as can be known about it in an imperfect-information game) at the time that the task will be performed. Consequently, we can write each method's preconditions as arbitrary computer code, rather than using the stylized logical expressions found in most AI planning systems. This enables us to encode the complex numeric computations needed for reasoning about the probable locations of the opponents' cards. For example, by knowing the current state, Tignum 2 can decide which of 19 finesse situations are applicable: with partial-order planning, it would be much harder to decide which of them *can be made* applicable.

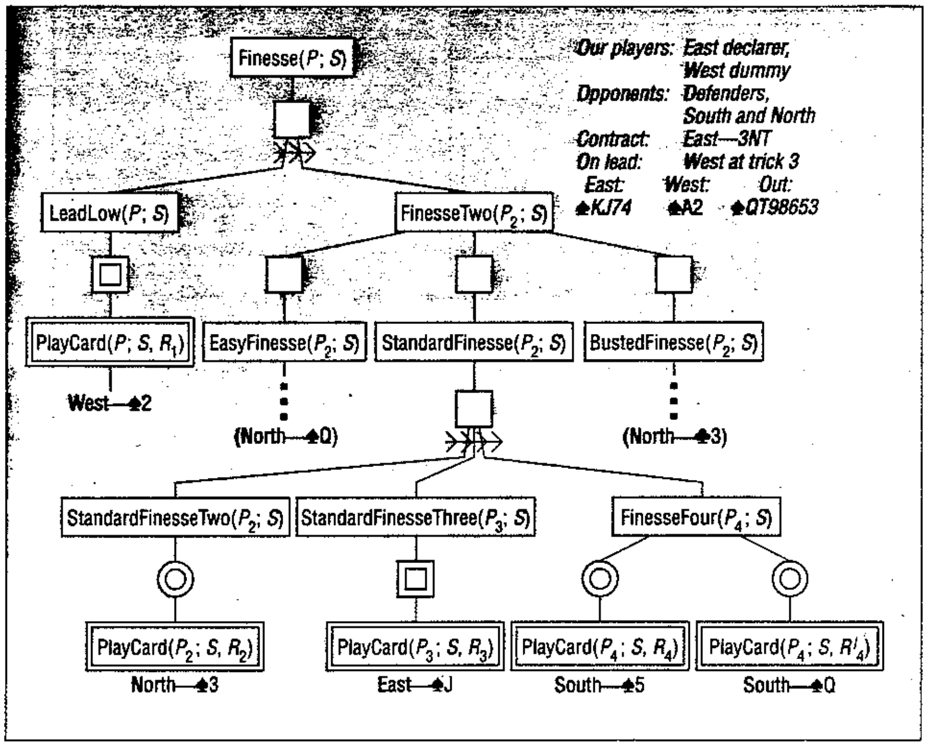

Figure 1. A portion of a finesse method.

Performance

To test Tignum 2, we played it against Bridge Baron, from Great Game Products. Winner of a number of important bridge competitions, Bridge Baron is probably the best program for declarer play at contract bridge. In reviewing seven commercially available bridge-playing programs, the American Contract Bridge League rated Bridge Baron best and also best of the five that do declarer play without "peeking" at the opponents' cards.[8]

When we tested Tignum 2 against Bridge Baron on 1,000 randomly generated bridge deals (including both suit and no-trump contracts), Tignum 2 beat Bridge Baron by 254 to 202, with 544 ties. These results are statistically significant at the $\alpha = 0.05$ level. We had never run Tignum 2 on any of these deals before this test, so these results are free from any training-set biases.

Conclusions

The use of HTN planning techniques in Tignum 2 enables it to do bridge declarer play better than Bridge Baron. Tignum 2 is being incorporated into Bridge Baron to improve the Baron's declarer play.

We have been quite successful in using the same modified version of HTN planning (as well as some of the same code!) in another very different application domain: the task of generating process plans for the manufacture of complex electro-mechanical devices.[9] That this same approach works well in two such widely varying areas is quite striking, suggesting that our approach may be useful in a number of practical planning problems.

Acknowledgments

This work was supported in part by an AT&T PhD scholarship to Stephen J. J. Smith, by Maryland Industrial Partnerships (MIPS) Grant 501.15, by Great Game Products, by ARPA Grant DABT 63-95-C-0037, and by National Science Foundation Grants NSF EEC 94-02384 and IRI-9306580.

References

1. M. Ginsberg, "How Computers Will Play Bridge," to appear in *Bridge World*.
2. I. Frank and D. Basin, "Search in Games with Incomplete Information: A Case Study Using Bridge Card Play," Research Paper 780, Dept. of AI, Univ. of Edinburgh, Edinburgh, Scotland, 1995.
3. D.E. Wilkins, *Practical Planning*, Morgan Kaufmann, San Francisco, 1988.
4. K. Erol, J. Hendler, and D.S. Nau, "UMCP: A Sound and Complete Procedure for Hierarchical Task-Network Planning," *Proc. Second Int'l Conf. AI Planning Systems*, AAAI Press, Menlo Park, Calif., 1994, pp. 249–254.
5. E.D. Sacerdoti, *A Structure for Plans and Behavior*, American Elsevier, New York, 1977.
6. S.J.J. Smith, D.S. Nau, and T. Throop, "A Planning Approach to Declarer Play in Contract Bridge," *Computational Intelligence*, Vol. 12, No. 1, Feb. 1996, pp. 106–130.
7. S.J.J. Smith, D.S. Nau, and T. Throop, "Total-Order Multiagent Task-Network Planning for Contract Bridge," *AAAI-96*, AAAI Press, 1996, pp. 108–113.
8. B. Manley, "Software 'Judges' Rate Bridge-Playing Products," *The Bulletin*, Vol. 59, No. 11, Nov. 1993, pp. 51–54.
9. K. Hebbar et al., "Plan-Based Evaluation of Designs for Microwave Modules," *ASME. Design for Manufacturing Conf.*, Am. Soc. Mechanical Engineers, New York, 1996, p. 262.

begins by entering the specifics of a spill incident—location, time of day, spill rate, and so on—then forecasting the spill trajectory, considering the uncertainty in its spreading caused by wind and waves. This forecast determines which environmentally sensitive shore sectors the oil will hit, and when. The planner works from this forecast, together with geographic information, such as the sectors into which the region is divided and the USCG requirements for protection of these areas. In addition, the planner works with the database of the quantities and capabilities of available equipment and resources, and where they are located. The planner, SIPE-2, and scheduler, Tachyon, then work interactively with the user to generate a plan of equipment deployment and employment actions that meet constraints among oil spreading, equipment cleanup capabilities and transport times, and environmental protection requirements. Finally, the evaluation module uses the scheduler output and the projected flows from the trajectory model to determine the effectiveness of the plan.[2]

Most of the user's interaction with SRCS is mediated by a map interface, implemented in the Arcview commercial geographical information system. The user thus can immediately see both the extent of the spill and where resources are employed at various times.

Evaluating the plan

In the SRCS domain, plans are distinguished by the degree to which they achieve the overall objective of cleaning up the spilled oil. In many spills, much of the oil will escape, no matter how much equipment is available, because of the difficulty of operations and speed of spreading due to the weather. Furthermore, for any spill, SRCS can generate many possible plans, and users can partially or completely sacrifice a sector cleanup goal if they believe equipment that would have been assigned to a sector better serves the overall goals by being used elsewhere. The plan and the oil flows determined by the trajectory model become the input to the evaluation model. The evaluation model accounts for the quantities of oil contained and removed in each sector, for each period. From this accounting, it can calculate measures of plan merit, such as the final fraction of oil removed under each plan.

Because the evaluation model is graphic and efficiently computed, SRCS has

Stephen J.J. Smith is an assistant professor at Hood College. His research interests include task-network planning, forward pruning, and search. He earned a BS in computer science, mathematics, and Latin from Dickinson College and an MS in computer science from the University of Maryland, College Park, and is expecting his PhD in computer science from the University of Maryland. Contact him at the Dept. of Mathematics and Computer Science, Hood College, 401 Rosemont Ave., Frederick, MD 21701-8575; sjsmith@nimue.hood.edu.

Dana Nau is a professor of computer science at the University of Maryland's Institute for Systems Research. His research interests include computer-integrated manufacturing, AI planning, and search algorithms. He received a BS in applied mathematics from the University of Missouri-Rolla and an AM and a PhD in computer science from Duke. He is a fellow of the AAAI. Contact him at the Dept. of Computer Science, Univ. of Maryland, College Park, MD 20742; nau@cs.umd.edu; http://www.cs.umd.edu/~nau.

Thomas Throop is the president of Great Game Products, maker of Bridge Baron, the leading computer bridge-playing program. He has a BS in electrical engineering from Swarthmore College. Contact him at (301) 365-5277; greatgames@delphi.com.

John Mark Agosta is a computer scientist in SRI's Applied AI Technology Program. He specializes in the formulation and design of probabilistic and economic models, specifically Bayesian and decision-theoretic methods applied to automated planning and decision making under uncertainty. He earned his BS from Yale, his MA from George Washington, and his PhD in engineering from Stanford. Contact him at SRI International, 333 Ravenswood Ave., Menlo Park, CA 94025; johnmark@sri.com; http://www.erg.sri.com/people/johnmark/.

David E. Wilkins is a senior computer scientist at the SRI AI Center, where his research focuses on planning and reasoning about actions, knowledge representation, and design and implementation of AI systems. He holds a BS from Iowa State University, an MSc from the University of Essex, and a PhD from Stanford. He published *Practical Planning: Extending the Classical AI Planning Paradigm*, and recently led a project to develop Cypress, a system for creating taskable, reactive agents. Contact him at SRI International, 333 Ravenswood Ave., Menlo Park, CA 94025; wilkins@ai.sri.com; http://www.ai.sri.com/~wilkins.

Austin Tate is technical director of the Artificial Intelligence Applications Institute and holds the Personal Chair of Knowledge-Based Systems at the University of Edinburgh. Besides engaging in the research, development, and application of knowledge-based methods, he also has a background in databases and software engineering. He holds a PhD in machine intelligence from the University of Edinburgh. He is an editorial board member of *IEEE Expert*. Contact him at the AIAI, Univ. of Edinburgh, 80 South Bridge, Edinburgh EH1 1HN, UK; a.tate@ed.ac.uk; http://www.ed.ac.uk.

Steve Chien is technical group supervisor of the AI Group of the Jet Propulsion Laboratory, California Institute of Technology, where he leads efforts in research and development of automated planning and scheduling systems. He is also an adjunct assistant professor in the Department of Computer Science at the University of Southern California. He holds a BS, MS, and PhD in computer science from the University of Illinois. His research interests are in the areas of planning and scheduling, operations research, and machine learning. Contact him at steve.chien@jpl.nasa.gov; http://www-aig.jpl.nasa.gov.

Anita Govindjee is a member of the technical staff in JPL's AI Group. She holds an MS in computer science from Stanford and a BS in computer science from the University of Illinois. Her research interests are in AI and cognitive science. Contact her at anita.govindjee@jpl.nasa.gov.

XueMei Wang is a member of the technical staff at the Rockwell Science Center in Palo Alto, California. She holds a PhD in computer science from Carnegie Mellon and a BS in applied mathematics from Tsinghua University. Her research interests are AI—including planning, machine learning, and knowledge acquisition. Reach her at mei@rpal.rockwell.com.

Tara Estlin is a PhD candidate in computer science at the University of Texas at Austin. She holds a BS in computer science from Tulane University and an MS in computer science from the University of Texas at Austin. Her research interests include machine learning and planning and scheduling. Contact her at estlin@cs.utexas.edu.

Randall Hill Jr. is a research computer scientist at the Information Sciences Institute, University of Southern California, and also a research assistant professor in the USC Department of Computer Science. He holds a BS from the United States Military Academy at West Point, and MS and PhD degrees in computer science from USC. His research interests lie in the areas of intelligent agents and computer-assisted learning environments. Reach him at hill@isi.edu.

Thomas J. Lee is a senior research engineer at SRI International. His research interests include crisis planning, machine learning, and game playing. He received an MS in computer science from the University of Wisconsin and a BS in computer science from the University of Nebraska. He can be reached at SRI International, 333 Ravenswood Ave., Menlo Park, CA 94025, tomlee@erg.sri.com.