Total-Order Multi-Agent Task-Network Planning for Contract Bridge *

S. J. J. Smith and **D. S. Nau**
Computer Science Department and
Institute for Systems Research
University of Maryland
College Park, MD 20742, USA
sjsmith@cs.umd.edu nau@cs.umd.edu

T. A. Throop
Great Game Products
8804 Chalon Drive
Bethesda, MD 20817, USA
bridgebaron@mcimail.com

Abstract

This paper describes the results of applying a modified version of hierarchical task-network (HTN) planning to the problem of declarer play in contract bridge. We represent information about bridge in a task network that is extended to represent multi-agency and uncertainty. Our game-playing procedure uses this task network to generate game trees in which the set of alternative choices is determined not by the set of possible actions, but by the set of available tactical and strategic schemes.

This approach avoids the difficulties that traditional game-tree search techniques have with imperfect-information games such as bridge—but it also differs in several significant ways from the planning techniques used in typical HTN planners. We describe why these modifications were needed in order to build a successful planner for bridge.

This same modified HTN planning strategy appears to be useful in a variety of application domains—for example, we have used the same planning techniques in a process-planning system for the manufacture of complex electro-mechanical devices (Hebbar et al. 1996). We discuss why the same technique has been successful in two such diverse domains.

Introduction

Tignum 2 is a computer system for declarer play at the game of contract bridge. Tignum 2 currently performs better at declarer play than the strongest commercially available program.[1] On 5000 randomly generated deals (including both suit contracts and notrump contracts), Tignum 2 beat the strongest commercially available program 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 in favor of Tignum 2.

This paper discusses the following issues:

- Unlike traditional game-playing computer programs, Tignum 2 is based not on brute-force game-tree search but on a modified version of Hierarchical Task-Network (HTN) planning. We discuss why bridge presents problems for the traditional approach, and why an HTN planning approach has worked well in Tignum 2.
- Although Tignum 2 is an HTN planner, it incorporates several significant modifications to the planning techniques used in typical HTN planners. We extended the HTN framework to incorporate multi-agency and uncertain information, but restricted it to allow only totally-ordered plans. We describe why these modifications were needed in order to build a successful planner for the game of bridge.
- This same modified HTN planning strategy appears to be useful in a variety of application domains. For example, as described in (Hebbar et al. 1996), the same planning techniques (and some of the same code!) used in Tignum 2 have been used to build a process-planning system for the manufacture of complex electro-mechanical devices. We discuss why the same kind of planning technique has been successful in two such diverse domains.

In this paper, we present only a sketch of our approach. Full details of our approach are in (Smith, Nau, & Throop 1996).

*This material is based on work supported in part by an AT&T Ph.D. 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 the National Science Foundation under Grants NSF EEC 94-02384 and IRI-9306580. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funders.

[1] It is probably safe to say that the Bridge Baron is the best program in the world for declarer play at contract bridge. It has won a number of important computer bridge competitions—and in their review of seven commercially available programs (Manley 1993), the ACBL rated the Bridge Baron to be the best of the seven, and the skill of the Bridge Baron to be the best of the five that do declarer play without "peeking" at the opponents' cards.

Background

Although game-tree search works well in perfect-information games (such as chess (Levy & Newborn 1982; Berliner et al. 1990), checkers (Samuel 1967; Schaeffer et al. 1992), and othello (Lee & Mahajan 1990)), it does not always work as well in other games. One example is the game of bridge. Bridge is an imperfect-information game, in which no player has complete knowledge about the state of the world, the possible actions, and their effects. Thus the branching factor of the game tree is very large. Because the bridge deal 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.

To address this problem, some researchers have tried making assumptions about the placement of the opponents' cards based on information from the bidding and prior play, and then searching the game trees resulting from these assumptions. However, such approaches have several limitations, as described in Section .

In our work, we have taken a different approach to this problem, based on the observation that bridge is a game of planning. For addressing various card-playing situations, the bridge literature describes a number of *tactical schemes* (short-term card-playing tactics such as finessing and ruffing), and *strategic schemes* (long-term card-playing tactics such as crossruffing). It appears that there is a small number of such schemes for each bridge deal, and that each of them can be expressed relatively simply. To play bridge, many humans use these schemes to create plans. They then follow those plans for some number of tricks, replanning when appropriate.

We have taken advantage of the planning nature of bridge, by adapting and extending some ideas from task-network planning. To represent the tactical and strategic schemes of card-playing in bridge, we use *multi-agent methods*—structures similar to the "action schemas" or "methods" used in hierarchical single-agent planning systems such as Nonlin (Tate 1977), NOAH (Sacerdoti 1977), O-Plan (Currie & Tate 1985), and SIPE (Wilkins 1984; Wilkins 1988), but modified to represent multi-agency and uncertainty.

To generate game trees, we use a procedure similar to task decomposition. The methods that perform our tasks correspond to the various tactical and strategic schemes for playing the game of bridge. We then build up a game tree whose branches represent moves generated by these methods. This approach produces a game tree in which the number of branches from each state is determined not by the number of actions an agent can perform, but instead by the number of different tactical and strategic schemes the agent can employ. If at each node of the tree, the number of applicable schemes is smaller than the number of possible actions, this will result in a smaller branching factor, and a much smaller search tree.

Related Work

HTN Planning. Our work on hierarchical planning draws on (Tate 1977; Sacerdoti 1977). In addition, some of our definitions were motivated by (Erol, Hendler, & Nau 1994; Erol, Nau, and Subrahmanian, 1995). In particular, Wilkins's SIPE planning system (Wilkins 1984; Wilkins 1988) was a very successful hierarchical planning system. However, these works do not address the uncertainty and incomplete information required in bridge.

Planning with Uncertainty. Some work has been done on planning with uncertainty and incomplete information (Peot & Smith 1992; Draper, Hanks, & Weld 1994; Kushmerick, Hanks, & Weld 1994; Collins & Pryor 1995). However, these works do not address problems on the scale of bridge, where there is incomplete information about twenty-five cards. Encouragingly, problems on a grander scale are starting to be studied (Haddawy, Doan, & Goodwin 1995; Boutilier, Dearden, & Goldszmidt 1995; Lin & Dean 1995).

Multi-Agent Planning. Much of the previous research on multi-agent planning has dealt with different issues than those that concern us here. In reactive planning (Dean et al. 1993), the agent must respond in real time to externally-caused events—and the necessity of making quick decisions largely precludes the possibility of reasoning far into the future. In cooperative multi-agent planning (Gmytrasiewicz & Durfee 1992; Pednault 1987), the primary issue is how to coordinate the actions of cooperating agents—and this makes it largely unnecessary for a single planning agent to generate a plan that accounts for all of the alternative actions that another agent might perform.

Bridge. Some of the work on bridge has focused on bidding (Lindelof 1983; Gamback, Rayner, & Pell 1990; Gamback, Rayner, & Pell 1993). Stanier (1975) and Quinlan (1979) took some tentative steps towards the problem of bridge play. Berlin (1985) an approach to play of the hand at bridge similar to ours; sadly, he never had a chance to develop the approach (his paper was published posthumously).

There are no really good computer programs for card-playing in bridge, especially in comparison to the success of computer programs for chess, checkers, and othello; most computer bridge programs can be beaten by a reasonably advanced novice. Sterling and Nygate (1990) wrote a rule-based program for recognizing and executing squeeze plays, but squeeze opportunities in bridge are rare. Recently, Frank, Basin, and Bundy

(1992) have proposed a proof-planning approach, but thus far, they have only described the results of applying this approach to planning the play of a single suit. Khemani (1994) has investigated a case-based planning approach to notrump declarer play, but hasn't described the speed and skill of the program in actual competition. The approaches used in current commercial programs are based almost exclusively on domain-specific techniques.

One approach is to make assumptions about the placement of the opponents' cards based on information from the bidding and prior play, and then search the game tree resulting from these assumptions. This approach was taken in the *Alpha Bridge* program (Lopatin 1992), with a 20-ply (5-trick) search. However, this approach didn't work very well: at the 1992 Computer Olympiad, Alpha Bridge placed last.

Game-Tree Search with Uncertainty. Play of better quality can be achieved by generating several random hypotheses for what hands the opponents might have, and doing a full complete-information game-tree search for each hypothesized hand. This approach is used late in the deal in Great Game Products' *Bridge Baron* program. Ginsberg (1996) has proposed using such an approach throughout the deal, employing clever techniques that make it possible to perform such a full game-tree search in a reasonable amount of time. However, Frank and Basin (1996) have shown some pitfalls in any approach that treats an incomplete-information problem as a collection of complete-information problems, as these approaches do. There is not yet any evidence that these pitfalls can be overcome.

Some work has been done on extending game-tree search to address uncertainty, including Horacek's (1990), and Ballard's (1983) work on backgammon. However, these works do not address the kind of uncertainty involved in bridge, and thus it does not appear to us that these approaches would be sufficient to accomplish our objectives.

Planning in Games. Wilkins (1980; 1982) uses "knowledge sources" to generate and analyze chess moves for both the player and the opponent. These knowledge sources have a similar intent to the multi-agent methods that we describe in this paper—but there are two significant differences. First, because chess is a perfect-information game, Wilkins's work does not address uncertainty and incomplete information, which must be addressed for bridge play. Second, Wilkins's work was directed at specific kinds of chess problems, rather than the problem of playing entire games of chess; in contrast, we have developed a program for playing entire deals of bridge.

Problem Characteristics

In our work, we consider the problem of declarer play at bridge. Our player (who may be a person or a computer system) controls two agents, declarer and dummy. Two other players control two other agents, the defenders. The auction is over and the contract has been fixed. The opening lead has been made and the dummy is visible. The hands held by the two agents controlled by our player are in full view of our agent at all times; the other two hands are not, hence the imperfect information. Bridge has the following characteristics that are necessary for our approach:

1. Only one player may move at at time.
2. In general, no player has perfect information about the current state S. However, each player has enough information to determine whose turn it is to move.
3. A player may control more than one agent in the game (as in bridge, in which the declarer controls two hands rather than one). If a player is in control of the agent A whose turn it is to move, then the player knows what moves A can make.
4. If a player is not in control of the agent A whose turn it is to move, then the player does not necessarily know what moves A can make. However, in this case the player does know the set of possible moves A *might* be able to make; that is, the player knows a finite set of moves M such that every move A can make is a member of M.

Our approach is applicable to any domain with these characteristics. Modifications of our approach are possible if some of these characteristics are missing.

Conclusion

By using techniques adapted from task-network planning, our approach to playing imperfect-information games reduces the large branching factor that results from uncertainty in such games. It does this by producing game trees in which the number of branches from each state is determined not by the number of actions an agent can perform, but instead by the number of different tactical and strategic schemes the agent can employ. By doing a modified game-tree search on this game tree, one can produce a plan that can be executed for multiple moves in the game.

An especially efficient reduction in the branching factor occurs when a agent plans a string of actions in succession that are all part of one strategic scheme; frequently, at a given point in time, only one action is consistent with the strategic scheme. Another important reduction occurs when an opponent is to follow to the play of a trick; the opponent's move is selected by finding the matching actions in the task network, and frequently there are only one or two matching actions.

Tignum 2, our implementation of the above approach, uses the above techniques to do card-playing for declarer in the game of bridge. In previous work, we showed that Tignum 2 performed better in playing notrump contracts than the declarer play of the strongest commercially available program; we have now shown that Tignum 2 performs better in playing **all** contracts (both notrump and suit).

We hope that the approach described in this paper will be useful in a variety of imperfect-information domains, possibly including defensive play in bridge. We intend to investigate this issue in future work. In addition, we have a number of observations about planning and game playing; these appear below:

Total-Order HTN Planning

Unlike almost all other HTN planners Tignum 2 is a total-order planner: in all of its task networks and methods—and thus all of the plans that it generates—the tasks are totally ordered. Also unlike most HTN planners, Tignum 2 expands tasks in the order in which they will be achieved: in choosing which task to expand in a task network, Tignum 2 will always choose the task that needs to be performed first.

We adopted the above approach because of the difficulty of reasoning with imperfect information. It is difficult enough to reason about the probable locations of the opponents' cards. If our task networks were partially ordered, then in many planning situations we wouldn't know what cards the opponents had already played. This would make reasoning about the probable locations of the opponents' cards nearly impossible; this reasoning is a more serious problem than the problems with uninstantiated variables that occur in perfect-information domains.

Being forced into total ordering, however, can be turned to our advantage. To provide a coherent framework for reasoning about partially-ordered plans, most current AI planning systems are restricted to manipulating predicates in order to decide whether to apply methods and operators. In Tignum 2 no such restriction is needed: to decide whether to apply a method, Tignum 2 can use arbitrary pieces of code. This gives us a mechanism for reasoning about the probable locations of the opponents' cards. In addition, these arbitrary pieces of code are often simpler than the predicates would be.

For example, consider a method that takes a finesse in a suit. Tignum 2 currently recognizes **nineteen** different kinds of finesse situations, such as Jx opposite KTx, xx opposite AJT, and x opposite AQ; one of these finesse situations must exist as a precondition for using the method. Using an arbitrary piece of code, it's easy to check whether one of these finesse situations exists in the suit, and then to apply the method, or not, as appropriate. If we were to use the method while leaving some of the variables in the precondition uninstantiated, and then later in planning try to achieve the precondition through the play of tricks earlier in the deal, we would have to decide which of the nineteen situations to try to achieve—and it wouldn't immediately obvious which of them were even possible to achieve.

We have been quite successful in applying Tignum 2's total-order HTN planning technique (as well as some of the same code used in Tignum 2!) to another application domain very different from bridge: the task of generating process plans for the manufacture of complex electro-mechanical devices such as microwave transmit-receive modules (Hebbar et al. 1996). That this same set of techniques should occur in two such widely varying areas is quite striking. In particular, we can make the following observations:

- HTN planning has long been thought to be more useful in practical planning domains than planning with STRIPS-style operators (Wilkins 1988), and our experience confirms this opinion. Bridge has a natural element of hierarchical planning. Humans use hierarchies of schemes to create plans to play bridge deals. The bridge literature describes many such schemes. Hierarchical planning gives each play a context; without such a context, one might search through many methods at each play. Hierarchical planning is also quite natural in process planning for complex electro-mechanical devices. In this case, the planning hierarchy derives naturally from the part-whole hierarchy of the device itself.
- Our experience also suggests that in a number of application domains, it may work well to develop plans in a totally-ordered manner, expanding tasks in the order that they are to be performed in the final plan.

  In AI planning, it is more common to expand tasks in some order other than the order in which they are to be performed. This way, the planner can constrain its search space by making some "important" or "bottleneck" decision before commiting to other less-important steps. For example, if I want to fly to another continent, it probably is better for me first to decide what flight to take, rather than how to get to the airport. By considering the "fly" task before the "get to airport" task, I can probably constrain the search space a great deal.

  However, deciding on a step that will come later in a plan before deciding on a step that will come earlier in the plan also incurs a drawback: when I am deciding on the later step, cannot know what its input state will be, because I have not yet decided what sequence of steps to use to produce that input state. This fundamental source of uncertainty can make the planning mechanisms much more complex than they would be otherwise.

  In both of the problem domains we have examined

(contract bridge, and process planning for microwave modules), there can be situations where it might be desirable to make a decision about a later step before making a decision about an earlier step in the plan—but in both domains, such situations do not seem to occur often enough to make it worth the trouble to put in the necessary planning mechanisms.

Planning in Games

As in most games, the plan existence problem in bridge is rather trivial; a sequence of legal plays is all that is required. We focus instead on the optimization problem: coming up with the best, or nearly the best, line of play. To do this, our planning procedure produced structures similar to game trees. Evaluation and selection of plans occured in these trees, and we learned that these structures seem to be a natural solution to the optimization problem. In addition, the wealth of developed efficiency improvements for game trees—such as transposition tables, adaptations of alpha-beta pruning, and the like—are available, although we have not yet implemented any of them.

Because our approach avoids examining all possible moves for all agents, it is related to the idea of forward pruning in game-playing. The primary difference from previous approaches to forward pruning is that previous approaches used heuristic techniques to prune "unpromising" nodes from the game tree, whereas our approach simply avoids generating nodes that do not fit into a tactical and strategic scheme for any player. Although forward pruning has not worked very well in games such as chess (Biermann 1978; Truscott 1981), our recent study of forward pruning (Smith & Nau 1994) suggests that forward pruning works best in situations where there is a high correlation among the minimax values of sibling nodes. Part of our motivation for the development of Tignum 2 is our belief that bridge has this correlation.

Some tasks and methods we added for suit play turned out to improve notrump play as well. For example, because discarding losers is much more a factor in suit play than it is in notrump play, it wasn't until we concentrated on suit play that some vulnerabilities in the discarding routines became clear. From this, we learn that broadening the focus of a task network can improve the task network on a narrower focus.

There are some pitfalls in suit play that didn't exist in notrump play. For example, to get a ruff, one might have to cross to another hand. One way to cross to that other hand might be in trump, which might use up the trump for the ruff. Once the trump is used up, there's no way to get it back. From this, we learn that so-called "white knights"—actions that make a false precondition, that was once true, true again—rarely arise in bridge.

References

Ballard, B. W. 1983. The *-minimax search procedure for trees containing chance nodes. *Artificial Intelligence* **21**:327–350.

Berlin, D. L. 1985. SPAN: integrating problem solving tactics. In *Proc. 9th International Joint Conference on Artificial Intelligence*, 1047–1051.

Berliner, H. J.; Goetsch, G.; Campbell, M. S.; and Ebeling, C. 1990. Measuring the performance potential of chess programs. *Artificial Intelligence* **43**:7–20.

Biermann, A. W. 1978. Theoretical issues related to computer game playing programs. *Personal Computing*, September 1978:86–88.

Boutilier, C.; Dearden, R.; and Goldszmidt, M. 1995. Exploiting structure in policy construction. In *Proc. 14th International Joint Conference on Artificial Intelligence.*

Collins, G. and Pryor, L. 1995. Planning under uncertainty: some key issues. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, 1670–1676. Morgan Kaufmann, San Mateo, California.

Currie, K. and Tate, A. 1985. O-Plan—control in the open planner architecture. BCS Expert Systems Conference, Cambridge University Press, UK.

Dean, T.; Kaelbling, L. P.; Kirman, J.; and Nicholson, A. 1993. Planning with deadlines in stochastic domains. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, 574–579. MIT Press, Cambridge, Massachusetts.

Draper, D.; Hanks, S., and Weld, D. 1994. Probabilistic planning with information gathering and contingent execution. In *Proceedings of the 2nd International Conference on AI Planning Systems*, Kristian Hammond, editor. AAAI Press, Menlo Park, California.

Erol, K.; Hendler, J.; and Nau, D.S. 1994. UMCP: A sound and complete procedure for hierarchical task-network planning. In *Proc. Second International Conf. on AI Planning Systems (AIPS-94)*, pages 249–254.

Erol, K.; Nau, D. S.; and Subrahmanian, V. S. 1995. Complexity, decidability and undecidability results for domain-independent planning. *Artificial Intelligence* **76**:75–88.

Frank, I.; Basin, D.; and Bundy, A. 1992. An adaptation of proof-planning to declarer play in bridge. In *European Conference on Artificial Intelligence.*

Frank, I. and Basin, D. 1996. Search in games with incomplete information: a case study using bridge card play. Under review.

Gamback, B.; Rayner, M.; and Pell, B. 1990. An architecture for a sophisticated mechanical bridge player. In Beal, D. F. and Levy, D.N.L., editors, *Heuristic Programming in Artificial Intelligence—The Second Computer Olympiad.* Ellis Horwood, Chichester, UK.

Gamback, B.; Rayner, M.; and Pell, B. 1993. Pragmatic reasoning in bridge. Tech. Report 299, Computer

Laboratory, University of Cambridge.

Ginsberg, M. 1996. How computers will play bridge. *Bridge World*, to appear.

Gmytrasiewicz, P. J. and Durfee, E. H. 1992. Decision-theoretic recursive modeling and the coordinated attack problem. In *Proceedings of the 1st International Conference on AI Planning Systems*, James Hendler, editor. Morgan Kaufmann, San Mateo, California.

Haddawy, P.; Doan, A.; and Goodwin, R. 1995. Efficient decision-theoretic planning: techniques and empirical analysis. In *Proceedings UAI95*, 229–236.

Hebbar, K.; Smith, S. J. J.; Minis, I.; and Nau, D. S. 1996. Plan-based evaluation of designs for microwave modules. ASME Design for Manufacturing conference, to appear.

Horacek, H. 1990. Reasoning with uncertainty in computer chess. *Artificial Intelligence* **43**:37–56.

Khemani, D. 1994. Planning with thematic actions. In *Proceedings of the 2nd International Conference on AI Planning Systems*, Kristian Hammond, editor. AAAI Press, Menlo Park, California.

Kushmerick, N.; Hanks, S.; and Weld, D. 1994. An algorithm for probabilistic least-commitment planning. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, 1123–1128. AAAI, Menlo Park, California.

Lee, K.-F. and Mahajan, S. 1990. The development of a world class othello program. *Artificial Intelligence* **43**:21–36.

Levy, D. and Newborn, M. 1982. *All About Chess and Computers*. Computer Science Press.

Lin, S.-H. and Dean, T. 1995. Generating optimal policies for Markov decision processes formulated as plans with conditional branches and loops. In *Third European Workshop on Planning*.

Lindelof, E. 1983. COBRA: the computer-designed bidding system. Victor Gollancz Ltd, London, UK.

Lopatin, A. 1992. Two combinatorial problems in programming bridge game. *Computer Olympiad*, unpublished.

Manley, B. 1993. Software 'judges' rate bridge-playing products. *The Bulletin* (published monthly by the American Contract Bridge League), **59(11)**, November 1993:51–54.

Pednault, E. P. D. 1987. Solving multiagent dynamic world problems in the classical planning framework. In *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop*, 42–82. Morgan Kaufmann, Los Altos, California.

Peot, M. and Smith, D. 1992. Conditional nonlinear planning. In *Proc. First Internat. Conf. AI Planning Systems*, 189–197. Morgan Kaufmann, San Mateo, California.

Quinlan, J. R. 1979. A knowledge-based system for locating missing high cards in bridge. In *Proc. 6th International Joint Conf. Artificial Intelligence*, pp. 705–707.

Sacerdoti, E. D. 1977. *A Structure for Plans and Behavior*. American Elsevier Publishing Company.

Samuel, A. L. 1967. Some studies in machine learning using the game of checkers. ii–recent progress. *IBM Journal of Research and Development* **2**:601–617.

Schaeffer, J.; Culberson, J.; Treloar, N.; Knight, B.; Lu, P.; and Szafron, D. 1992. A world championship caliber checkers program. *Artificial Intelligence* **53**:273–290.

Smith, S. J. J. and Nau, D. S. 1994. An analysis of forward pruning. In *Proc. 12th National Conference on Artificial Intelligence*, pp. 1386–1391.

Smith, S. J. J.; Nau, D. S.; and Throop, T. 1996. A planning approach to declarer play in contract bridge. *Computational Intelligence*, **12:1**, February 1996, 106–130.

Stanier, A. 1975. Bribip: a bridge bidding program. In *Proc. 4th International Joint Conf. Artificial Intelligence*.

Sterling, L. and Nygate, Y. 1990. Python: an expert squeezer. *Journal of Logic Programming* **8**:21–39.

Tate, A. 1977. Generating project networks. In *Proc. 5th International Joint Conf. Artificial Intelligence*.

Truscott, T. R. 1981. Techniques used in minimax game-playing programs. Master's thesis, Duke University, Durham, NC.

Wilkins, D. E. 1980. Using patterns and plans in chess. *Artificial Intelligence* **14**:165–203.

Wilkins, D. E. 1982. Using knowledge to control tree searching. *Artificial Intelligence* **18**:1–51.

Wilkins, D. E. 1984. Domain independent planning: representation and plan generation. *Artificial Intelligence* **22**:269–301.

Wilkins, D. E. 1988. *Practical Planning*. Morgan Kaufmann, San Mateo, California.