Multipurpose Strategic Planning In the Game of Go

Shui Hu and Paul E. Lehner

Abstract—A heuristic method for adversarial planning is developed to address the problem of multipurpose planning in the game of Go. Static analysis and dynamic look ahead on both strategic and tactical levels are used to generate possible goals and to identify interactions among the achievability of various goals. Strategic, multipurpose goals are composed of sets of interacting goals.

Index Terms—Computer Go, adversarial planning, automated planning, artificial intelligence.

1 Introduction

Go is a board game that originated in China about 4,000 years ago. It is an alternating move game like Chess, but differs from chess in its main objective (occupy territory rather than capture a piece) and in the size of search space (substantially larger than Chess). Others have noted (e.g., [14]) that there are many elements of Go that are found in our daily lives. In particular, success in Go requires the simultaneous pursuit of multiple goals in an adversarial context. During a game, a player may need to address several "urgent demands" with limited resources (only one stone per turn).

Computer Go requires both strategic and tactical planning. Strategic planning determines the goals to pursue and the general pattern for pursuing those goals. Tactical planning is concerned with the detailed selection of specific actions that best implement a chosen strategy. Although the necessity of strategic perception and planning for effective play is widely recognized, current Go programs demonstrate a lack of strategic focus, which is reflected in an apparent lack of direction in their play. In this paper we present an approach to integrated strategic and tactical planning in Go.

Our approach is founded on the concept of a multipurpose move—a move that supports multiple goals. Other researchers have discussed and implemented multipurpose planning in Go by trying to find mutivalued moves ([1], [6], [9], [10]). To find a multivalued move, candidate moves are generated for the current board position, each move is assigned a value which represents its potential gain, and if a move is generated by more than one generator, it has multiple values. Although useful, this approach to multipurpose planning is passive and can not find moves that take advantage of interactions among moves (e.g., where one move changes the value of another move); or when one move intentionally creates a position which makes a follow-on multipurpose move feasible. A more proactive approach to multipurpose planning is needed.

2 General Description of Our Approach

Static analysis and dynamic look ahead are two major techniques used in computer Go. Static analysis extracts and analyzes game information from a current position. Dynamic look ahead creates a

- *S. Hu is with Hughes Information Technology Systems, 1616 McCormick Dr., Upper Marlboro, MD 20774. E-mail: whu@eos.hitc.com.*
- *P.E. Lehner is with The Mitre Corporation, 1820 Dolly Madison Blvd., McLean, VA 22101-3481. E-mail: plehner@mitre.org.*

Manuscript received 29 Apr. 1996; revised 9 June 1997. Recommended for acceptance by T. Ishida.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 105312.

0162-8828/97/$10.00 © 1997 IEEE

sequence of possible future positions to analyze the results of applying specific strategies and tactics. In our approach, static analysis is used to identify possible goals from a current board position. Then, where necessary, dynamic tactical look ahead is used to determine the major consequences of a proposed goal by examining some possible move sequences [11]. The results of static analysis and dynamic look ahead are then merged to estimate the achievability of various goals as well as goal interactions, where a move to achieve one goal impacts the achievability of another goal. The interactions are used to compose alternative strategic multipurpose goals that coordinate the pursuance of individual goals. The value (territory gain) of each strategy is then estimated. The highest value strategy is selected and a detailed tactical look ahead is performed to determine the best move for implementing the selected strategy.

3 More Details on Approach

3.1 Game and Knowledge Representations

As in most computer Go programs (e.g., [5], [8]), we use object hierarchies to model the game. The base level object is a stone, sets of stones form strings, sets of strings form chains, and sets of chains form groups.

Current Goal Structures (CGS) are used to characterize the hierarchy of goals that may be pursued in a given position. As illustrated in Fig. 1, a CGS is a multiple level goal tree that is composed of a common top level goal, Occupy Territory; strategic subgoals that represent position-dependent ways to pursue the top goal, and a bottom level of tactical goals that are used to generate possible moves.

Fig. 1. Example of Current Goal Structure.

A strategic goal's target or owner (if any exist) is always a group. "Owner group" means that a goal is "requested" (described below) by the group. Some goals have more than one owner group, which means that the goals are requested by multiple groups and merged together. Some strategic goals do not have an owner, which means that the goals are requested by the master goal, Occupy Territory, instead of a group. A tactical goal's target or owner could be any object except a stone. Tactical goals are a subclass of goals that inherit all of the attributes from the goal class, but have an extra attribute for actions (moves). A knowledge base of goal definitions is used to control the generation of goals and moves. Figs. 2 and 3 give abstract descriptions of a strategic and tactical goal definition.

Goal: Defend weak group
Subgoals: **if** neighbor enemy group is weak
then counter attack
else if my weak group has freedom
then escape
else if my weak group has room for eyes
then survive

Fig. 2. Example of strategic goal definition.

When planning is initiated, strategic level goals are generated by a static assessment of the current position. For instance, if an enemy group's safety value is below a certain threshold, an Attack Group goal will be requested. The goal definitions for the strategic goals are then invoked to generate tactical subgoals that implement the strategic goal (e.g., ladder attack, cut attack, ...). After the tactical goals are generated, the achievability of those goals is evaluated. Depending on the position, this may be done by either static analysis or look ahead. For example, when the escape route of a ladder attack's target string is unobstructed, a static analysis is sufficient to deduce whether the ladder attack will succeed. When the escape route is obstructed by a mixture of black and white stones a look ahead is launched. If the result shows that a goal is unachievable it is pruned from the CGS.

Goal:	Ladder Attack (or Ladder Escape)
Subgoal:	(Procedure to generate moves according to current position)
Failed:	(Procedure to assess the goal's Failure)
Succeeded:	(Procedure to assess the goal's Success)

Fig. 3. Example of tactical goal definition.

The look ahead control algorithm is [12] which is derived from, and very similar to, the [4] tactical search algorithm. It begins with an initial goal for the planning agent and a hypothesized goal for an adversary. Beginning with the initial goal, CP2 generates a goal stack by recursively generating subgoals until a bottom level action is generated (which may be no action if it's the other player's turn). A planner-adversary goal stack pair defines the next hypothetical position. In each hypothetical future position, the goal stacks that led to that position (from the previous position) are processed to

1) determine if any of the goals in the planner or adversary stack have succeeded or failed and if not then,
2) remove any subgoals that are invalid (e.g., the goal Move-to(LocA) is invalid if LocA is already occupied),
3) generate new goal stacks by recursively generating subgoals from the lowest valid subgoal in the stack.

If a position is marked as success for either side, then CP2 records this and backs up to a previous position to explore the other possible goal stacks.

Based on the results of the static analysis or look ahead, an evaluation function is used to give each tactical goal a value that estimates how many potential territory points may be obtained by pursuing that goal.

3.2 Goal Achievability, Relationships, and Composition

After [2], [13] we define multiple levels of achievability of a goal. These are shown in Fig. 4. A goal is achievable if the player can make a move that guarantees (after play and counter play) eventual attainment of the goal. It is under achievable if a player must make two moves in succession to guarantee attainment, over achievable if a player can let the opponent move first and still guarantee attainment, under always achievable if he has two moves to give, etc.

Relationships may exist between some goals. Often this rela-

tionship is established when these goals are generated. If a goal B is generated because it promotes the attainment of another goal A, then goals A and B have a Master and Servant relationship. This relationship is used in the proactive multipurpose planning mechanism described later.

	Never	Under Never	Under	Achievable	Over	Under Always	Always
#Moves:	infinity	3	2	1	-1	-2	-infinity

Fig. 4. Levels of achievability. Negative numbers mean the opponent's moves.

An individual goal has just one purpose and one target. A composed goal consists of more than one individual goal. We consider three types of composition:

- ANDSIM (And Simultaneous): Conjunction of Achievable goals
- ORSIM (Or Simultaneous): Conjunction of Under Achievable goals
- ANDINI (And Initiative): Conjunction of Achievable and Under Achievable goals

3.3 Planning Procedure

Our planning procedure consists of eight steps.

- Step 1. Generate CGS by static analysis.
- Step 2. Evaluate the achievablilty of each goal that is a leaf of the CGS with static analysis or look ahead. For each Under Achievable goal, a servant goal is generated to support the attainment of the Under Achievable goal.
- Step 3. Process any servant goals and validate that an attempt to attain the servant goal will promote attainment of its master goal.
- Step 4. Calculate the potential gain of each tactical action in the leaf of the CGS.
- Step 5. Propagate the tactical action's achievability and its gain to its parent goal.
- Step 6. Construct a decision table by composing ANDSIM, ORSIM, and ANDINI goals.
- Step 7. Select as the planner's strategy the single or composed goal with the highest potential gain.
- Step 8. Perform detailed tactical search to select the best move to implement selected strategy.

From the steps above, we can see the integration of strategic and tactical planning. The planning procedure iterates between strategic and tactical level reasoning. In this mechanism, the strategic view is a part of every move.

4 Example

A prototype Go program was implemented to evaluate our approach to multipurpose planning in Go. Our general approach to evaluation was to develop a limited tactical knowledge base that would be adequate to support strategic multipurpose planning for some interesting and difficult Go positions. A complete knowledge base was not implemented since this would have been well outside the scope of our research effort, and would not have contributed substantially to our research objective. As will be seen below, the performance of our multipurpose planner is very strong on these selected problems, which we believe constitutes evidence for the general effectiveness of our approach.

We present an ORSIM example from our Go program. An ORSIM planning example was selected because it is relatively easy to explain. The most interesting and technically challenging examples are ANDINI planning, where the program tries to maintain initiative by finding moves that achieve one goal while threatening to achieve another one - thereby forcing the opponent to respond defensively and denying him the opportunity to initiate a new attack. The reader is encouraged to examine [15] for a detailed example of ANDINI planning.

In ORSIM planning, the planner looks for a move that supports two or more Under achievable goals, where no counter measure can be found for the opponent to prevent the attainment of all goals. The gain for an ORSIM goal is the minimum gain of individual Under Achievable goals inside the composed goal set.

Consider Fig. 5. A potentially desirable move for black would be a ladder attack to String-15. But this is not feasible because of white String-2 and String-6 on the lower left corner. With ORSIM planning, a clever solution can be found.

Fig. 5. ORSIM Example. Original example from [3]. Black's turn. The numbers on the stones are string identifiers.

At step 1, a CGS corresponding to this position was generated by static analysis (see Fig. 6). In this CGS, we can see the system tries to attack String-15 with a net attack and a ladder attack in goals 10 and 11 respectively. At step 2, all of the goals' achievability were evaluated by static analysis and tactical look ahead. Both Goal-10 and Goal-11 are identified as Under Achievable. In the look ahead for the ladder attack on String 15 (Goal 10), an obstacle String-6 was found. Therefore, a supporting goal was generated to attack Group-2. The purpose of the supporting goal is to promote the attainment of Ladder Attack on String 15. At this time, no specific tactic was determined. The strategic Goal-12 Attack Group-2 is added to the CGS tree. This goal is Goal-10's servant. At step 3, the servant goal was processed and a subgoal was generated to attack the obstacle String-6. A Cut attack was selected by static analysis. After look ahead, the Cut goal was identified as Under Achievable.

A validation function was then called to validate that the Cutting Attack on String 6 supports the Ladder Attack on String 15. This process tries to find a conjunction between Master and Servant goals. The "cutting" tactic was played during look ahead. Two alternative moves were found to cut the connection to white String-6. After the opponent made a hypothetical response to each cutting move, the look ahead of the ladder attack, from the right upper corner, was launched again (in the hypothetical position). This time the ladder attack became Achievable because of the cutting move played in the previous turn. Of course, white had another move choice: extend String-15 to prevent the ladder attack

on the upper right corner. The corresponding look ahead is validated and added to the CGS tree.

At this point the planner knows that a beneficial synergy of two Under Achievable goals exists. The action D16, created by cutting, should either cut the connection between String-6 and String 2 or makes the ladder attack achievable. This is a proactive method for multipurpose planning. At steps 4 and 5, the values were calculated and propagated for each strategy. Finally, the CGS is completed (see Fig. 1). At step 6, a decision table was created by composing multipurpose goals. In this case, one of the options is to pursue the composed goal ORSIM (Goal 5, Goal 12). Assuming the worst, the value of an ORSIM goal is the minimum of the component goals. In this case, its value is 23 from the ladder attack rather than 25 from the cutting attack.

An interesting note, if an Over Achievable goal is on the CGS, the planner must consider the possible negative impact on the Over Achievable goal from the chosen strategy. Although an Over Achievable goal is rarely selected for pursuit, a selected strategy should avoid demoting an Over Achievable goal to Achievable; or the caused demotion may give the opponent a possible multipurpose move. If a demotion is detected by a look ahead, it means that the planner might encounter an opponent's ORSIM goal in its pursuit of the chosen strategy. The reviewing of this contingency must be performed before the selection of the strategy.

Fig. 6. CGS of ORSIM example: After Step 1.

5 Conclusion

In this paper, we presented an approach to multipurpose planning in Go. Our approach exploits interactions among various individual goals to find strategic multipurpose goals. Our approach performed well on selected problems. With a more complete knowledge base, we believe it would continue to perform well as part of a general computer Go program.

Regarding computational complexity, our approach requires a number of localized searches to be performed across the entire board at both the strategic and tactical levels. At first, this may appear to be computationally unrealistic. However, the opposite is true. During strategic level analysis we use only static analysis and small beam-width searches to learn about interactions across the board. We then use the results of strategic planning to carefully focus tactical analysis, thereby limiting the amount of high beam-width search that occurs at the tactical level. We note further that, if needed, search breadth and depth can be artificially limited as, for instance, was done by [4]. This is not to say that our approach is not computationally complex, but only that it is no more complex than other reasonable knowledge-based approach.

References

[1] J.L. Ryder, "Heuristic Analysis of Large Trees as Generated in the Game of Go," PhD thesis, Stanford Univ., 1971, Microfilm 71-03, pp. 162.

[2] D.B. Benson, "Life in the Game of Go," *Information Sciences*, vol. 10, pp. 17-29, 1976. Reprinted in [7].

[3] T. Kageyama, *Lessons in the Fundamentals of Go.* The Ishi Press, 1978.

[4] W. Reitman and B. Wilcox, "The Structure and Performance of the INTERIM.2 Go Program," *Proc. Int'l Joint Conf. Artificial Intelligence*, pp 711-719, 1979. Reprinted in [7].

[5] K.J. Friedenbach, "Abstraction Hierarchies: A Model of Perception and Cognition in the Game of Go," PhD thesis, Univ. of California, Santa Cruz, 1980, Microfilm.

[6] B. Wilcox, "Computer Go" *Amer. Go J.*, vol. 13, nos. 4, 5, and 6; vol. 14, nos. 1, 5-6; vol. 19, 1978, 1979, and 1984). Reprinted in [7].

[7] D. Levy, ed., *Computer Game,* vols. 1 and 2. New York: Springer-Verlag, 1988.

[8] K. Chen, "Group Identification in Computer Go," *Heuristic Programming in Artificial Intelligence*, D. Levy and D. Veal, eds., pp. 195-210, Ellis Horwood, Fall 1989.

[9] K. Chen, A. Kierulf, J. Nievergelt, and M. Muller, "The Design and Evolution Go Explorer," *Computer, Chess, and Cognition*, T. Marsland and J. Schaeffer, eds., pp. 271-285. Springer-Verlag, 1990.

[10] A. Kierulf, K. Chen, and J. Nievergelt, "Smart Game Board and Go Explorer: A Study in Software and Knowledge Engineering," *Comm. ACM*, pp. 152-166, Feb. 1990.

[11] P.E. Lehner, "Strategic Planning in Go," *Computer Game Playing: Theory and Practice*, M. Bramer, ed. Ellis Harwood Ltd., 1983.

[12] P.E. Lehner, "Automated Adversarial Planning Search Procedures With Provable Properties," *Advanced Technology for Command and Control Systems Engineering*, S. Andriole, ed. Fairfax, Va.: AFCEA International Press, 1990.

[13] R. Popma and L.V. Allis, "Life and Death Refined," *Heuristic Programming in Artificial Intelligence 3*, J. van den Herik and L.F. Allis, eds. Ellis Horwood, 1992.

[14] Y. Miura, *Go: An Asian Paradigm for Business Strategy*, Kiseido, 1995.

[15] S. Hu, "Multipurpose Adversary Planning in the Game of Go," PhD thesis, George Mason Univ., 1995, Microfilm 9608536.