Adversarial Planning in Complex Domains

Steven Willmott[1], Alan Bundy[2], John Levine[3], Julian Richardson[4]

Abstract. Most current planning research relies on the assumption of a benign domain containing only co-operative agents. There has been little work on applying modern planning techniques to adversarial domains where this assumption does not hold. This paper discusses the development of an HTN Adversarial Planning architecture and the issues involved in successfully applying such a goal driven approach in complex domains. The architecture was tested by using it to build a reasoning system name GoBI able to plan for Go problems of the types commonly found in teaching books. GoBI only has limited knowledge but still performs well on a significant subset of Go problems.

The Results also opened up interesting new approaches to tackling the problem of programming for Go but this paper concentrates on the planning aspect.

1 Introduction

This paper addresses the key issue of goal decomposition schemes for HTN Adversarial Planning and suggests that the knowledge of relationships and interactions between goals is the key influence in deciding upon a decomposition scheme. We describe an approach which uses "early grounding" to cope with these interactions, which in an adversarial domain are generally very significant. We also highlight the possibility of adding data driven aspects into such a system which may be essential in some domains (Go almost certainly being one).

The planning architecture performed well when it was applied to the game of Go, a very complex game-playing domain. This demonstrates that the HTN Adversarial Planning approach can be useful for such hard problems.

Section 1.1 introduces HTN (Hierarchical Task Network) Adversarial Planning, outlines the general approach adopted here and presents Go as a complex test domain. Section 2 describes the form of an adversarial plan. Section 3 gives an overview of the planning architecture. Section 4 focuses on the goal decomposition schemes which are the heart of the planning system. Sections 5 and 6 give results of applying the architecture to Go followed by conclusions.

[1] Laboratoire d'Intelligence Artificielle, École Polytechnique Fédérale de Lausanne, Lausanne. e-mail: *willmott@lia.di.epfl.ch*
[2] Department of Artificial Intelligence, University of Edinburgh. e-mail: *bundy@dai.ed.ac.uk*
[3] Artificial Intelligence Applications Institute, University of Edinburgh. email: *johnl@aiai.ed.ac.uk*
[4] Department of Artificial Intelligence, University of Edinburgh. e-mail: *julianr@dai.ed.ac.uk*

Submitted to ECAI 98
January 20, 1998

1.1 Adversarial Planning

Most approaches used in adversarial domains (currently mostly game domains) use the rules and legal operators in each state to generate possible actions and then apply evaluation functions to the successor states, picking the action leading to the best outcome. This results in a state based tree search and is analogous to STRIPS style planning (Fikes and Nilsson [7]).

HTN Adversarial Planning uses a goal driven approach: taking abstract goals in the domain and decomposing these to lower and lower levels of abstraction to determine which primitive actions attain the goals. An example goal[5] might be *kill-group*, satisfiable by an abstract plan made up of the subgoals: *surround-group* (stop it running and connecting), *reduce-space* and *prevent-defensive-formation*. This approach has two main advantages:

1. Removing the need for enumerating all possible actions in each state, which must be done in any search based approach. The possible goals and aims in the domain therefore determine the size of the space to be searched - not the number of options at each turn.

2. Reducing the complexity of evaluation functions. In state based search each state must be evaluated according to its desirability which in some domains may depend upon many different criteria. Using a goal driven approach the complexity of evaluation functions to do this can be greatly reduced since the problem reduces to checking if low level goals (in terms of abstraction) in the plan can be satisfied.

In domains with a large number of possible actions at each step and states which are hard to evaluate the goal driven approach becomes more and more attractive.

Early work on Adversarial Planning includes [10], [13], [4] and [9]. There has however been little recent work in this area[6] - possibly because, alongside its benefits, the approach also presents the following long standing problems which must be addressed:

1. **Complexity:** The HTN Adversarial Planning problem is inherently more complex than standard planning. A plan can no longer be seen as a sequence of actions with a successful outcome but must be a tree of contingencies for all possible orderings of the opponent's actions[7]. Any

[5] Taken from the Go domain.
[6] Exceptions to this are [1], [16] and [12]
[7] The structure of an adversarial plan is described in Section 2.

system using this approach must therefore be able to control this work load effectively to stand a chance of being useful.

2. **Modelling abstract knowledge:** It often turns out to be difficult to represent the abstract knowledge in the domain well enough to be of use.

This paper identifies some key features of the adversarial planning problem which give rise to these difficulties and presents an approach to addressing them. The ideas and insights were incorporated into a hierarchical planning system which models two agents interacting over a series of moves. The architecture was tested by applying it to the game of Go which is becoming a worthy successor to Chess as a challenging testing ground for AI techniques. The Go reasoning system's performance was promising, demonstrating that this approach can be successfully applied in complex domains.

1.2 "Early Grounding"

The architecture described here differs from a standard HTN planning scheme (such as the one given in [6]) in several ways. One is that it models two agents interacting. Another is that it uses linearisations[8] during goal decomposition and decomposes some goals to the lowest level of abstraction as soon as possible. This allows a world model to be used to take care of modelling the complex interactions in the domain. Furthermore the world model can also be used to add data driven aspects to the system - which may be essential for some domains.

[12] notes that enforcing a total-order on an HTN system for Bridge declarer play is very useful in controlling the combinatorial explosion caused by the uncertainty in the domain. Linearisations are also used in [1] which presents an architecture for battle planning mixing execution and plan generation (this necessarily requires linearisations).

In this paper we go on to suggest that a total-order "early grounding" strategy is in fact essential for successful HTN Adversarial Planning in most complex domains - whether uncertainty is involved or not.

1.3 Playing Go with Plans

The rules of Go are very simple yet the depth and complexity arising from the myriad of different tactics and strategies for achieving the overall goals is astounding. The size of the search space is estimated to be 10^{170} states[9], there are approximately 300 moves per game[10] and the branching factor at each turn is approximately 235 states[11]. Therefore Go presents a stern challenge to computational approaches and a challenging domain for the development and evaluation of an Adversarial Planning architecture. The huge search space and complex game states make a goal driven approach potentially very useful for Go. More details of current, mainly search based, approaches can be found in [3] and [5].

[8] Thereby making it a total-order planning system.
[9] Compare chess at $\approx 10^{50}$.
[10] Compare chess at ≈ 80.
[11] Compare chess at ≈ 35.

2 An Adversarial Plan

Before going any further it is useful to outline what kind of result one would like from an HTN Adversarial Planning system.

The plan need only have a desirable outcome for one agent (named the *calling* agent[12]) in the system, in fact assuming opponents have conflicting goals the outcome will probably be *undesirable* for all the opponent agents in the system.

A complete adversarial plan is a tree of actions (primitives) in the world which represents a *contingency plan* for the calling agent. The root is the first action of the calling agent and the branches represent the counter-plans of the opponents in the system with corresponding follow up actions of the calling agent.

- The plan is *valid* for the calling agent if the leaves of all branches result in states in which the calling agent's top level goals have been achieved.

- The plan is *complete* if it represents all the feasible counter-plans which can be generated by the models of the opposing agents in the system for the current situation.

Thus given a situation and a calling agent it would be desirable for the planner to return a valid, complete plan for the situation. The tree acts as a justification of the effectiveness of the first action in the tree[13].

3 An Overview of The Planning Architecture

The scheme described here is a generic architecture for two agent, turn based, complete information domains.

The architecture models two agents (named Alpha and Beta) which pass partially formed plans between them. Figure 1 shows the flow of control during planning: both agents use the same mechanisms but only one is in control at any one time. The primitives are actions in the world and each takes up one "turn" in the world (this models turn based domains), so as Figure 1 shows the agents take it in turns to act.

The system is modelled on hierarchical task network (HTN) planning systems ([6],[11]) which use the notion of abstract goals decomposing to lower and lower levels of abstraction before reaching the level of concrete actions in the world. These levels of abstraction allow reasoning *above* the level of individual actions. Actions in the world are only considered if the system thinks they may be useful in satisfying a goal, i.e. - in the context of a plan of action. This greatly reduces the problem of having a huge number of choices at each step. Goals at the lowest level of abstraction are usually simple, thus checking if they are satisfiable is generally much easier than trying to judge the comparative value of an entire state (which is what must be done in search based systems).

The framework allows the expression of both object-level and (through the use of critics) meta-level knowledge about the planner's own plans and the adversary's plans. There are

[12] This is the agent modelled in the planner to represent the point of view of the agent in the system which *called* for a plan to be formed (in the system described here this is always agent Alpha).
[13] The contingency tree could also be used to play out subsequent moves; however this touches on issues related to deciding when the plans need to be updated which were not investigated.

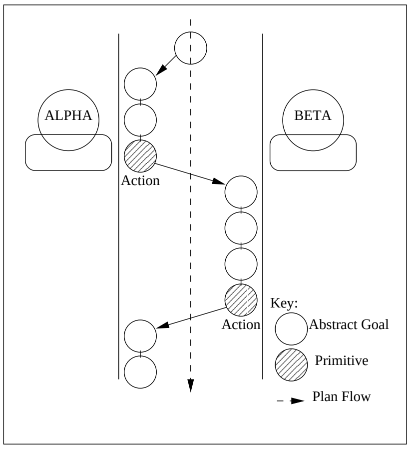

Figure 1. Planning steps alternating between two agents.

also no restrictions on orderings between the goals as found in [16] which forces the goals and subgoals of the adversaries to be paired off with one another (this was found often to be too restrictive when expressing knowledge). The system therefore dynamically manages the turn ordering between the agents as goals are decomposed and it becomes clear varying numbers of primitive actions are needed to fulfill them. Goal decompositions can also be checked against pre-conditions which indicate whether they might be useful in the current situation.

4 Goal Decomposition Schemes

As mentioned above, HTN planning works by decomposing goals into sets of subgoals (called schemas in [6]) at lower and lower levels of abstraction. An HTN planning system therefore needs a scheme to control the order of decomposition and shape of the plan at each level of abstraction. It is this decomposition scheme which defines the fundamental characteristics of a planning system.

4.1 Standard HTN Goal Decomposition

The standard scheme for HTN decomposition (used to good effect in non-adversarial domains) is as follows[14], at each step:

1. Choose a goal to decompose.
2. Select a promising decomposition schema.
3. Substitute this schema into the plan in place of the decomposed goal.
4. Return to step 1).

In general all goals are kept at roughly equal levels of abstraction. The average abstraction level of the goals in the plan then gradually decreases as the goals are decomposed, eventually reaching the stage where all goals are concrete actions in the world (primitives) and the plan can be executed.

At each level of abstraction (in fact after each decomposition) the relationships and interactions between goals can be used to guide the choice of schemas and (in a partial order scheme) constrain available orderings.

4.1.1 Applying Standard HTN Goal Decomposition in Adversarial Domains

When planning in an adversarial domain, there are two sources of search which we must deal with:

1. An adversarial plan is in general a *tree* of actions which represents the possible counter-plans of opponents and contingencies for them. Since plans should take into account all of the opponent's counter-plans (i.e. to be complete in the sense described in Section 2), we are obliged to perform search to refute each of these counter-plans. By contrast, in standard HTN planning, a plan is a sequence of actions (possibly partially ordered).

2. The refinement of an abstract plan to a concrete plan can fail when some knowledge about the domain is missing from the abstract plan operators. In any complex domain, such deficiencies in the abstract operators are almost inevitable. In an adversarial domain (such as Go), the situation is even worse — the game environment is *designed* to generate complex interactions between low level actions in the world, designed in fact to frustrate, deceive and entertain human players.

The first of these two kinds of search is an essential feature of the domain and cannot be avoided. It can also be difficult to keep track of the search (and the solution plan) when operators are being applied at a number of different levels of abstraction.

The second kind can be avoided to some extent by ensuring that the abstract operators encode the information necessary to reason effectively about the domain and about possible goal interactions, particularly those involving the other agents. Unfortunately it is hard to foresee and represent such goal interactions. For example two seemingly unrelated goals such as *X: Keep knight pinned* and *Y: Queen a pawn* may interact at the lowest level in the plan. This might happen if the pawn must advance into the line of sight between the pinning piece and the knight on its way to the edge of the board. Another chess example might be two pieces needing to occupy the same square to achieve completely different goals.

As the level of unmodelled interactions between abstract goals increases, it becomes less and less likely that abstract solutions are refinable into real solutions. This dramatically increases the amount of work wasted on exploring refinements of abstract plans which turn out to be useless. The detrimental effect of increased backtracking between abstraction levels is well documented in work on the advantages of abstraction hierarchies in planning such as that in [8] and [2]. If we attempt to apply standard HTN planning techniques in such an adversarial domain, then the lower the proportion of interactions that can be represented at higher levels of abstraction the more the standard approach tends to developing *two independent detailed plans* (one for each agent) and then trying to merge them by interleaving their actions. This merging step is likely to fail since the opposing agents actively work to increase problematic interactions.

[14] This is an outline, simplified version - more details can be found in [6].

There are therefore two main problems with the standard HTN decomposition scheme when applied to adversarial domains:

- Keeping track of the solution plan during planning as backtracking moves between levels of abstraction. It is particularly important to keep a tight enough control to avoid redundancy in search.

- Successful planning using this scheme depends critically on abstract knowledge of the *interactions* between goals at each level. As knowledge of this decreases, the amount of wasted work rapidly increases.

4.2 "Early Grounding HTN"

In view of these difficulties one key feature of the architecture developed here and a way in which it differs significantly from a standard HTN approach is in the use of an "early grounding" strategy in goal decomposition. The system attempts to make goals concrete as soon as possible and models the effect of these actions in a "world model". In outline the scheme works as follows:

1. Choose a goal to decompose.
2. *Mark this goal to be next in the order of actions.*
3. Select a promising decomposition schema *(taking into account in the current world model).*
4. Substitute this schema into the plan in place of the decomposed goal.
5. *Choose one of the new goals added into the plan by the decomposition.*
6. - IF this goal is primitive THEN *perform the action in the world model* and return to step 1).
   - ELSE decompose this goal further by returning to step 2).

The significant differences from the standard HTN decomposition scheme are marked in italics. Steps 2 and 5 linearise the plan by selecting a single goal to decompose first[15]. The second main difference is the acting out of all primitive actions in the world model (step 6). This allows each goal in turn to be decomposed in the context of its *final* place in the plan (step 3).

Figure 2 shows goal decomposition for both a standard HTN model and the "early grounding" model. The dashed boxes represent the items each system can reason about when decomposing the goal marked X.

- In the standard HTN model (lower diagram) the planner can reason about all the goals at their current level and about the world in its *initial* state. The effects of actions subsequent to the initial state must be modelled explicitly by the plan operators.

- In the upper diagram the decomposition progresses left to right in time order: here the planner can reason about the outstanding abstract goals and about the *current* state of the world[16]. The first two primitive actions (one chosen by Alpha and one by Beta) are now represented in the world model and their effect on the state of play clearly seen.

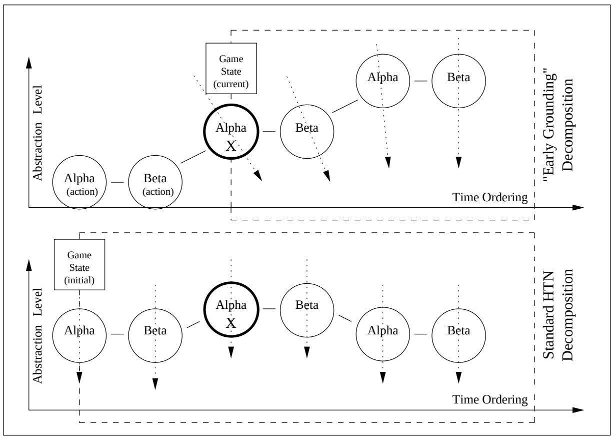

Figure 2. Goal decomposition in "pure" HTN planning and in the formalism presented here.

The "early grounding" decomposition scheme allows the use of a world model to simulate[17] the complex effects of the previously decomposed goals, providing an up to date context for the decomposition of X.

4.2.1 A Chess Example

Let us return to the Chess example given in Section 4.1.1 with two goals *X: Keep knight pinned* and *Y: Queen a pawn.*

In the standard HTN decomposition scheme an abstract plan containing both of these goals would, in the absence of detailed knowledge about their possible low level interaction, be decomposed further and further towards the concrete level (along with the other goals in the plan). Hence many refinements of the sub-plan could be explored and each fail because of the interaction between these two goals (the pawn interrupting the pinning of the knight).

In the "early grounding" scheme one goal (*Y* say) would be decomposed first, leading to one or more moves in the world model. *X* is then decomposed in the context of this updated world model. Once *X* reaches the lowest level of moving pieces around it will be clear that moving the pawn will interact with the pinned and pinning pieces.

The world model therefore immediately shows that these two goals interact with each other in this situation and may cause the whole plan to fail.

4.2.2 Adopting "Early Grounding"

In outline the scheme has the following main advantages:

- It essentially allows the system to make choices for every decomposition step which are "feasible" w.r.t all the decompositions already made. This removes much of the

[15] This approach necessarily leads to a total-order planner. At each stage there can be goals of all abstraction levels in the plan but there must also be a linearisation and goals are decomposed in time order.

[16] The dashed box can be enlarged again upon backtracking.

[17] The simulation can be perfect in a complete information domain (such as Go).

dependence on having accurate abstract representations of all important interactions. Early grounding allows effective planning using incomplete plan operators.

- The method of decomposition models the adversarial planning problem very well. As the system chooses actions and backtracks it automatically creates the tree structure of the contingency plan. As noted above this would have to be explicitly modelled in the standard HTN decomposition scheme.

The price paid for these plus points is:

- A reduction in scope for reasoning about the interactions between goals at each abstract level.

- A requirement for the planner to be total-order.

In fact it would also seem that as the amount of knowledge about goal interactions decreases the value of having a partial-order system and of having full flexibility of reasoning at each abstraction level also falls.

Together with the use of a world model the two restrictions above mean that the architecture is no longer pure HTN in the strictest sense. Choices of actions still come about in a top down, goal driven way but the system is less reliant on the representation of every important interaction. The "early grounding" approach can be compared to using abstract plans to guide search, an approach first proposed in [10] and the successful PARADISE system for chess ([13])[18].

4.2.3 Using Opportunities

As the plan is developed the changing situation is reflected in the world model. This could highlight interesting opportunities (or obvious problems) which arise as a result of the plan actions but were not expected effects. The architecture developed includes the possibility of adding critics which have access to the world model and are allowed to insert goals into the current agenda if opportunities or problems arise. These new goals are things for the planner to consider *alongside* the current track of reasoning. The critics represent a data driven aspect inside a goal driven framework

An example might be a chess playing system with a critic that knew how to find current one-move mates. The planner may have a good plan of attack but have plan knowledge falling short of suggesting the final killing move and hence the system would normally decide that the plan had failed and try another approach[19]. With the critic switched on the strong positions from the good attack can be checked to see if there are opportunities for mate and the planner told about them.

This effectively increases the search space but makes the system more robust in situations where the a priori plan knowledge is not enough. This can be seen as using the abstract plan knowledge to guide search into promising areas and allowing the system to have its eyes open wider when it gets there.

[18] Although PARADISE only had two levels of abstraction.

[19] All moves have to be part of a subgoal to be suggested.

4.2.4 When to Apply an "Early Grounding" Scheme

In [12], Smith, Nau and Throop also use a total order system in their Bridge program Tignum 2, noting that without this the problem of working out where certain cards might be would be nearly impossible[20]. In fact it would seem that the functions employed to do this could effectively be seen as an implicit world model. The bridge domain also has the property that it is very hard to abstractly model the interactions between goals since the uncertainty in the domain even precludes knowledge of which possible options are available for decomposition.

In a system mixing plan generation and execution (such as that in [1]) the decomposition scheme is also analogous to "early grounding", using the real (or simulated) world as a world model.

The scheme proved essential to making progress in the complex Go test domain where only a small proportion of the important effects of abstract goals could be represented directly in plan knowledge.

Our results suggest that an "early grounding" approach should be considered in the following cases:

- In domains where there is little knowledge available about the true interactions between low level goals. This would appear to include almost all game domains.

- In systems which are to include a data driven aspect. This may be an important consideration since in some domains it may be essential to integrate top-down and bottom-up approaches (it certainly seems that in Go human players often mix both types of knowledge).

The two key problems identified with the standard HTN decomposition scheme (Section 4.1.1) are both manifestations of the general difficulties with HTN Adversarial Planning outlined in the Introduction (Section 1). The "early grounding" scheme goes some way towards addressing the second problem (problem 2) in removing the dependence on knowledge of all inter-goal relationships. It also appears to provide a good mechanism for controlling the work load in planning which suggests it is also useful in the context of problem 1.

5 The Method in Practice

To test the effectiveness of the approach outlined above the planning architecture was instantiated in the form of a Go reasoning system.

5.1 Using Abstract Knowledge for Go

The Go reasoning system (GoBi) took the form of a knowledge base for the planning architecture described above. Knowledge was limited to plans involving the attacking and defending of groups and strings on the board. Plan fragments (schemas) held in the system were at five levels of abstraction,

[20] They however enforce a total-order by excluding partially ordered schemas which is considerably more restrictive than choosing a linearisation and allowing the system to backtrack upon this.

the lowest level being moves on the board and the highest being goals such as *kill this group* or *save this string* (there are many diverse ways of doing either).

Two critics which alerted the planning agents (Alpha and Beta) to groups in Atari[21] on the board were also added to allow the agents to consider defending or attacking these alongside the main plan. GoBI was tested both with and without these additional data driven aspects.

5.2 Success at Playing Go

Though limited in the scope of its knowledge GoBI showed a lot of promise. The system was tested on a set of 85 problems taken from Volume I of the *Graded Go problems for beginners* series ([15])[22].

The system successfully answered 74% of these examples, which is a significant achievement given its limited knowledge and the complexity of some of the problems.

The two data driven plan critics considerably improved the robustness of the system and increased the scope of problems GoBI could deal with. With the critics active the plans produced were more complete, taking into account more of the possible attacks.

The success of GoBI certainly demonstrates the usefulness of abstract knowledge for Go and highlights some of the features of the game which make the approach suitable:

- The search spaces involved are so large that standard search techniques are difficult to apply, making knowledge based and goal driven approaches more attractive.

- There is a wealth of tactical and strategic knowledge available for Go which has clear parallels with abstract plans. Ideas in teaching books are often expressed as abstract aims and sets of goals.

- Knowledge about the game state can be represented at various levels; from single stones on the board to macro structures such as strings, groups and armies. This gives an excellent basis for expressing abstract plans.

6 Conclusions

A more detailed breakdown of implementation, testing and results for the architecture and the Go system can be found in [14]. The following main results summarize the work presented here:

- The paper addresses the key issue of goal decomposition schemes for HTN Adversarial Planning and suggests that the knowledge of relationships and interactions between goals is the key influence in deciding upon a decomposition scheme.

- Although schemes similar to "early grounding" have been used elsewhere, we formalise these and identify their key role in making HTN Adversarial Planning useful in domains such as Bridge, Chess and Go.

- We also highlight the possibility of adding data driven aspects into such a system which may be essential in some domains (Go almost certainly being one).

- The planning architecture performed well for a domain as complex as Go. This demonstrates that the HTN Adversarial Planning approach can be useful for such hard problems. Hopefully this work will re-awaken interest in using this type of approach to solve problems in adversarial domains.

Lastly the "early grounding" HTN scheme may also be useful in some non-adversarial domains. In domains where there is little a priori knowledge about the interactions of goals but which still have other factors favoring a goal driven approach[23] this method may provide a useful compromise.

ACKNOWLEDGEMENTS

The first author is very grateful for financial support from EPSRC under grant number 96417312. The second and fourth authors are supported by EPSRC grant GR/L/11724. The third author is partly funded by DARPA/AFRL contract F30602-95-1-0022. Thanks also go to Prof. Boi Faltings and Gilles Trombettoni for helpful comments on earlier drafts and to Neil Firth for numerous instructive games of Go.

REFERENCES

[1] C. Applegate, C. Elsaesser, and D. Sanborn, 'An architecture for adversarial planning', *IEEE Transactions on Systems, Man and Cybernetics*, **20**(1), 186–294, (1990).

[2] F. Bacchus and Q. Yang, 'The expected value of hierachical problem-solving', *Proceedings AAAI-92*, 369–374, (1992).

[3] J. Burmeister and J. Wiles, 'An introduction to the computer Go field and associated internet resources', Technical report, The University of Queensland, (January 1997). Available online at: http://www/psy.uq.edu.au/~jay/go/go_page.html.

[4] J. G. Carbonell, 'Counterplanning: A strategy based model of adversarial planning in real world situations', *Artificial Intelligence*, **16**(1), 295–329, (1981).

[5] Cgml, 'Computer Go mailing list', Technical report, (1997). Archive at: ftp://bsdserver.ucsf.edu/.

[6] K. Erol, D. Nau, and J. Hendler, 'Umcp: A sound and complete planning procedure for hierarchical task-network planning', *AIPS-94*, (June 1994).

[7] R. E. Fikes and N. J. Nilsson, 'Strips: A new approach to the application of theorem proving to problem solving', *Artificial Intelligence*, (2), 189–208, (1971).

[8] C.. Knoblock, 'Search reduction in hierachical problem solving', *Proceedings AAAI-91*, 686–691, (1991).

[21] i.e. which can be captured in one move.

[22] These books are Go teaching books and well known to many Go players. (Volume I contains approximately 240 problems; this set represented approximately a third of these, including many of the most complex.) There are four volumes in the series and problems go from beginner level to expert.

[23] Such as good abstract plan knowledge for decompositions, large numbers of available actions and difficulties in estimating the distance from achieving the goal.

[9] P. Lehner, 'Strategic planning in Go', in *Computer Game Playing: Theory and Practice*, ed., M. A. Bramer, 167–176, Ellis Horwood, (1983).

[10] J. Pitrat, 'A Chess program which uses plans', *Artificial Intelligence*, **8**(1), 275–321, (1977).

[11] E. D. Sacerdoti, *A structure for plans and behaviour*, Elsevier, 1977.

[12] S. J. J. Smith, D. S. Nau, and T. A. Throop, 'Total-order multi-agent task-network planning for contract Bridge', *Proceedings AAAI-96*, 108–113, (1996).

[13] D. E. Wilkins, 'Using patterns and plans in Chess', *Artificial Intelligence*, **14**(1), 165–203, (1980).

[14] S. N. Willmott, *Adversarial Planning and the game of Go*, Master's thesis, Department of Artificial Intelligence, University of Edinburgh, September 1997.

[15] K. Yoshinori, *Graded Go Problems for beginners (Volume I)*, The Ishi Press Inc, 1985.

[16] P. R. Young and P. Lehner, 'Applications of a theory of automated adversarial planning to command and control', *IEEE Transactions on Systems, Man and Cybernetics*, **16**(6), 186–294, (1990).