Heuristic Planning in Adversarial Dynamic Domains *

Simon Chamberland and **Froduald Kabanza**
Département d'informatique
Université de Sherbrooke
Sherbrooke (Québec), Canada
{simon.chamberland, kabanza}@usherbrooke.ca
http://aaai2011:aaai2011@planiart.usherbrooke.ca/~simon/projects/agent/

Introduction

Agents in highly dynamic adversarial domains, such as RTS games, must continually make time-critical decisions to adapt their behaviour to the changing environment. RTS games involve two players who build structures, recruit armies and fight for space and resources in order to control strategic points, destroy the opposing force and ultimately win the game. Other examples of adversarial domains include security and defense applications. In such a context, the planning agent must consider his opponent's actions as uncontrollable, or at best influenceable. Turn-taking domains, such as chess, enjoy heuristic search methods which somewhat explicitly take into account the potential actions of the opponent to guide the search algorithm. However, for more general nondeterministic domains where there is no clear turn-taking protocol, most heuristic search methods to date do not explicitly reason about the opponent's actions when guiding the state space exploration towards goal or high-reward states.

In contrast, we are investigating a domain-independent heuristic planning approach which reasons about the dynamics and uncontrollability of the opponent's behaviours in order to provide better guidance to the search process of the planner. Our planner takes as input the opponent's behaviours recognized by a plan recognition module and uses them to identify opponent's actions that lead to low-utility projected states. It relies on a planning graph (Ghallab, Nau, and Traverso 2004) to determine the agent's actions that are likely to void the preconditions of these undesirable opponent actions or counter their effects. We believe such explicit heuristic reasoning about the potential behaviours of the opponent is crucial when planning in adversarial domains, yet is missing in today's planning approaches.

Adversarial Planning Problem

Adversarial planning problems usually involve finding a plan that allows an agent to achieve its goal regardless of the behaviour adopted by the environment. Such problems are traditionally studied in various fields under different formulations, mainly reactive program synthesis (Abadi, Lamport, and Wolper 1989), discrete-event controller synthesis (Asarin et al. 1998), and nondeterministic planning (Ghallab, Nau, and Traverso 2004). Game theory (Fudenberg and Tirole 1991) provides a utility-based framework where explicit reasoning on the possible environment behaviours is conducted. As all these approaches need in one way or another to explore a state space, they are all limited by the state-space explosion problem. The area of AI planning has contributed several techniques to cope with this problem, including heuristic techniques as well as compact state space representations (Boutilier, Dearden, and Goldszmidt 2000). However, even with these advancements, planning in adversarial domains remains very challenging.

Unlike these approaches, our method efficiently exploits prior knowledge about the environment's primitive actions and hypothesized behaviours in order to determine actions for the planning agent that best counters these behaviours. In practice, the hypothesized behaviours of the environment would be recognized by a plan recognition system such as (Sadilek and Kautz 2010). For testing purposes, we also allow them to be specified manually.

Planning Method

Framing the Problem

We consider one player (A) as the planning agent and his opponent (E) as the environment. Both A and E execute actions simultaneously, with no information on each other's simultaneous decisions. Each action a has a precondition c_a and an effect e_a. The planner is given a description of A's and E's feasible actions, which it uses to explore a *game tree* conveying their concurrent execution. We assume that the initial state is completely known and the states are fully observable, and that all actions have the same unit duration. These simplifying assumptions help provide a starting point for our approach but we intend to relax them in the long run.

Both A and E strive to steer the gameplay towards what they respectively perceive as favourable states. We use a utility function to approximate the desirability of a given state from A's perspective – the desirability from E's point of view is the exact opposite. In the following, we refer to *strategies* as mappings from states to actions. A *pure* strategy associates states to a unique action, while a *mixed* strategy associates states to probability distributions over possi-

*The work presented herein was supported by the NSERC.

ble actions. Formally, we formulate the problem as a two-player zero-sum simultaneous game where the objective is to find a strategy maximizing A's expected utility within the alloted time frame. Our approach optionally takes as input E's behaviours recognized by a plan recognition module, which specify a mixed strategy defined for a certain set of states S_E. The strategy might be based on a specific *a priori* model of the opponent – for example, attack behaviours could be considered more probable when playing against an aggressive opponent.

Main Algorithm

The planning approach algorithm is essentially an iterative deepening procedure exploring a limited subset of the game tree and progressing further ahead until time runs out. Within each iteration, the algorithm computes a strategy based on the possible environment's behaviours restricted by the current time horizon. States $s \in S_E$ must only be associated an appropriate action for A; these are states for which we know E's strategy. However, states $s \notin S_E$ do not contain any information about E's actions; in these states a suitable strategy for A must be found, given an assessment of E's strategy which must also be computed. E's strategy can be estimated using various opponent models. In our approach we combine the two models detailed below.

Worst-Case Opponent Model The worst-case opponent model consists in assuming E chooses the *best response* to A's action choice – that is, the action inducing the greatest possible loss for A, given A's action. Expanding states with the worst-case opponent model can be achieved via a simultaneous-move variant of the well-known alpha-beta pruning algorithm. Since this model is not realistic, we use it as a starting point for further refinement.

Nash Equilibrium Model We use the *Nash equilibrium* model to adequately represent the agents' lack of knowledge about each other's strategy. A set of strategies (in our case A's and E's strategies) is in a Nash equilibrium when neither agent has an incentive to unilaterally deviate from its current strategy, given the other agents' strategies. In a two-player zero-sum setting, the equilibrium can be found by solving a Linear Programming problem, whose complexity is polynomial in the number of actions. This yields a probability distribution over the possible actions. However, accurate computation of the equilibrium is too time-consuming for game trees of reasonable size. We are considering sampling-based approaches (Lanctot et al. 2009) converging to an ϵ-Nash equilibrium, i.e. an approximation of the equilibrium with a difference of ϵ in terms of expected utility.

Heuristics

The heuristics presented in this section are independent from the opponent model used in the game tree expansion process; their purpose is to suggest beneficial actions for A that will hopefully mitigate the future undesirable outcomes of E's estimated strategy.

Let s_0, s_1, and s_2 denote successive states expanded by the algorithm. When expanding the game tree, the algorithm compares the utility of successive states – say s_1 and s_2 – and marks a specific E's action as *undesirable* whenever it causes a utility loss greater than a certain threshold. Let a_1 be such an undesirable action. When propagating the utility value of s_2 back to s_1, the algorithm also includes the undesirable actions that s_1 may lead to – in this case, a_1 – and the time before which this action must be prevented. Regression is performed on the undesirable actions' preconditions to include prior actions on which they depend. For example, E may have chosen action a_0 in state s_0 specifically to enable action a_1 in the next state s_1. This process yields, for each state, a set of actions Act which the planning agent A aims to prevent and the time at which they take place.

To generate alternative actions for A, we maintain a planning graph rooted at the initial state from which we extract sequences of actions specifically chosen to falsify some undesirable action's precondition. The main advantage provided by the planning graph is the guarantee that, given an action $a \in Act$ taking place at time t, the action a cannot be prevented in time if level t of the planning graph is not compatible with $\neg c_a$.

Conclusion

Adversarial planning problems form an important research topic in many areas, yet remain very challenging to address with recent planning methods. The approach we propose tackles a new dimension of the problem which to the best of our knowledge has not been examined so far. Moreover, it also corresponds to how humans intuitively approach adversarial planning problems in practice: by pondering over their possible options, given situations that they would like to prevent. Future work will aim to implement our approach in an RTS-playing agent and relax the simplifying assumptions of perfect information and actions having similar duration.

References

Abadi, M.; Lamport, L.; and Wolper, P. 1989. Realizable and unrealizable specifications of reactive systems. In *Proc. of the 16th Int. Colloquium on Automata, Languages and Programming, LNCS Vol. 372*, 1–17.

Asarin, E.; Maler, O.; Pnueli, A.; and J.Sifakis. 1998. Controller synthesis for timed automata. In *IFAC conference on System Structure and Control*, 467–474.

Boutilier, C.; Dearden, R.; and Goldszmidt, M. 2000. Stochastic dynamic programming with factored representations. *Artificial Intelligence* 121:49–107.

Fudenberg, D., and Tirole, J. 1991. *Game Theory*. MIT Press.

Ghallab, M.; Nau, D.; and Traverso, P. 2004. *Automated Planning: Theory and Practice*. Morgan Kaufmann.

Lanctot, M.; Waugh, K.; Zinkevich, M.; and Bowling, M. 2009. Monte Carlo sampling for regret minimization in extensive games. In *Advances in Neural Information Processing Systems 22 (NIPS)*, 1078–1086.

Sadilek, A., and Kautz, H. 2010. Recognizing multi-agent activities from GPS data. In *Proceedings of the 24th Conference on Artificial Intelligence (AAAI)*.