OBDD-Based Optimistic and Strong Cyclic Adversarial Planning

Rune M. Jensen, Manuela M. Veloso and Michael H. Bowling
Computer Science Department,
Carnegie Mellon University,
5000 Forbes Ave, Pittsburgh,
PA 15213-3891, USA
Email: {runej,mmv,mhb}@cs.cmu.edu
Tel.: +1 (412) 268-{3053,1474,3069}
Fax: +1 (412) 268-4801

Abstract

Recently, universal planning has become feasible through the use of efficient symbolic methods for plan generation and representation based on reduced ordered binary decision diagrams (OBDDs). In this paper, we address adversarial universal planning for multi-agent domains in which a set of uncontrollable agents may be adversarial to us. We present two new OBDD-based universal planning algorithms for such adversarial non-deterministic finite domains, namely *optimistic adversarial planning* and *strong cyclic adversarial planning*. We prove and show empirically that these algorithms extend the existing family of OBDD-based universal planning algorithms to the challenging domains with adversarial environments. We further relate adverserial planning to positive stochastic games by analyzing the properties of adversarial plans when these are considered policies for positive stochastic games. Our algorithms have been implemented within the Multi-agent OBDD-based Planner, UMOP, using the Non-deterministic Agent Domain Language, *NADL*.

Keywords: adversarial universal planning, multi-agent planning, non-deterministic domains, stochastic games.

1 Introduction

Universal planning, as originally developed by Schoppers (1987), is an approach for handling environments with contingencies. Universal planning is particularly appealing for active environments causing actions to be non-deterministic. A universal plan associates each possible world state with actions relevant in that state for achieving the goal. Due to the non-deterministic outcome of actions, a universal plan is executed by iteratively observing the current state and executing an action in the plan associated with that state.

In the general case the non-determinism forces a universal plan to cover all the domain states. Since planning domains traditionally have large state spaces, this constraint makes the representation and generation of universal plans nontrivial. Recently, reduced ordered binary decision diagrams (OBDDs,[1]) have been shown to be efficient for synthesizing and representing universal plans [2, 3, 6]. OBDDs are compact representations of Boolean functions that have been successfully applied in *symbolic model checking* [7] to implicitly represent and traverse very large state spaces. Using similar techniques, a universal plan represented as an OBDD can be efficiently synthesized by a backward breadth-first search from the goal to the initial states in the planning domain.[1]

There are three OBDD-based planning algorithms: *strong*, *strong cyclic* and *optimistic planning*. Strong and strong cyclic planning were contributed within the MBP planner [2, 3]. MBP specifies a planning domain in the action description language $\mathcal{AR}$ [4]. The strong planning algorithm tries to generate a strong solution, i.e., a plan where all possible outcomes of the actions in the plan change the state to a new state closer to the goal. The strong cyclic planning algorithm returns a strong solution, if one exists, or otherwise tries to generate a plan that may contain loops but is guaranteed to achieve the goal, given that all cyclic executions eventually terminate.

Optimistic planning was contributed within the UMOP planner [6]. UMOP specifies a planning domain in the non-deterministic agent domain language, *NADL* , that explicitly defines a controllable system and an uncontrollable environment as two sets of synchronous agents. Optimistic planning tries to generate a relaxed plan where the only requirement is that there exists an outcome of each action that leads to a state nearer the goal.

None of the previous algorithms are generally better than the others, as their strengths and limitations depend on the structure of the domain [6]. A limitation of the previous algorithms, however, is that they do not reason explicitly about the joint actions of system and environment agents. Instead the environment actions are only *implicitly* represented by the non-determinism they cause on system actions.[2] This is an important restriction for adversarial domains, as for the strong cyclic and optimistic algorithms, an adversarial environment may be able to prevent goal achievement.

In this paper we contribute two new planning algorithms robust for adversarial environments: *optimistic adversarial planning* and *strong cyclic adversarial planning*. These algorithms explicitly represent environment actions. The planner can then rea-

[1]This work assumes that the non-deterministic domain definition is known and the focus is on the development of effective universal planning algorithms under this assumption.

[2]Figure 1(b) illustrates the representation for an example domain introduced in next section.

son about these actions and take adversarial behavior into account. We prove that, in contrast to strong cyclic plans, strong cyclic *adversarial* plans guarantee goal achievement independently of the environment behavior. Similarly, we prove that optimistic adversarial solutions improve the quality of optimistic solutions by guaranteeing that a goal state can be reached from any state covered by the optimistic adversarial plan independently of the behavior of the environment. The results are verified empirically in a scalable example domain using an implementation of our algorithms in the UMOP planning framework.

Adversarial planning is related to game theory. The main difference is that the goal is represented in terms of a set of states instead of a utility function. Unlike strong cyclic adversarial planning, game tree algorithms, such as alpha-beta mini-max [5], can only guarantee goal achievement if the search is complete and the opponent uses a strict mini-max strategy. In practice, though, the explicit-state search has to be depth-bounded in which case the approach is reduced to heuristic action selection. Matrix games are stateless and therefore strictly less expressive. The game-theoretic framework that is closest in relation to adversarial planning is stochastic games (SGs). Stochastic games extend Markov decision processes (MDPs) to multiple agents. An MDP has transition probabilities and is thus more expressive than the non-deterministic transition model of adversarial planning. We show, though, that an adversarial planning problem can be translated into an SG problem by adding non-zero transition probabilities. We further prove that an optimistic adversarial plan exists if and only if the corresponding positive stochastic game solution has a positive expected reward. Moreover, if a strong cyclic adversarial plan exists, then the stochastic game solution has maximum expected reward.

The restricted domain model of adversarial planning is suitable for problems where transition probabilities are irrelevant (e.g. worst case analysis). The advantage of this domain model compared to the MDP model of SGs is that it allows the application of symbolic solution methods that potentially scale to much larger domains than can be handled by the explicit-state value iteration methods (e.g. [9]) used for solving stochastic games.

The remainder of the paper is organized as follows. Section 2 defines our representation of adversarial domains and introduces an example domain used throughout the paper. Section 3 defines the optimistic and strong cyclic adversarial planning algorithms and proves their key properties. In Section 4 we define and prove properties of the stochastic game representation of the adversarial planning problems. Finally, Section 5 draws conclusions and discusses directions for future work.

2 Adversarial Plan Domain Representation

An *NADL* domain description consists of: a definition of *state variables*, a description of *system* and *environment agents*, and a specification of *initial* and *goal conditions*. The set of state variable assignments defines the state space of the domain. An agent's description is a set of *actions*. At each step, all of the agents perform exactly one action, and the resulting action tuple is a *joint action*.[3] The system agents model the behavior

[3] In the remainder of the paper we will often refer to joint-actions as simply actions.

of the agents controllable by the planner, while the environment agents model the uncontrollable environment. There are two causes of non-determinism in *NADL* domains: (1) actions having multiple possible outcomes, and (2) uncontrollable concurrent environment actions. System and environment actions are assumed to be independent, such that an action chosen by the system in some state can not influence the set of actions that can be chosen by the environment in that state and vice versa. No assumptions are made about the behavior of environment agents. They might be *adversarial*, trying to prevent goal achievement of the system agents.

We represent the transition relation of an *NADL* domain with a Boolean function, $T(S, a_s, a_e, S')$. S is the current state, a_s and a_e are system and environment actions and S' is a next state. $T(S, a_s, a_e, S')$ is true if and only if S' is a *possible* next state when executing (a_s, a_e) in S. We assume that a next state is possible only if it is associated with a non-zero transition probability in the world we model.

A *planning problem* is a tuple (T, I, G), where T is a transition relation, and I and G are Boolean functions representing the initial and goal states, respectively. A *universal plan*, U, is a partial mapping from the domain states to the power set of system actions. A universal plan would be executed by the system agents by iteratively observing the current state and executing one of the actions in the universal plan associated to that state.

As an example, consider the domain shown in Figure 1. This domain has a single system agent that can execute the actions $+s$ and $-s$, and a single environment agent that can execute the actions $+e$ and $-e$. Edges in the figure are labeled with the corresponding joint action. There are 5 states, namely I, F, D, U and G. I and G are the only initial and goal states, respectively. D is a dead end state, as the goal is unreachable from D. This introduces an important difference between F and U, that captures a main aspect of the adversarial planning problem. We can view the two states F and U as states in which the system and the environment have different opportunities. Observe that the system "wins", i.e., reaches the goal, only if the signs of the two actions in the joint action are different. Otherwise it "loses", as there is no transition to the goal with a joint action with actions with the same sign. The goal is reachable from both states F and U. However the result of a "losing" joint action is different for F and U. In F, the system agent remains in F. Thus, the goal is still reachable for a possible future action. In U, however, the agent may transition to the dead end state D, which makes it impossible to reach the goal in subsequent steps.

Now consider how an adversarial environment can take advantage of the possibility of the system reaching a dead end from U. Since the system may end in D, when executing $-s$ in U, it is reasonable for the environment to assume that the system will always execute $+s$ in U. But now the environment can prevent the system from ever reaching the goal by always choosing action $+e$, so the system should completely avoid the path through U.

This example domain is important as it illustrates how an adversarial environment can act purposely to obstruct the goal achievement of the system. We will use it in the following sections to explain our algorithms. A universal plan, guaranteeing that G is eventually reached, is $\{(I, \{+s\}), (F, \{+s, -s\})\}$. In contrast to any previous universal planning algorithm, the strong cyclic adversarial planning algorithm can generate this plan as shown in Section 3.3.

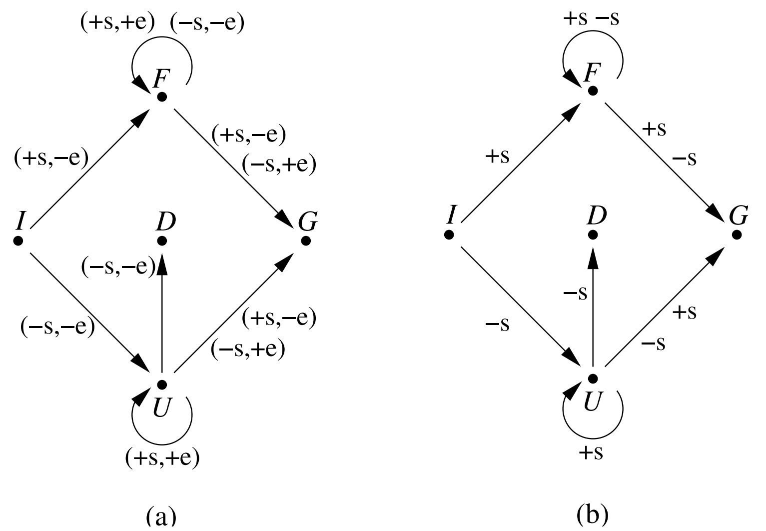

Figure 1: An adversarial planning domain example with initial state I and goal state G. There is a single system and environment agent with actions $\{+s, -s\}$ and $\{+e, -e\}$, respectively. (a) shows the explicit representation of environment actions used by our adversarial planning algorithms, while (b) shows the implicit representation used by previous algorithms, where the effect of environment actions is modeled by the non-determinism of system actions.

3 Adversarial Planning

We introduce a generic function *Plan*(T, I, G) for representing OBDD-based universal planning algorithms. The algorithms, including ours, only differ by definition of the function computing the precomponent (*PreComp*(T, V)).

The generic function performs a backward breadth-first search from the goal states to the initial states. In each step the precomponent, U_p, of the visited states, V, is computed. The precomponent is a partial mapping from states to the power set of system actions. Each state in the precomponent is mapped to a set of relevant system actions for reaching V.

function *Plan*(T, I, G)
  $U := \emptyset$; $V := G$
  while $I \not\subseteq V$
    $U_p := PreComp(T, V)$
    if $U_p = \emptyset$ **then return** *failure*
    else $U := U \cup U_p$
      $V := V \cup states(U_p)$
  return U

If the precomponent is empty a fixpoint of V has been reached that does not cover the initial states. Since this means that no universal plan can be generated that covers the initial states, the algorithm returns *failure*. Otherwise, the precomponent is added to the universal plan and the states in the precomponent are added to the set of visited states. All sets and mappings in the algorithm are represented by OBDDs. In particular, the universal plan and the precomponent are represented by the characteristic function of the set of state-actions pairs in the mapping.

3.1 The Optimistic Adversarial Precomponent

The core idea in adversarial planning is to be able to generate a plan for the system agents that ensures that the environment agents, with complete domain knowledge, are unable to choose actions that prevent the system from reaching the goal. We formalize this idea in the definition of a *fair state*. A state s is fair with respect to a set of states, V, and a universal plan, U, if, for each applicable environment action, there exists a system action in U such that the joint action has a possible next state in V. Formally we have:

Definition 1 (Fair State) *A state, s, is fair with respect to a set of states, V, and a universal plan, U, if and only if $\forall a_e \,.\, \exists a_s \in U(s) \,.\, T(s, a_s, a_e, s') \wedge s' \in V$.*

For convenience we define an *unfair* state to be a state that is not fair. The optimistic adversarial precomponent is an optimistic precomponent pruned for unfair states. In order to use a precomponent for OBDD-based universal planning, we need to define it as a boolean function represented by an OBDD. The optimistic adversarial precomponent (OAP) is the characteristic function of the set of state-action pairs in the precomponent:

Definition 2 (OAP) *Given a transition relation, T, the optimistic adversarial precomponent of a set of states, V, is the set of state-action pairs given by the characteristic function:*

$$\begin{aligned}
OAP(T,V)(s,a_s) &= \big(\forall a_e \,.\, A(s,a_e) \Rightarrow J(s,a_e)\big) && (1)\\
&\quad \wedge \exists a_e, s' \,.\, T(s,a_s,a_e,s') && (2)\\
&\quad \wedge s' \in V \wedge s \notin V && (3)\\
J(s,a_e) &= \exists a_s, s' \,.\, T(s,a_s,a_e,s') && (4)\\
&\quad \wedge s' \in V && (5)\\
A(s,a_e) &= \exists a_s, s' \,.\, T(s,a_s,a_e,s'). && (6)
\end{aligned}$$

Line (1) ensures that the state is fair. It says that, for any state in the precomponent, every applicable environment action (defined by $A(s, a_e)$) must also be included in a joint action leading to V (defined by $J(s, a_e)$). Line (2) and (3) say that every system action a_s relevant for a state $s \notin V$ must have at least one next state in V.

Figure 2 shows the optimistic adversarial precomponent of state G for the example domain ($OAP(G) = \{(F,+s),(F,-s),(U,+s),(U,-s)\}$). For clarity we include the transitions of the actions in the precomponent.

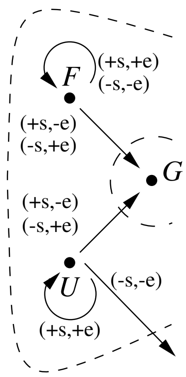

Figure 2: The OAP of G for the example of Figure 1.

3.2 The Strong Cyclic Adversarial Precomponent

A strong cyclic adversarial plan is a strong cyclic plan where every state is fair. Thus, all the actions in the plan lead to states covered by the plan. In particular, it is guaranteed that no dead ends are reached. The strong cyclic adversarial precomponent (SCAP) consists of a union of optimistic precomponents where each state in one of the optimistic precomponents is fair with respect to all states in the previous precomponents and the set of visited states.

The algorithm for generating an SCAP adds one optimistic precomponent at a time. After each addition, it first prunes actions with possible next states not covered by the optimistic precomponents and the set of visited states. It then subsequently prunes all the states that are no longer fair after the pruning of outgoing actions. If all the states are pruned from the precomponent, the algorithm continues adding optimistic precomponents until no actions are outgoing. Thus, in the final SCAP, we may have cycles due to loops and transitions crossing the boundaries of the optimistic precomponents. Again, we define the precomponent as the characteristic function of a set of state-action pairs:

Definition 3 (SCAP) *Given a transition relation, T, the strong cyclic adversarial precomponent of a set of states, V, is the set of state-action pairs given by the characteristic function $SCAP(T,V)(s,a_s)$.*

function $SCAP(T,V)$
  $i := 0$; $OA_0 := OAP(T,V)$
  while $OA_i \neq \emptyset$
    $SCA := PruneSCA(T,V,\mathbf{OA},i)$
    if $SCA \neq \emptyset$ **then**
      return SCA
    else $i := i+1$
      $OA_i := OAP\big(T, V \cup states(\mathbf{OA})\big)$
  return $\emptyset$

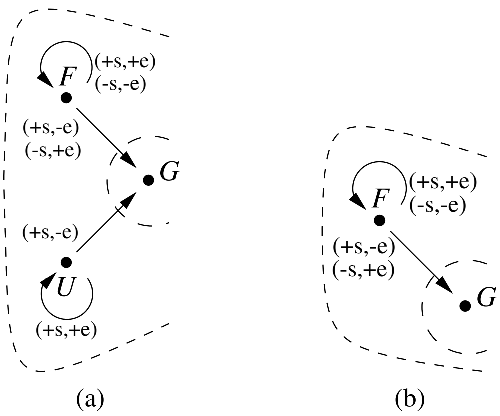

Figure 3: (a) The OAP pruned for actions with outgoing transitions; (b) The SCAP of G, for the example of Figure 1.

function $PruneSCA(T, V, \mathbf{OA}, i)$
  repeat
    $SCA := \cup_{j=0}^{i} OA_j$
    $V_C := V$; $V_T := V \cup states(\mathbf{OA})$
    for $j = 0$ **to** i
      $P := PruneOutgoing(T, V_T, OA_j)$
      $OA_j := PruneUnfair(T, V_C, P)$
      $V_C := V_C \cup states(OA_j)$
  until $SCA = \cup_{j=0}^{i} OA_j$
  return SCA

$$
\begin{aligned}
PruneOutgoing(T, V, OA) &= OA(s, a_s) \land \forall a_e, s' \,.\, \neg T(s, a_s, a_e, s') \lor V(s') \\
PruneUnfair(T, V, OA) &= SA(s, a_s) \land \forall a_e \,.\, A(s, a_e) \Rightarrow J(s, a_e)
\end{aligned}
$$

$SCAP(T, V)$ adds an optimistic adversarial precomponent until the pruning function $PruneSCA(T, V, \mathbf{OA}, i)$ returns a non-empty precomponent. The pruning function keeps a local copy of the optimistic adversarial precomponents in an array $\mathbf{OA}$. SCA is the precomponent found so far. The pruning continues until SCA reaches a fix point. V_T is the set of states in the current precomponent. In each iteration transitions leading out of V_T are pruned. States turning unfair with respect to V_C, because of this pruning, are then removed. V_C is the union of all the states in the previous optimistic precomponents and the set of visited states V.

For an illustration, consider again the OAP of G (see Figure 2). Action $-s$ would have to be pruned from U since it has an outgoing transition. The pruned OAP is shown in Figure 3(a). Now there is no action leading to G in U when the environment chooses $+e$. U has become unfair and must be pruned from the precomponent. Since the precomponent is non-empty no more optimistic precomponents have to be added. The resulting precomponent, $SCAP(G) = \{(F, +s), (F, -s)\}$, is shown in Figure 3(b).

3.3 Advantages of Adversarial Planning

Optimistic and strong cyclic adversarial planning extend the previous OBDD-based universal planning algorithms by pruning unfair states from the plan. For example, for the domain of Figure 1, the strong cyclic planning algorithm would generate the solution $\{(I, \{+s, -s\}), (F, \{+s, -s\}), (U, \{+s\})\}$, while the strong cyclic adversarial planning algorithm, as introduced above, generates the plan $\{(I, \{+s\}), (F, \{+s, -s\})\}$. It is capable of pruning U from the plan, since it becomes unfair. Also note that the solution is not a plain strong plan since progress towards the goal is not guaranteed in every state. It is easy to verify that there actually is no strong solution for this domain.

In order to state formal properties of adversarial planning, we define the *level* of a state to be the number of optimistic adversarial precomponents from the goal states to the state. We can now prove the following:

Theorem 1 (SCA Termination) *A strong cyclic adversarial plan eventually reaches a goal state if it is initiated in some state in the plan and actions in the plan are chosen randomly.*

Proof: Since all unfair states and actions with transitions leading out of the strong cyclic adversarial plan have been removed, all the visited states will be fair and covered by the plan. From the assumption about non-zero transition probabilities to possible next states and the random choice of actions in the plan, it then follows that each execution of a plan action has a non-zero probability of leading to a state at a lower level. Let m be the maximum level of some state in the strong cyclic adversarial plan (m exists since the state space is finite). Let p denote the smallest probability of progressing at least one level for any state in the plan. Then, from every state in the plan, the probability of reaching a goal state in m steps is at least p^m. Thus, the probability of reaching a goal state in mn steps is at least $1 - (1 - p^m)^n$, which approaches 1 for $n \to \infty$. Thus, a goal state will eventually be reached. $\Box$

The termination theorem makes strong cyclic adversarial solutions more powerful than strong cyclic solutions since termination of strong cyclic solutions only can be guaranteed by assuming no infinite looping (i.e. a "friendly" environment). For optimistic game solutions, termination can not be proved since dead ends may be reached. However, since optimistic solutions only consist of fair states, there is a chance of progressing towards the goal in each state:

Theorem 2 (OA Progress) *There is a non-zero probability of reaching the goal from each state in an optimistic adversarial plan if the actions in the plan are chosen randomly.*

Proof: This follows directly from the assumption about non-zero transition probabilities and the fact that each state in the plan is fair. $\Box$

Optimistic solutions do not have this property since unfair states may be included in the plans. Thus, it may be possible for the environment to prevent the system from progressing towards the goal either by forcing a transition to a dead end or by making transitions cyclic.

3.4 Empirical Results

The adversarial and previous OBDD-based universal planning algorithms have been implemented in the publicly available UMOP planning framework. In order to illustrate the scalability of our approach, we use a general version of the example domain of Figure 1, as shown in Figure 4.

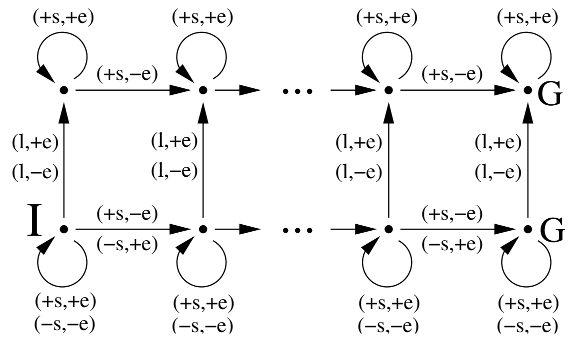

Figure 4: A general version of the domain of Figure 1.

The domain has a single system and environment agent with actions $\{+s, -s, l\}$ and $\{+e, -e\}$, respectively. The system progresses if the signs of the two actions in the joint action are different. At any time, it can switch from the lower to the upper row of states by executing l. In the upper row, the system can only execute $+s$. Thus, in these states an adversarial environment can prevent further progress by always executing $+e$.

Figure 5 shows, in a logarithmic scale, the running time and the plan size as a function of the number of domain states when finding strong cyclic and strong cyclic adversarial solutions.[4] In this domain both algorithms scale well. There is a low number of OBDD nodes growing linearly with the logarithm of the size of the state space. For the largest domain with 65,536 states the CPU time is less than 32 seconds for generating the strong cyclic adversarial plan. The results also demonstrate the efficiency of OBDDs for representing universal plans in structured domains. The strong cyclic adversarial plans only consider executing $-s$ and $+s$, while the strong cyclic plans consider all applicable actions. Hence, the strong cyclic adversarial plans have about twice as many OBDD nodes and take about twice as long time to generate. But this slightly higher cost of generating a strong cyclic adversarial plan pays off well in plan quality. The strong cyclic adversarial plan is guaranteed to achieve the goal when

[4]The experiment was carried out on a 500 MHz Pentium III PC with 512 MB RAM running Red Hat Linux 5.2.

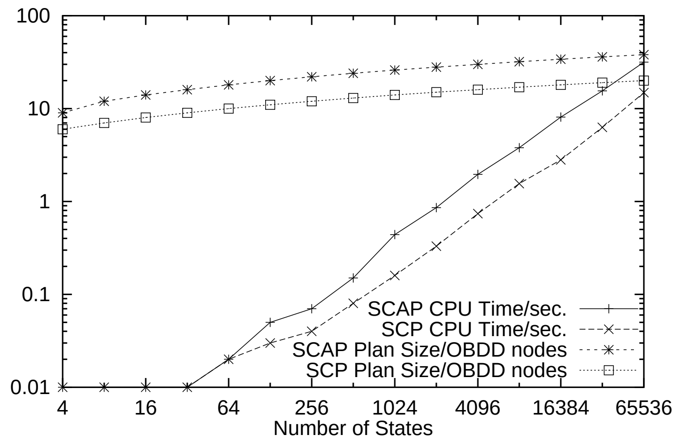

Figure 5: CPU time and Plan size of strong cyclic and strong cyclic adversarial solutions as a function of domain size.

choosing actions in the plan randomly. In contrast, the probability of achieving the goal in the worst case for the strong cyclic plan is less than $(\frac{2}{3})^{N/2-1}$, where N is the number of states in the domain. Thus, for an adversarial environment the probability of reaching the goal with a strong cyclic plan is practically zero, even for small instances of the domain.

4 Relation to Stochastic Games

A *stochastic game* is a tuple $(n, S, A_{1\ldots n}, T, R_{1\ldots n})$, where n is the number of agents, S is the set of states, A_i is the set of actions available to player i (and A is the joint action space $A_1 \times \ldots \times A_n$), T is the transition function $S \times A \times S \rightarrow [0,1]$, and R_i is a reward function for the ith agent $S \rightarrow \mathcal{R}$. A solution to a stochastic game is a stationary though possibly stochastic policy, $\rho : S \rightarrow PD(A_i)$, for an agent in this environment that maps states to a probability distribution over its actions. The goal is to find such a policy that maximizes the agent's future discounted reward. In a zero-sum stochastic game, two agents have rewards adding up to a constant value, in particular, zero. The value of a discounted stochastic game with discount factor γ is a vector of values, one for each state, satisfying the equation:

$$V_\gamma(s) \quad = \quad \max_{\sigma \in PD(A_s)} \min_{a_e \in A_e} \sum_{a_s \in A_s} \sigma(a_s) \left(\sum_{s' \in S} T(s, a_s, a_e, s') \left(R(s') + \gamma V_\gamma(s')\right) \right)$$

For positive stochastic games, the payoffs are summed without a discount factor, which can be computed as $V(s) = \lim_{\gamma \rightarrow 1^-} V_\gamma(s)$.

The derived stochastic game from a universal planning problem is given by the following definition:

Definition 4 *A* derived stochastic game *from a planning problem, (T, I, G), is a zero-sum stochastic game with states and actions identical to those of the planning problem. Reward for the system agent is one when entering a state in G, zero otherwise. Transition probabilities, $\bar{T}$, are any assignment satisfying,*

$$\begin{array}{ll} \bar{T}(s, a_s, a_e, s') \in (0, 1] & \textit{if } T(s, a_s, a_e, s') \\ \bar{T}(s, a_s, a_e, s') = 0 & \textit{otherwise} \end{array}$$

We can now prove the following two theorems:

Theorem 3 *An optimistic adversarial plan exists if and only if, for* **any** *derived stochastic game, the value at all initial states is positive.*

Proof: ($\Rightarrow$) We prove by induction on the level of a state. For a state, s, at level one, we know the state is fair with respect to the goal states. So, for every action of the environment, there exists an action of the system that transitions to a goal state with some non-zero probability. If T_{min} is the smallest transition probability, then if the system simply randomizes among its actions, it will receive a reward of one with probability $\frac{T_{min}}{|A_s|}$. Therefore, $V(s) \geq \frac{T_{min}}{|A_s|} > 0$. Consider a state, s, at level n. Since s is fair, we can use the same argument as above that the next state will be at a level less than n with probability $\frac{T_{min}}{|A_s|}$. With the induction hypothesis, we get,

$$V(s) \geq \frac{T_{min}}{|A_s|} V(s') > 0$$

Therefore, all states in the set V have a positive value. Since the algorithm terminates with $I \subseteq V$, then all initial states have a positive value.
($\Leftarrow$) Assume for some derived stochastic game that the value at all initial states is positive. Consider running the optimistic adversarial planning algorithm and for, the purpose of contradiction, assume the algorithm terminates with $I \not\subseteq V$. Consider the state $s^* \notin V$ that maximizes $V_\gamma(s^*)$. We know that, since the algorithm terminated, s^* must not be fair with respect to V. So there exists an action of the environment, a_e, such that, for all a_s, the next state will not be in V. So we can bound the value equation by assuming the environment selects action a_e,

$$V_\gamma(s^*) \quad \leq \quad \max_\sigma \sum_{a_s \in A_s} \sigma(a_s) \left(\sum_{s' \notin V} T(s^*, a_s, a_e, s') \left(\gamma V_\gamma(s') \right) \right)$$

Notice that we do not have to sum over all possible next states since we know the transition probabilities to states in V are zero (by the selection of a_e). We also know that immediate rewards for states not in V are zero, since they are not goal states. By our selection of s^* we know that $V_\gamma(s')$ must be smaller than the value of s^*. By substituting this into the inequality we can pull it outside of the summations which now sum to one. So $V(s^*) = 0$, as we need to satisfy:

$$V_\gamma(s^*) \leq \gamma V_\gamma(s^*) \cdot 1$$

Since any initial state is not in V, $V(s^*)$ must have value equal to zero, which is a contradiction to our initial assumption. So, the algorithm must terminate with $I \subseteq V$, and therefore an optimistic adversarial plan exists. □

Theorem 4 *If a strong cyclic adversarial plan exists, then for* **any** *derived stochastic game, the value at all initial states is 1.*

Proof: Consider a policy π that selects actions with equal probability among the unpruned actions of the strong cyclic adversarial plan. For states not in the adversarial plan select an action arbitrarily. We will compute the value of executing this policy, V_γ^π.

Consider a state s in the strong cyclic adversarial plan such that $V_\gamma^\pi(s)$ is minimal. Let $L(s) = N$ be the level of this state. We know that this state is fair with respect to the states at level less than N, and therefore (as in Theorem 1) the probability of reaching a state s' with $L(s') \leq N-1$ when following policy π is at least $p = \frac{T_{min}}{|A_s|} > 0$. This same argument can be repeated for s', and so after two steps, with probability at least p^2, the state will be s'' where $L(s'') \leq L(s') - 1 \leq N-2$. Repeating this, after N steps with probability p^N, the state will be s^N where $L(s^N) \leq L(s) - N \leq 0$, so s^N must be a goal state and the system received a reward of one.

So we can consider the value at state s when following the policy π. We know after N steps if it has not reached a goal state it must be in some state still in the adversarial plan due to the enforcement of the strong cyclic property. In essence, all actions with outgoing transitions are pruned and therefore are never executed by π. The value of this state by definition must be larger than $V_\gamma^\pi(s)$. Therefore,

$$\begin{aligned} V_\gamma^\pi(s) &\geq p^N\gamma^N \cdot 1 + (1-p^N)\gamma^N V_\gamma^\pi(s) \\ &\geq \frac{\gamma^N p^N}{1-(1-p^N)\gamma^N} \\ \lim_{\gamma \to 1} V_\gamma^\pi(s) &\geq \frac{p^N}{1-(1-p^N)} \geq 1 \end{aligned}$$

So $V^\pi(s) = 1$ and since s is the state with minimal value, for any initial state s_i, $V^\pi(s_i) = 1$. Since $V(s_i)$ maximizes over all possible policies, $V(s_i) = 1$. □

5 Conclusion

This paper contributes two new OBDD-based universal planning algorithms, namely optimistic and strong cyclic adversarial planning. These algorithms naturally extend the previous optimistic and strong cyclic algorithms to adversarial environments. We have proved that, in contrast to optimistic solutions, optimistic adversarial solutions always have a non-zero probability of reaching a goal state. Furthermore, we have proved and shown empirically that, in contrast to strong cyclic solutions, a strong cyclic adversarial solution always eventually reach the goal. Finally, we introduced and proved several relations between adversarial universal planning and positive stochastic games. An interesting direction for future work is to investigate if adversarial planning can be used to scale up the explicit-state approaches for solving stochastic games by pruning states and transitions irrelevant for finding an optimal policy.

Acknowledgments

We wish to thank Scott Lenser for early discussions on this work. The research is sponsored in part by the Danish Research Agency and the United States Air Force under Grants Nos F30602-00-2-0549 and F30602-98-2-0135. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency (DARPA), the Air Force, the Danish Research Agency, or the US Government.

References

[1] R. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 8:677–691, 1986.

[2] A. Cimatti, M. Roveri, and P. Traverso. OBDD-based generation of universal plans in non-deterministic domains. In *Proceedings of AAAI-98*, pages 875–881. AAAI Press, 1998.

[3] A. Cimatti, M. Roveri, and P. Traverso. Strong planning in non-deterministic domains via model checking. In *Proceedings of the 4th International Conference on Artificial Intelligence Planning System (AIPS'98)*, pages 36–43. AAAI Press, 1998.

[4] E. Giunchiglia, G. Kartha, and Y. Lifschitz. Representing action: Indeterminacy and ramifications. *Artificial Intelligence*, 95:409–438, 1997.

[5] T. P. Hart and D. J. Edwards. The tree prune (TP) algorithm. Technical report, MIT, 1961. Artificial Intelligence Project Memo 30.

[6] R. Jensen and M. Veloso. OBDD-based universal planning for synchronized agents in non-deterministic domains. *Journal of Artificial Intelligence Research*, 13:189–226, 2000.

[7] K. L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publ., 1993.

[8] M. Schoppers. Universal planning for reactive robots in unpredictable environments. In *Proceedings of IJCAI-87*, pages 1039–1046, 1987.

[9] L. S. Shapley. Stochastic games. *PNAS*, 39:1095–1100, 1953.