On Adversarial Policy Switching with Experiments in Real-time Strategy Games

Brian King[1] and **Alan Fern**[2] and **Jesse Hostetler**[2]
Department of Electrical Engineering and Computer Science
Oregon State University
[1]kingbria@lifetime.oregonstate.edu
[2]{afern, hostetje}@eecs.orst.edu

Abstract

Given a Markov game, it is often possible to hand-code or learn a set of policies that capture a diversity of possible strategies. It is also often possible to hand-code or learn an abstract simulator of the game that can estimate the outcome of playing two strategies against one another from any state. We consider how to use such policy sets and simulators to make decisions in large Markov games such as real-time strategy (RTS) games. Prior work has considered the problem using an approach we call minimax policy switching. At each decision epoch, all policy pairs are simulated against each other from the current state, and the minimax policy is chosen and used to select actions until the next decision epoch. While intuitively appealing, our first contribution is to show that this switching policy can have arbitrarily poor worst case performance. Our second contribution is to describe a simple modification, whose worst case performance is provably no worse than the minimax fixed policy in the set. Our final contribution is to conduct experiments with these algorithms in the domain of RTS games using both an abstract game engine that we can exactly simulate and a real game engine that we can only approximately simulate. The results show the effectiveness of policy switching when the simulator is accurate, and highlight challenges in the face of inaccurate simulations.

1 Introduction

In many complex games, such as real-time strategy (RTS) games, the space of possible actions is much too large for an agent to reason about directly while still meeting real-time constraints. One way to deal with this is to create a set of policies, each of which fully specifies the primitive actions to take in each state, that captures the diversity of possible strategies. The reasoning problem is then reduced to choosing, in a computationally efficient way, which policy to follow at any point in time, effectively switching opportunistically among policies. In this work, we consider formal properties of one such *policy switching* approach, and report on preliminary experiments in an RTS game.

Prior work (Chang, Givan, and Chong 2004) investigated policy switching for (non-adversarial) MDPs, where a simulator is used at each state to select and execute the policy from a given set with highest estimated value. It was proven that the value of the switching policy is never worse than that of the best policy. In practice switching often leads to significant improvements compared to the best individual policy.

Policy switching has also been investigated in the adversarial setting of Markov games. Sailer, Buro, and Lanctot (2007) used policy switching to control military strategies in an abstract strategy game. Given a set of strategies, they simulate all pairs against one another at each time step and select one according to a minimax or Nash criterion. This approach outperformed the individual policies. Chang (2006) gave a formal analysis of the switching criterion, bounding the difference between the worst-case performance of the switching policy compared to the best fixed policy.

Our first contribution is to show that the above performance bound can be arbitrarily large and is in fact a tight bound. Thus, unlike the case of MDPs, naive policy switching in Markov games can perform arbitrarily poorly compared to a fixed policy. Next, we define a modified approach that is guaranteed to be no worse than a fixed policy. Finally, we summarize experimental results from applying policy switching to an RTS game agent, which provide evidence for the effectiveness of policy switching and also highlight the impact of having an inaccurate simulator.

2 Minimax Policy Switching

A two player, zero-sum Markov game is a tuple, (S, A_1, A_2, P, c), where S is a set of states, A_1 and A_2 are the actions sets for the minimizing and maximizing players respectively, $P : S \times A_1 \times A_2 \times S \mapsto [0, 1]$ is the transition probability distribution, and $c : S \times A_1 \times A_2 \mapsto \mathbb{R}$ is the instantaneous cost function that assigns a cost to each pair of action choices in each state. A policy for a player is a possibly stochastic mapping from states to actions, possibly depending on time in the case of non-stationary policies.

We will consider a finite-horizon Markov game setting with horizon H. To simplify our notation we will assume that the only non-zero costs are received at the horizon (i.e. the end of the game). This is without loss of generality, since any finite-horizon Markov game can be converted into an equivalent Markov game of this form. The h steps-to-go cost (with respect to the minimizer) of minimizer policy π and maximizer policy ϕ starting in state $s \in S$ is

$$C_h(\pi, \phi)(s) = E\left[\sum_{t=0}^{h} c(S_{h-t}, \pi(S_{h-t}), \phi(S_{h-t}))\Big| S_h = s\right]$$

where S_h is a random variable denoting the state when there are h steps to go after following π and ϕ starting in s.

The policy switching approaches that we consider assume the availability of policy sets for the minimizing and maximizing agent, denoted by Π and Φ respectively. Ideally the policy sets capture the typical policies that one might encounter in actual game play and may be the same for the minimizer and maximizer. In the context of robust optimization, the maximizer policies might correspond to possible conditions that we would like to be robust against. The *minimax cost* with respect to these sets for a given state s and horizon h is defined as

$$\text{MMC}_h(\Pi, \Phi)(s) = \min_{\pi \in \Pi} \max_{\phi \in \Phi} C_h(\pi, \phi)(s).$$

We also define the *minimax policy* for s and h, denoted by $\text{MMP}_h(\Pi, \Phi)(s)$, to be a minimizer policy that achieves the minimax cost. That is, a policy π^* such that $\max_{\phi \in \Phi} C_h(\pi^*, \phi)(s) = \text{MMC}_h(\Pi, \Phi)(s)$.

The *minimax switching policy* $\pi_{\text{ps}}(s, h)$ of Chang (2006) and Sailer, Buro, and Lanctot (2007) can now be defined as

$$\pi_{\text{ps}}(s, h) = \pi^*_h(s)$$
$$\pi^*_h = \text{MMP}_h(\Pi, \Phi)(s).$$

We see that $\pi_{\text{ps}}(s, h)$ simply selects a policy pair that achieves the minimax value with respect to Π and Φ and then returns the action selected by the minimizer policy. Note that this approach requires the ability to evaluate the cost $C_h(\pi, \phi)(s)$ for any policy pair. In practice, this is often approximated via Monte-Carlo sampling using an approximate or exact simulator of the Markov game.

A question now is what guarantees can be made about the performance of π_{ps}. Recall that in (non-adversarial) MDP policy switching, one can show that the switching policy will do no worse than the best policy in the set. The analogous result for Markov games under a minimax objective would show that for any horizon h and state s,

$$\max_{\phi \in \Phi} C_h(\pi_{\text{ps}}, \phi)(s) \leq \min_{\pi \in \Pi} \max_{\phi \in \Phi} C_h(\pi, \phi)$$

If we believe that the set of opponent policies Φ captures the typical range of opponents to be encountered, then this guarantee is quite powerful, since we can gain the potential benefits of switching without a downside.

Chang (2006) provided a first result in this direction in the infinite-horizon discounted cost setting.

Theorem 2.1 (Chang 2006)**.** *In an infinite-horizon Markov game with discount rate γ, for all $s \in S$, the worst-case performance of π_{ps} is bounded by*

$$\max_{\phi \in \Phi} C(\pi_{ps}, \phi)(s) \leq \min_{\pi \in \Pi} \max_{\phi \in \Phi} C(\pi, \phi)(s) + \frac{\gamma \epsilon}{1 - \gamma},$$

where $C(\pi, \phi)(s)$ is the expected discounted future cost of playing π and ϕ starting from state s, and

$$\epsilon = \max_{s \in S} \Big(\min_{\pi \in \Pi} \max_{\phi \in \Phi} C(\pi, \phi)(s) - \min_{\pi \in \Pi} \min_{\phi \in \Phi} C(\pi, \phi)(s) \Big).$$

It is straightforward to derive an analogous result for our undiscounted finite-horizon setting.

Unfortunately, this bound is not very informative because it involves ϵ, and ϵ can be arbitrarily large. Since ϵ measures the difference between the minimax value and the best-case value, it is small precisely when the maximizer's choice of policy has little influence over the cost; specifically, when the minimizer has a response to every maximizer action that achieves close to the best-case cost.

The question remains of whether the bound is tight. If it is, then there is a fundamental deficiency in minimax policy switching, in that the switching policy may perform arbitrarily poorly compared to a fixed policy. We now show that this is indeed the case, with the following counterexample.

Consider a Markov game with horizon one, and two policies each for the minimizer (π_1 and π_2) and the maximizer (ϕ_1 and ϕ_2). The game begins in state s_0. From there, the game deterministically transitions to state s_{ij}, with i and j determined by the actions π_i and ϕ_j taken in s_0. The cost functions are defined in terms of an arbitrary positive constant c. The instantaneous cost functions for this game (i.e. the costs of the actions chosen by the policies) for each pair of policies in each state are given by

$c(s_{11})$	ϕ_1	ϕ_2
π_1	-1	$c+1$
π_2	c	c

$c(s_{12})$	ϕ_1	ϕ_2
π_1	$c+1$	-1
π_2	c	c

$c(s_{21})$	ϕ_1	ϕ_2
π_1	c	c
π_2	0	$c+1$

$c(s_{22})$	ϕ_1	ϕ_2
π_1	c	c
π_2	$c+1$	0

The 1-horizon cost function for this game is

$C_1(s_0)$	ϕ_1	ϕ_2	max	minimax
π_1	-1	-1	-1	*
π_2	0	0	0	

The minimax fixed policy in this game is π_1, which achieves a worst-case cost of -1. Now consider the behavior of π_{ps}. Starting in s_0, policy π_1 will be chosen since it has a 1-horizon minimax value of -1. Suppose that the maximizer arbitrarily chooses ϕ_1 in response. There is no cost in s_0, and the game deterministically transitions to s_{11}. In s_{11}, π_2 is the minimax choice, and π_{ps} selects it, causing the minimizer to receive a cost of c. The result is the same if the maximizer chooses ϕ_2 in s_0. Thus, the worst case performance of π_{ps} is equal to c, while the fixed policy π_1 achieves a worst case of -1. This shows that minimax policy switching can perform arbitrarily poorly with respect to worst case performance.

Minimax policy switching can incur an arbitrarily high cost because the definition of $C_h(\pi, \phi)(x)$ assumes that the chosen policies will be followed forever. However, this assumption is invalid since π_{ps} can switch policies at future time steps. The above example shows the consequences of this inconsistency. At s_0, policy π_1 looks good under the assumption that both players will follow their chosen policies for the rest of the game. When arriving at s_{11}, however, the algorithm forgets about this assumption and switches away from π_1, receiving a cost of c.

3 Monotone Policy Switching

There are at least two ways to overcome the inconsistency inherent in minimax policy switching. First, when making switching decisions, the algorithm could reason about the

possibility of both players switching policies in the future. This amounts to solving a Markov game over the reduced action space defined by the policy sets. While this may be easier than solving the original Markov game, it will often still be impractical for time-constrained problems. A second approach is to have decisions account for the assumptions made at previous time steps, by considering the possibility that *both* players do not switch away from the current minimax policy pair. In this paper, we consider the second approach, which we call *monotone minimax policy switching*.

Monotone policy switching is nearly identical to minimax policy switching except that at each step it takes into consideration the minimax policy π^* selected in the previous step, and its expected worst-case cost c^*. At horizon h and current state s, monotone switching will only consider switching to a new policy π if the worst case cost of π is better than c^*. Otherwise, π^* is again selected in state s, and c^* is maintained as the expected cost.

More formally, let s_h denote the state with h steps-to-go and $\bar{\pi}_h(\pi^*_{h+1}, c^*_{h+1})$ denote the monotone switching policy at horizon h parameterized by the minimax policy π^*_{h+1} selected at the previous time step and its expected cost c^*_{h+1}. With these definitions, we can define the monotone switching policy recursively in two cases:

If $c^*_{h+1} \leq \text{MMC}_h(\Pi, \Phi)(s_h)$:

$$\pi^*_h = \pi^*_{h+1}$$
$$c^*_h = c^*_{h+1}$$

If $c^*_{h+1} > \text{MMC}_h(\Pi, \Phi)(s_h)$:

$$\pi^*_h = \text{MMP}_h(\Pi, \Phi)(s_h)$$
$$c^*_h = \text{MMC}_h(\Pi, \Phi)(s_h)$$

The action selected by the monotone switching policy is simply the action prescribed by its current policy choice,

$$\bar{\pi}_h(\pi^*_{h+1}, c^*_{h+1})(s_h) = \pi^*_h(s_h).$$

From this definition, one way to view $\bar{\pi}$ is simply as minimax policy switching in which the minimizer has the option to stick to the previous policy pair instead of switching. Note that at the first time step in initial state s_H, where H is the problem horizon, there is no previous policy, so we define $(\pi^*_{H+1}, c^*_{H+1}) = (\text{null}, +\infty)$. This causes the choice at the first time step to behave just as in minimax policy switching.

It turns out that we can prove the desired worst case guarantee for monotone policy switching. We now state the main result, which takes into account the accuracy of the cost estimates used to compute the switching policy. For this purpose, we define a cost function $\hat{C}$ to be *δ-accurate* if

$$|\hat{C}_h(\pi, \phi)(s) - C_h(\pi, \phi)(s)| \leq \delta$$

for all $\pi \in \Pi$, $\phi \in \Phi$, $s \in S$, and $h = 0, \ldots, H$.

Theorem 3.1. *For any Markov game (S, A_1, A_2, P, c), for any state $s \in S$ and horizon H, where $\bar{\pi}$ makes decisions using a δ-accurate cost function $\hat{C}$,*

$$\max_{\phi \in \Phi} C_H(\bar{\pi}(\text{null}, +\infty))(s) \leq \min_{\pi \in \Pi} \max_{\phi \in \Phi} C_H(\pi, \phi)(s) + 2\delta.$$

Proof. (sketch) Let $\bar{\pi}^i$ denote a restricted version of the switching policy $\bar{\pi}$ that is allowed to consider switching only in the first i time steps. We will prove the theorem by induction on the number of switches i.

In the base case when $i = 1$, $\bar{\pi}^1$ can switch only in the first time step, and so the algorithm will play the fixed policy $\bar{\pi}^1 = \arg\min_{\pi \in \Pi} \max_{\phi \in \Phi} \hat{C}_H(\pi, \phi)(s)$, which under our assumption about $\hat{C}$ will have a true worst case cost of at most $\min_{\pi \in \Pi} \max_{\phi \in \Phi} C_H(\pi, \phi)(s) + 2\delta$. This completes the base case.

For the inductive step, we assume that

$$\max_{\phi \in \Phi} C_H(\bar{\pi}^{i-1}, \phi)(s) \leq \min_{\pi \in \Pi} \max_{\phi \in \Phi} C_H(\pi, \phi)(s) + 2\delta.$$

First, observe that $\bar{\pi}^i$ and $\bar{\pi}^{i-1}$ are equivalent for the first $i - 1$ steps. So $S_{H-i}(\bar{\pi}^i, \phi^*) =^d S_{H-i}(\bar{\pi}^{i-1}, \phi^*)$, where $=^d$ means "equal in distribution." Call this random variable S_{H-i} for short.

Let s_{H-i} be an arbitrary instantiation of S_{H-i}. We now need to compare $\bar{\pi}^i$ and $\bar{\pi}^{i-1}$ at state s_{H-i}. We can identify two cases. If $\bar{\pi}^i(s_{H-i}) = \bar{\pi}^{i-1}(s_{H-i})$, then the conclusion trivially follows from the inductive hypothesis.

Now consider the case when $\bar{\pi}^i(s_{H-i}) \neq \bar{\pi}^{i-1}(s_{H-i})$. Let $\pi^i = \bar{\pi}^i(s_{H-i})$ be the concrete policy selected at horizon $H - i$ and $\phi^* \in \Phi$ be a fixed but arbitrary maximizer policy. We then have the following:

$$\begin{aligned} C_{H-i}(\bar{\pi}^i, \phi^*)(s_{H-i}) &\leq \hat{C}_{H-i}(\bar{\pi}^i, \phi^*)(s_{H-i}) + \delta \\ &< c^*_{H-i+1} + \delta \\ &\leq \min_{\pi \in \Pi} \max_{\phi \in \Phi} C_H(\pi, \phi)(s) + 2\delta. \end{aligned}$$

The second inequality holds due to our assumption that the policy switched at horizon $H - i$. The final inequality holds because c^*_h is non-increasing across time steps and the initial value c^*_H is guaranteed to be no worse than $\text{MMC}_H(s) + \delta$. Since ϕ^* was an arbitrary maximizer policy and s_{H-i} was an arbitrary instantiation of S_{H-i}, the result follows. □

4 Experiments

RTS games are widely popular video games that involve building large economies to support military production and tactics in order to defeat an opponent. They pose serious challenges to existing AI techniques and most computer players use relatively inflexible hand-coded strategies with no look-ahead and minimal autonomy (Buro and Churchill 2012). We have constructed a game-playing agent for the free RTS game *Wargus* that uses policy switching to manage its high-level strategy. This adds a level of lookahead and reactivity to the agent that purely script-based solutions generally lack, while leveraging a designer's strategic knowledge and existing policies via the required policy sets.

Our policy-switching agents uses an abstract, approximate simulator of the game engine to estimate the cost of each strategy pair from a state. The simulator is fast enough to allow for faster than real-time game-play using the switching approach (with 9 policies per player). We present two sets of results: 1) Perfect model results, where

the simulator is used in place of the game engine for evaluation of the agents, modeling a situation where our cost estimates are perfect; 2) Game engine results, where evaluation is done using the real game engine, in which case the simulator only provides approximate cost estimates. The details of the simulator and RTS agent infrastructure are beyond the scope of this paper, but can be found in (King 2012).

Policy Set. Our switching agents are provided with a set of 9 hand-coded policies. Each policy is a prioritized assignment of a certain number of military units to each of several goal locations. We defined three goal locations for each game map: the friendly base ("base"), the enemy base ("enemy"), and the choke point separating the bases ("chokepoint"). Though simple, this formulation gives us quite a bit of flexibility. For example, defensive strategies are obtained when "base" and "chokepoint" have higher priority than "enemy". Early aggression can be specified by prioritizing "enemy" and assigning a small number of units (so that they will be ready sooner). Our experiments used the same 9 policies for both the minimizer and maximizer. Each such policy instantiates a hierarchical controller that issues appropriate actions to the game engine or simulator.

Perfect Model Results. We evaluated the switching algorithms on two game maps from the Wargus distribution called *2bases* and *the-right-strategy* using our simulator in place of the game engine. Costs correspond to the difference in total unit hit points at the end of a game (negative values indicate wins). Table 1 shows the worst case cost for *monotone* switching and ordinary *minimax* switching when played against each of the 9 policies in the policy set on each map. We also played all 9 policies against one another to find the minimax policy and cost within that set, denoted as *Fixed* in the table, which is zero since the map is balanced and policies can play against themselves. The results show that both switching strategies significantly outperform the fixed strategies in terms of worst-case performance, showing the potential benefits of switching. We also see that in one map the monotone strategy outperforms the ordinary minimax player. The reason for the improved performance is unclear and perhaps coincidental since an analysis of the games did not show any obvious flaws of the minimax player.

Score	Fixed	minimax	monotone
2bases	0	-8122	-8122
the-right-strategy	0	-2702	-5375

Table 1: Results using the simulator for game play (allowing for perfect cost estimation) on two different game maps.

Game Engine Results. Table 2 shows the same results as above but in games played in the real game engine, so that the simulator only provides approximate cost estimates. Our analysis of the simulator shows that it is accurate for some strategies, but less accurate for others. A major source of error is that the simulator estimates battle outcomes based only on unit numbers and types, while in reality terrain has a large impact. Whereas policy switching outperformed the minimax fixed policy in simulation, the results show that the switching strategies can do worse in terms of game costs compared to the best fixed strategy. Note that this is not inconsistent with the theory, which accounts for loss in performance given inaccurate cost estimates. Our analysis showed that these results are due to some systematic biases of the simulator when evaluating particular policy pairs in certain situations. This causes the switching policies to sometimes switch to a policy that is actually inferior. The score differences, however, do not correspond to large differences in win rates. We played 50 games between all pairs of agents including the fixed strategies and switching strategies and measured the worst case win rate for each agent. Table 3 shows the results and illustrates that the win rates for the methods are comparable, though the best fixed policy is still the best overall. Finally, Table 3 also shows the win rate of the switching policies against the built-in Wargus AI. The switching strategies win nearly always on the first map and win 57% of the games on the second map.

Score	Fixed	minimax	monotone
2bases	630	242	749
the-right-strategy	86	588	814

Table 2: Gameplay results using the real game engine on two different game maps.

2bases	Fixed	minimax	monotone
Fixed	46%	49%	45%
minimax	-	-	48%
built-in	-	100%	100%
the-right-strategy			
Fixed	50%	45%	42%
minimax	-	-	42%
built-in	-	57%	57%

Table 3: Empirical win rates for the *column* player for games between the minimax fixed policy, the two switching policies, and the built-in Wargus AI. Win rates are statistically identical within rows.

5 Summary and Future Work

Unlike in the single-agent, MDP setting, policy switching based on a local optimality criterion is not guaranteed to improve upon the best fixed policy in Markov games. We have shown that there exist Markov games in which naive policy switching achieves the worst possible result. To remedy this, we have proposed *monotone minimax policy switching*, which is guaranteed to perform at least as well as the best fixed policy. When the opponent policy set captures the typical range of strategies that will be encountered, this is a strong guarantee. However, we are also interested in developing switching algorithms that provide guarantees with respect to a bounded amount of switching by the agents, which would allow for high performance even if the set of base policies is incomplete. A key issue illustrated by our experiments is dealing with inaccurate simulations. One interesting approach is to learn to correct simulator biases that are observed during game play. We are also interested in principled ways of integrating a model of an opponent's behavior, possibly learned from historical data (Dereszynski et al. 2011), into the policy switching framework.

References

Buro, M.; and Churchill, D. 2012. Real-time strategy game competitions. *AI Magazine* 33(3):106-108.

Chang, H.; Givan, R.; and Chong, E. 2004. Parallel rollout for online solution of partially observable Markov decision processes. *Discrete Event Dynamic Systems* 14:309341.

Chang, H. S. 2006. On combining multiple heuristic policies in minimax control. In *17th International Symposium on Mathematical Theory of Networks and Systems (MTNS 2006)*.

King, B. 2012. Adversarial planning by strategy switching in a real-time strategy game. Master's thesis, Oregon State University.

Dereszynski, E.; Hostetler, J.; Fern, A.; Dietterich, T.; Hoang, T.; and Udarbe, M. 2011. Learning probabilistic behavior models in real-time strategy games. In *Seventh Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE 2011)*.

Sailer, F.; Buro, M.; and Lanctot, M. 2007. Adversarial planning through strategy simulation. In *IEEE Symposium on Computational Intelligence and Games (CIG 2007)*, 80–87.