Adversarial Policy Switching with Application to RTS 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

Complex games such as RTS games are naturally formalized as Markov games. Given a Markov game, it is often possible to hand-code or learn a set of policies that capture the 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. 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, we show that this switching policy can have arbitrarily poor worst case performance. In response, we describe a modified algorithm, monotone policy switching, whose worst case performance, under certain conditions, is provably no worse than the minimax fixed policy in the set. We evaluate these switching policies in both a simulated RTS game and the real game Wargus. The results show the effectiveness of policy switching when the simulator is accurate, and also highlight challenges in the face of inaccurate simulations.

1 Introduction

In some 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. A successful agent requires some form of abstraction to reduce the effective branching factor. One approach is to create a set of fixed policies, each of which fully specifies the primitive actions to take in each state, that captures the diversity of possible behaviors or strategies. The reasoning problem is then reduced to choosing among the policies.

It is often possible to hand-code or learn an abstract simulator of the game that can be used to estimate the outcome of playing two policies against one another starting from an arbitrary state. Given such a simulator, one could simulate all policy pairs against one another from the initial game state and pick the policy that achieves some criterion of optimality, such as the minimax policy. However, we might wonder whether we can do better than choosing a single, fixed policy by occasionally re-evaluating our choice and possibly switching to a different policy. Since policy switching does not consider future actions when deciding whether to switch, its cost is linear in the length of the game, compared to the exponential cost of solving the full Markov game. In this work, we investigate the formal properties of one such *policy switching* approach, and provide a preliminary evaluation of its effectiveness in an RTS game.

Prior work (Chang, Givan, and Chong 2004) has investigated policy switching in the setting of (non-adversarial) Markov decision processes (MDPs). In that work, at each state s a simulator is used to estimate the value of each policy in a given set and the policy with the highest value is executed. The key result was to show that the value of the switching policy is never worse than the value of the best policy in the set. Further, in practice the switching policy is often significantly better than any individual policy in the set. In less closely related work, Comanici and Precup (2010) study how to learn a meta-policy for switching between policies in a set. This approach does not require a simulator to estimate policy values, but rather requires substantial experience in the target environment for learning and a meta-policy representation that facilitates generalization.

Policy switching has also been investigated in the adversarial setting of Markov games. Sailer, Buro, and Lanctot (2007) used a policy switching approach to switch between military strategies in an abstract RTS game. Given a set of strategies, they simulate all pairs against one another at each decision epoch and select one according to either a minimax or Nash criterion. This approach was shown to work well, outperforming the individual policies provided to the system. A formal analysis of this type of switching approach was provided by Chang (2006), who gave a bound on the difference between the worst-case (i.e. minimax) performance of the switching policy versus the best fixed policy in the set.

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, the straightforward way of formalizing policy switching for 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, up to an error term describing

*This document is a correction to a paper of the same name that appeared in the AIIDE 2012 Workshop on Artificial Intelligence in Adversarial Real-time Games. The published version omitted the Acknowledgements section.

the accuracy of the simulator. 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

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

where S_k is a random variable denoting the state when there are k steps to go.

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. 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 $\max_{\phi \in \Phi} C_h(\pi^*, \phi)(s) = \text{MMC}_h(\Pi, \Phi)(s)$. Here we assume that ties are broken deterministically using a lexicographic order over policies.

The *minimax switching policy* denoted by $\pi_{\text{ps}}(s, h)$, from 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. In practice the cost function $C_h(\pi, \phi)(s)$ cannot be evaluated exactly and hence Monte Carlo simulation using a possibly approximate simulator is used to estimate costs required for computing the minimax policy.

A question now is what guarantees if any can be made about the performance of π_{ps}. Recall that in MDP policy switching, one can show that the switching policy will do no worse than the best policy in the set. An analogous result for minimax policy switching would show that the switching policy has a worst case behavior (with respect to Φ) that is no worse than the best worst case behavior of any fixed policy in Π (i.e. the minimax policy). That is, we would like to say 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 is sufficiently rich to capture 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, showing a looser bound. Note that while the original result is in an infinite-horizon discounted cost setting, the result is easily modified for our finite horizon setting. His bound is in terms of the difference between the cost of the minimax policy and the best possible cost for the minimizer assuming a helpful adversary,

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

where $C(\pi, \phi)(s)$ is the expected discounted future cost of playing π and ϕ starting from state s. Using this error term, Chang then shows that the worst-case performance of the switching policy is bounded by

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

for all $s \in S$, where γ is the discount rate.

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. Intuitively, when this is the case, the minimizer does not need to consider future moves when deciding the current move, since there will always be a near best-case response to anything the maximizer does.

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 deterministic Markov game with two states, 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 is 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, even in a game with only terminal costs.

Minimax policy switching achieves an arbitrarily high cost because the definition of $C_h(\pi, \phi)(x)$, which is used to select a policy at horizon h, assumes that the chosen policies will be followed forever. However, this assumption is inconsistent with the fact that at future time steps π_{ps} has the option of switching policies. In the above example, we see the consequences of this inconsistency. At s_0, the 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 decides to move 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 in order to improve its worst case performance. First, when making switching decisions, the algorithm could reason about the possibility of both players switching strategies at future times. Reasoning about those possibilities, however, amounts to solving the Markov Game, which is generally not going to be practical and does not exploit the provided policy sets. A second approach is to have future 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 this 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 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 (with $h+1$ steps-to-go) and its expected cost c^*_{h+1}. To simplify notation, we will use $\theta_h = (\pi^*_h, c^*_h)$ to denote the parameter vector.

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 can suppress the option to switch if switching would be worse for the minimizer than sticking to the policy pair from the previous step. 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 θ_{H+1} to be the null vector $\theta_{\text{null}} = (\text{null}, +\infty)$. 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, at least in the case of deterministic transitions. We conjecture that it also holds in the case of stochastic transitions. We now state the main result.

Theorem 3.1. *For any Markov game (S, A_1, A_2, P, c) such that P is deterministic, for any state $s \in S$ and horizon h,*

$$\max_{\phi \in \Phi} C_h(\bar{\pi}_h(\theta_{\text{null}}), \phi)(s) \leq \min_{\pi \in \Pi} \max_{\phi \in \Phi} C_h(\pi, \phi)(s).$$

Conjecture. *Theorem 3.1 holds for stochastic P as well.*

4 Experiments

We have constructed a game-playing agent for the free RTS game *Wargus* that uses policy switching to manage its high-level strategy. Because of the complexity of RTS games, all full-game agents we are aware of, including ours, have relied on scripted behaviors to perform most gameplay tasks. A key advantage of the policy switching approach is that it offers a principled way of incorporating a limited amount of AI into a script-based system: the AI system can choose between high-level scripted behaviors, rather than attempting to produce a similar behavior from first principles.

Our policy-switching agent uses a simulator to estimate the value of each strategy pair, and selects the strategy with

the best simulated value. We present both simulation results and results from gameplay against the built-in Wargus AI. Due to space constraints, we can present only a representative selection of our results; the full results are available in (King 2012).

4.1 Strategy Set

We define a strategy as a prioritized assignment of a certain number of units to each of several goal locations. For these experiments, we defined three goal locations for each map: the friendly base ("base"), the enemy base ("enemy"), and the choke point separating our base from the enemy ("chokepoint"). A strategy specifies the number of units we want to assign to each goal, and a priority ordering such that goals with higher priority are pursued first. Though simple, this formulation gives us quite a bit of flexibility in defining strategies. For example, defensive strategies are obtained when "base" and "chokepoint" have higher priority than "enemy". Similarly, early aggression can be specified by prioritizing "enemy" and assigning a smaller number of units (so that they will be ready sooner). Some representative strategy definitions are given in Table 1.

	Goal priority			Units per goal		
	base	enemy	chkpt.	base	enemy	chkpt.
0. balanced 7	1	3	2	7	7	7
4. rush 7	1	3	2	5	7	3
7. offensive 5	0	1	0	0	5	0

Table 1: Selected strategies from our strategy set.

At regular intervals throughout the game, the agent simulates various strategy choices against one another, and switches strategies if the current strategy is found to be worse than an alternative. We used two agents for the experiments. The *minimax* agent implements simple minimax policy switching as described by Chang (2006), while the *monotone* agent implements our monotone policy switching algorithm described above.

4.2 Simulation Results

We implemented a deterministic simulator that can compute the value of different strategy pairs. Our first results compare the *monotone* switching algorithm to the ordinary *minimax* switching algorithm using the simulation results as the measure of success. We ran simulations for two different game maps, called *2bases* and *the-right-strategy*. The results for the fixed minimax policy and the two different policy switching algorithms are shown in Table 2.

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

Table 2: Simulation results comparing the fixed minimax policy to the two switching algorithms on two different game maps.

Both switching algorithms outperform the minimax fixed policy, and the *monotone* algorithm outperforms the *minimax* switching algorithm on one of the maps.

4.3 Gameplay Results

We also examined the algorithms' performance in actual gameplay on the same two maps. Because the values of actions are computed with a simulator, an inaccurate simulator could affect the results. Our experiments show that our simulator is accurate for some strategies, but less accurate for others (Figure 1). One likely source of error is that the simulator estimates combat strength based only on unit numbers and types, while in reality terrain has a large effect on the outcomes of battles.

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

Table 3: Gameplay results comparing games scores for the fixed minimax policy and the two switching algorithms on two different game maps.

2bases	Fixed	minimax	monotone
Fixed	46%	49%	45%
minimax	-	-	48%
monotone	-	52%	-
built-in	-	100%	100%

the-right-strategy	Fixed	minimax	monotone
Fixed	50%	45%	42%
minimax	-	-	42%
monotone	-	58%	-
built-in	-	57%	57%

Table 4: Empirical win rates for the *column* player for matchups between the minimax fixed policy, the two switching policies, and the built-in Wargus AI. Win rates are statistically identical within every row.

Whereas policy switching outperformed the minimax fixed policy in simulation, the results are inconclusive for actual gameplay (Table 3). We attribute this to the inaccuracy of the simulator. It seems that tactical factors that the simulator does not account for have a significant impact on performance.

The win rates of the various algorithms when played against one another (Table 4) show that the switching policies achieve a similar win rate to the minimax fixed policy. Policy switching is not guaranteed to be strictly better than the fixed policy, so these results are consistent with theory, accounting for inaccuracies in the simulator.

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 performance versus a fixed policy in adversarial Markov games. We have shown that there exist Markov games in which straightforward policy switching will produce the worst possible result. To remedy this, we have proposed *monotone minimax policy switching*, which is guaranteed to perform at least as well as any fixed policy when the simulator is perfect. For imperfect simulation, there is a

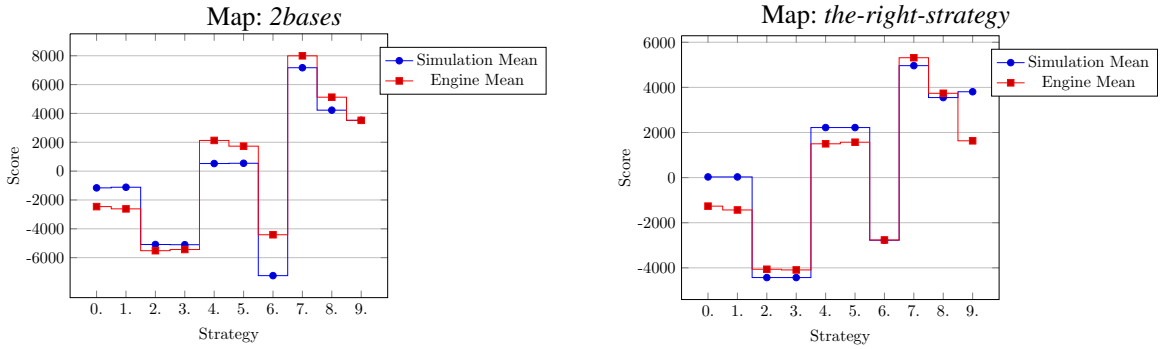

Figure 1: Comparison of simulated versus actual minimax costs for minimizer policies on maps *2bases* (left) and *the-right-strategy* (right).

potential additional worst-case cost that depends on the simulation accuracy.

Our *monotone* algorithm achieves its performance guarantee with respect to 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 opponent switching. Our results also point to the importance of the simulator for the policy switching approach. Currently the simulator is fixed and does not adapt even when large simulation errors are observed. We would like to integrate the system with a simulator learning component that improves simulation accuracy over time. Finally, the current approach does not take into account an opponent model, which could provide evidence about the likelihood that the opponent is following different policies. We would like to understand how to best take such a model into account in a policy switching framework.

Acknowledgements

This work was funded in part by NSF grant IIS-0905678 and by ARO grant W911NF-08-1-0242. The views and conclusions contained in this document are those of the authors and do not necessarily represent the official policies of ARO, NSF, or the United States Government.

Jesse Hostetler was supported in part by a scholarship from the ARCS Foundation of Portland, Oregon.

References

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)*.

Comanici, G., and Precup, D. 2010. Optimal policy switching algorithms for reinforcement learning. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, 709–714.

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

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.