PRELIMINARY THOUGHTS ON THE APPLICATION OF REAL-TIME AI GAME-TREE SEARCH TO CONTROL *

Todd W. Neller

Knowledge Systems Laboratory, Gates Building 2A, Stanford University, Stanford CA 94305-9020
email: neller@ksl.stanford.edu

Abstract: We introduce a formal definition of a general hybrid system control game, and a decision procedure for an agent which uses simulation to inform game-playing decisions in real-time. We discuss heuristics informing the estimation of expected utilities of actions, describe metalevel reasoning criteria which direct game-tree expansion according to decision relevance, and point out challenges in applying ideas from AI real-time game-tree search to control.

Keywords: real-time AI, game theory, hybrid systems, model-based control, artificial intelligence, agents

1. INTRODUCTION

The fundamental question of this research is: "How can a rational agent use simulation to choose intelligent actions for hybrid system control in real-time?" In this paper, we introduce a formal framework which views the reasoning of such an agent as real-time game-playing. The *general hybrid system control game* formalism is more general than that of discrete- and continuous-time dynamic games (Başar and Olsder, 1995), in that it allows general hybrid system behavior and strategies involving both discrete and continuous actions.

As problem frameworks become more general, constraints which are helpful in problem solving become fewer, so we must elucidate tradeoffs and architectural design decisions which make such a framework practical for application. This paper describes a spectrum of simulation-based approaches which range from weak, uninformed game-tree search to strong, informed (heuristic) approaches which can benefit from other forms of analysis. Since the expected utility of the agent's external game actions is time-dependent, the expected utility of performing internal simulation actions in order to deliberate about such external actions is time-dependent as well. We later provide a general decision procedure for such reasoning with simulation.

Defining a rational agent as that which chooses actions which maximize expected utility, our real-time limitations force us to examine the expected utility of the act of simulation itself. For example, imagine you are driving on a highway through a city with a prior notion of a best route. Suddenly, you see a sign for an impending exit which presents the possibility of a much better route. Another exit with similar possibilities follows. You're faced with a decision on how to deliberate as you quickly grab your map: Should you examine the route from the immediately approaching exit, or is your deliberation better focussed on examining the route from the exit beyond? Such deliberation about deliberation, or *metalevel rea-*

* This work was supported by the Defense Advanced Research Projects Agency and the National Institute of Standards and Technology under Cooperative Agreement 70NANB6H0075, "Model-Based Support of Distributed Collaborative Design".

soning, is necessary to focus computational deliberation in promising directions.

Of course, metalevel deliberation is itself deliberation and one could regress infinitely were it not for the fact that such metalevel deliberation also needs to be justified in terms of maximization of expected utility. Metalevel reasoning must use less runtime than the reasoning it obviates. Encouraging research by Russell and Wefald (Russell and Wefald, 1991) indicates that, provided good heuristics for focusing search, metalevel reasoning in game-tree search justifies its cost. Their research is the inspiration for our simulation approach to game-playing. Of course, many different types of analysis can better inform play of a hybrid system control game, and we shall see that such analysis is beneficial for forming the heuristics to productively focus metalevel deliberation. We will suggest domain-independent and -dependent heuristics for informing the metalevel of game-tree search.

Before delving into formalism and discussion, we should first discuss something of the motivation of this research. Why have we chosen to pursue our particular line of inquiry? First, controllers are being called on to meet more complex objectives than ever before. Particularly, systems which are highly reconfigurable may present the control engineer with such complexity as cannot be humanly or analytically overcome. We believe that enabling controllers to improvise control with online models will push the envelope of applicability of both artificial intelligence and control engineering. Second, we observe that larger-scale real-time constraints may play a role in the design and testing of controllers. Such techniques could spawn a new generation of design tools which deal with real-time design constraints explicitly within their reasoning framework. In short, we see an opportunity for the fruitful application of real-time AI game-tree search to practical control problems.

The paper is structured as follows: In Section 2, we formally define a hybrid system control game. The remainder of the paper informally introduces the concepts of AI real-time game-tree search and discusses the challenges in applying such ideas to hybrid system control game play. In Section 3, we discuss different means of evaluating game-trees. In Section 4, we discuss principles for expanding such game-trees in ways most relevant to rational decision-making. In Section 5, we bring real-time constraints into relevance consideration and present a general decision procedure for simulation-based hybrid system control game play. Finally, in Section 6 we describe ways of heuristically informing the calculation of estimates of expected utilities in the game-tree.

2. HYBRID SYSTEM CONTROL GAME

A common example of a hybrid system is a physical system controlled through mode-switching. Such a system may evolve through a sequence of continuous phases with instantaneous [1], discrete changes of control modes in between. If we assume that mode switching is under control of one or more agents (an agent being a controller, a disturbing influence, etc.), then the control of such a system may be viewed as a game where each agent is a player seeking to maximize some utility measure as their game score.

Consider this simple illustrative example of a hybrid system: A car is traveling along a road with a stop light which is known to have one of a constrained set of possible states and behaviors. The driver knows bounds on the different delay times associated with the different states of the light. From the driver's initial observation of the traffic light at a distance, the driver has a collection of utility influences: travel progress, fuel efficiency, safe speed, avoiding speeding tickets, and staying out of the intersection when the light is red. Suppose the driving agent sets its driving policy (accelerate, cruise, or decelerate) safely according to worst-case possible traffic light timings and delays. This then can be viewed as a sort of game where the driving agent is a player seeking to optimize its score (the utility of its performance), and the traffic light controller is an adversary seeking to reduce the driver's score by turning the light red when the car is in the intersection.

Informally, our hybrid system control game evolves with continuous and discrete (instantaneous) behaviors. For all non-terminal game states, the continuous behaviors define a unique trajectory until such time as a terminal state is reached. For any given state, some discrete transitions (also called actions) may be either forced by the system or enabled for player(s) to enact. We call a player-enacted discrete transition a *move*. Given an initial state, this then defines a possibly infinitely-branching tree of trajectories, where branches are caused by player moves. Given scoring functions over trajectories, and termination conditions for the game, we encode the desirability of each possible system trajectory for each player.

We seek to most generally and formally express what it means to play a control game with a hybrid system. Our formalism is based on that of Branicky's controlled hybrid dynamical system (CHDS) (Branicky, 1995). With a desire to both (1) preserve the expressive power of a CHDS, and (2) create a formal language suited to expressing hybrid system game play, we have found it most natural to describe transitions in terms of actions,

[1] for practical purposes of modeling and simulation

and allow for the possibility of multiple actions being taken simultaneously. We formalize a *general hybrid system control game* G as follows:

$$G = [Q, \Sigma, S_{\rm I}, A, V, p, W, F, E, R, s, \omega]$$

where

- Q is a set of discrete (or index) states.
- $\Sigma = \{\Sigma_q\}_{q \in Q}$ is the collection of constituent dynamical systems indexed by Q, where each dynamical system $\Sigma_q = [X_q, \Gamma_q, \phi_q, U_q]$ (or $\Sigma_q = [X_q, \Gamma_q, f_q, U_q]$), where X_q is an arbitrary topological space (the state space of Σ_q), transition semigroup Γ is a topological semigroup, U_q is the control set, transition map $\phi_q : X_q \times \Gamma_q \times U_q \to X_q$ satisfies identity, semigroup, and continuity properties, and transition function f_q is the generator of extended transition function ϕ_q. $U = \{U_q\}_{q \in Q}$ is the total control set. In short, each constituent dynamical system is defined as a standard controlled dynamical system.
  We define the total state space $S = \bigcup_{q \in Q}(q \times X_q)$.
- $S_{\rm I} \subseteq S$ is the set of possible initial states of the system.
- A is the set of discrete actions $\alpha_1, \alpha_2, \ldots$.
- V is the transition control set.
- $p \in \mathbb{Z}^+$ is the number of players (or agents). We define player index set $P = \{1, \ldots, p\}$.
- $W : U \cup V \to P$ maps the control sets to the players that control them.
- $F : S \to A \cap \emptyset$, the forcing function, maps each system state to a single forced (required) action or to an empty set indicating that no action is forced.
- $E : S \to 2^{P \times A}$, the enabling function, maps each system state to a set of pairs of players and actions which the players are enabled to enact.
- $R : S \times 2^A \times V \to S$, the result function, maps a system state and actions taken in that state to a set of possible resulting system states. Since forced actions take precedence over enabled actions, then for all A' and s such that $A' \subseteq A$, $s \in S$, $F(s) \neq \emptyset$, and $F(s) \in A'$, we require that $R(s, A') = R(s, F(s))$.

  We define a trajectory $\tau = [s_0, t_0, h, t_{n+1}]$ to be a unique evolution of the game where
  - $s_0 \in S$ is the start state (with q_0 and x_0 the discrete and continuous start states, respectively),
  - $t_0 \in \Gamma_{q_0}$ is the initial time,
  - $h = [[p_1, \alpha_1, t_1, s_1], \ldots, [p_n, \alpha_n, t_n, s_n]]$ is the move history, a list of $n \geq 0$ moves, $[p_i, \alpha_i, s_i, t_i]$ indicating that player p_i took enabled action α_i at time t_i to reach state s_i (for $1 \leq i \leq n$, $p_i \in P$, $\alpha_i \in A$, $t_i \in \Gamma_{q_i}$, $t_i \geq t_{i-1}$, and $s_i \in S$).
  - $t_{n+1} \in \Gamma_{q_{n+1}}$, $(t_{n+1} \geq t_n)$ is the final time, and
  - the system evolves from the initial state and time according to system dynamics (including forced discrete actions) except for enabled actions taken in the order listed until the final time.

  $\mathcal{T}$ denotes the set of all legal trajectories allowed by the system. Complete trajectories begin with an initial state and end with a single terminal state. Other legal trajectories are called partial trajectories. Trajectories beginning with an initial state are called initial trajectories.

- $s : P \times \mathcal{T} \to \mathbb{R}$ is the scoring function, mapping players and trajectories to scores (also called utilities),
- $\omega \subset S \times \mathbb{R}$ is the termination set, the set of states and times for which the game ends. We require $\exists t \forall s (t' \geq t \to [s, t'] \in \omega)$, i.e. a game must eventually end.

Within this paper, we will restrict our attention to a subset of general hybrid system control games, here called *hybrid system control games*, for which the set of discrete states Q is countable, each $X_q = \mathbb{R}^{d_q}$ (i.e. continuous state spaces are real-valued), $d_q \in \mathbb{Z}^+$, $\Gamma_q = \mathbb{R}^+$ (the system evolves forward in real-valued time), $S_{\rm I}$ is a singleton (contains a single initial state), R maps to a singleton (action results are deterministic), $U = V = \emptyset$ (no control sets), and behavior is non-Zeno (players must allow time to progress). Note that nondeterministic system behavior may be modeled by introducing a "chance" player controlling nondeterministic choices.

In a given state and time, game-playing consists of each player choosing between possible moves and inaction in order to maximize its score. Formally, a player maps initial trajectories and possible moves for the player in the final state of the trajectory to one such possible move or $\emptyset$ (inaction). If players' scores sum to zero for each trajectory, the game is zero-sum adversarial. If players' scores are equal for each trajectory, the game is identically cooperative. Each game represents a player's beliefs about how the world evolves, how the players can affect change in the world, and the intentions of each player.

3. IDEAL AND BOUNDED RATIONAL PLAY

Ideally a game-playing agent would be omniscient of all possible system behaviors and would play optimally with respect to models of the other agents' decision procedures. Simulation alone cannot give us such omniscience. However, we can use simulation to sample the infinitely branching game-tree and judge between alternatives.

Given a sampled game-tree, decision-making consists of propagating trajectory scores bottom-up through the tree according to the expected behavior of each player given choices at each branch. We could simply randomly sample possible trajectories, triggering random hypothetical moves for each player and hoping that the trajectories most relevant to each player's decision is contained in the sampling. However, we are not committed to randomly sampling the game-tree. This idea is introduced as a baseline of comparison for better approaches to come. At heart, we are attempting a complex type of optimization on two-levels. First, we wish to optimize our agent's choice of simulation according to relevance to decision making. Second, we wish to optimize our agent's choice of action according to the game-tree.

Although our game has termination conditions such that all trajectories are finite, our computational resources are limited and we generally cannot simulate complete trajectories and expect to gain an adequate sampling of possible behaviors. For this reason, we allow partial simulation, i.e. simulation which can terminate before the game terminates. Let us call the final state of each trajectory in the game-tree a *leaf*. The utility of reaching a terminal leaf is computed with a trajectory scoring function, but what of the expected utility of reaching non-terminal leaves? Ideally, the expected utility of a non-terminal leaf would be the score expected from taking ideal actions until game termination. However, we cannot expect to compute this ideal value. How can we estimate the expected utility of reaching a non-terminal node?

We could assume that the current trajectory score is a good indication of the relative utility of following that trajectory. This greedy sort of game-playing approach is vulnerable to short-term gain in score at the expense of greater long-term loss, but may be preferable for simplicity and effectiveness when short-term ramifications of actions are key. Another approach would be to limit our search to an arbitrary time horizon, restricting the agent's deliberation to only consider trajectories up to a given time limit. Knowledge of one's problem domain is necessary to set a good time horizon.

When longer-term ramifications are important, or choice of a time horizon is not obvious, we then need to have some form of heuristic estimate of the expected score from a non-terminal leaf until the game's end. Such heuristic estimates can possibly be efficiently derived from analysis of the simulation model, from previous experience, or prior expectations of the designer. We revisit this topic in Section 6. Note that if we uniformly overestimate the score, we can expect shorter branches to look preferable. If we uniformly underestimate the score, we can expect longer branches to look preferable. In fact, not only do we need such an estimate, we also need a way to express our uncertainty about the estimate as well.

In acting rationally under uncertainty, the agent must deliberate between actions of varying estimated expected utility with possibly differing certainty for such estimations. A densely sampled space should give a more certain estimate than a sparsely sampled one. The certainty of the heuristic estimate expected utility of an action informed by sound mathematical analysis should be greater than the certainty of estimates based on general rules-of-thumb. In providing both the estimate and a measure of certainty of the estimate, the agent can decide which estimates are worth performing simulation to reduce uncertainty. An uncertain estimate showing great promise may be worth following up if it has a high probability of being better than the current best action. Two very certain and similar estimates may not be worth further deliberation. These thoughts are expressed mathematically in (Russell and Wefald, 1991), and point to the importance of reasoning about the relevance of computation to an agent's decision.

In summary, designing a game-playing agent with bounded computational resources challenges us to be clever in our heuristic estimation of expected utilities, and in the choice of additional simulations to reduce the uncertainty of such estimations.

4. CHOOSING RELEVANT SIMULATIONS

Assume we are given a finite game-tree of trajectories where each of the infinite states represented has associated with it a probabilistic estimation of the expected utility of reaching that state. Except in degenerate cases, an infinite subset of these states enable actions which can be taken and offer possible simulations the agent may choose to perform to further inform its decision on how and when to act next. Ideally, the agent should choose to compute the simulation which is most expected to add value to its decision. Given probabilistic estimates of expected utilities, how can the agent choose simulations most relevant to maximizing utility?

For brevity, we will summarize some main points of (Russell and Wefald, 1991): At any point in time, the game-playing agent faces a decision between a set of internal computational actions and a commitment to a current default external game action. We make the simplifying assumption that our algorithm is *meta-greedy*, that is, our algorithm chooses the next action maximizing expected utility. (Such meta-greediness can mislead us if we place too much confidence in (i.e. underestimate uncertainty of) myopic utility estimates.) We then wish to greedily choose a simulation with greatest expected informational value for our decision. *The information a simulation is expected to gain has value to the extent that it will cause a change of the agent's belief in the best external action, and to the extent that the new best external action has significantly greater utility than the old.* Therefore, a simulation expected to increase expected utility of the agent's current best action, or decrease expected utilities of alternative actions has no direct, immediate informational value for the agent's decision.

Unlike the problem domain in (Russell and Wefald, 1991), we cannot assume the property of *subtree independence*, where expanding the game-tree affects the utility estimate of only one action at the root of the expanded subtree. For instance, in updating the player utility estimates of an action taken at a given time, we must also update the utility estimates of the same action taken at nearby times. As another example, discovery of a particularly (un)desirable set of states could affect the expected utilities of subtrees which possibly reach such states. In the worst case, each simulation could cause the agent to update all of its utility estimates. For example, consider a car crossing a bridge which suddenly detects that a section of the bridge is missing. One of the challenges of applying these ideas to real-time control is to develop an efficient means of propagating utility estimate updates through the tree. In fact, one does not need to propagate updates where they are provably irrelevant to the decision at hand.

The lack of subtree independence which presents a challenge for updating estimates of expected utilities is also cause for optimism that such game-playing approaches can be feasible. Such interdependence between the infinite possible subtrees gives us hope that our finite sampled game-tree can adequately inform our decision over the infinite possibilities which remain unsimulated.

5. DELIBERATION VERSUS COMMITMENT TO ACTION

Were it not for real-time computational constraints, one might think that one would simply perform simulations with greatest informational value until such a value went below a certain threshold. However, the utility of committing to an action at a specific time is meaningless once that time is in the past. Since the utility of such a commitment is time dependent, the utility of deliberating about that commitment is time dependent as well. Recall our highway exit deliberation example. Time spent deliberating whether or not to take the first exit is time not available to deliberate whether or not to take the next exit. Using the traffic light example, one would attach far less value to a search for an action yielding a marginally greater fuel efficiency than a search for an action which would successfully stop at a red light. Simply put, we must work with an explicit notion of computational value in our decision making framework. Let us now sketch the basic decision procedure we have been discussing:

GT := init_simulation(default_move(S_{I}))
while game not terminated
    and no adversary action **do**
  $\mathbf{u}$:= u_est(GT)
  $\mathbf{M}$:= generate_moves(GT)
  choose $m \in \mathbf{M}$ maximizing exp_net_gain(GT, m)
  if exp_net_gain(GT, m) > 0
    add_simulation(GT, m)
  else
    commit(best_action($\mathbf{u}$))
    GT := subtree after best_action($\mathbf{u}$)
    add_simulation(GT, default_move(root(GT)))
  end if
end while

We initialize the game-tree to contain a simulation of a trajectory reflecting the default next move of the agent. While there is a reason to continue working with this game-tree, we update estimated expected utilities of the agent's possible next moves in the game-tree. We generate a set of possible moves (of any player) and pick the one which maximizes expected net gain from the simulation (the expected improvement in the action utility after the computation minus the computational cost). If the net gain is positive, we perform the simulation and add it to the game-tree. Otherwise, we commit to our current best action, prune the tree accordingly, and simulate the next default move.

6. HEURISTICS FOR INTELLIGENT SIMULATION

In this section, we discuss means by which one might heuristically compute expected utilities for states in the game-tree and expected information gain of trajectories not yet simulated. We discuss the difficulty of dealing with adversarial moves in real-time, and give an example of how complementary techniques can inform game-tree search.

In classical game-playing, the expected utility of reaching a leaf state is computed using a heuristic evaluation function, and the expected utility of reaching a non-leaf state (or node) is computed as a function of its immediate children. If the node has too many children to fully expand, a sampling can inform a probabilistic expectation of the utility of reaching that node. This evaluation of non-leaf states relies on the property of subtree independence, which we do not have. However, it is still the case that evaluations of the leaves of the subtree beneath a state form the primary source of information in estimating the expected utility of non-leaf states in our game-tree.

Similar or symmetric trajectories are a second source of information. Just as similarity or symmetry play a role in reducing branching factors in AI search, we may also expect similar subtrees to inform each other's estimates of expected utility, possibly reducing the amount of simulation necessary. This is most easily seen in examining an action which may be taken with different timing. For example, consider a game such as baseball. If I swing too early, I foul to one side. Too late, I foul to the other side. With minimal domain knowledge, I can use such observations (or simulations) to inform proper timing of my swing. With each enabled action, we seek to optimize timing to provide the most informational value. Information-based optimization (Neller, 1998) is particularly suited to optimization problems where the objective function is computationally expensive to compute.

Another means of informing our estimation of expected utilities is through learning. If the game involves repetitive tasks, then past experiences may inform future experiences. A person learns to balance a pole through trial and error. In practice, we may not have the luxury of such learning where one error means the game is over for good. Still this is a source of knowledge which, applied skillfully and efficiently, may significantly inform utility expectations.

Finally, through analysis of one's model, one may acquire domain knowledge useful for estimating or bounding one's utility expectations. The work of Tomlin, Lygeros, and Sastry (Tomlin *et al.*, 1998) is complementary to game-tree search in this respect. Through analysis of a nonlinear hybrid game, the state space is divided into safe and unsafe states from which the game can be guaranteed to be won or lost, respectively. While presenting computational difficulties, such an algorithm working in parallel with a game-tree search can yield better utility estimates over the run of the system. If such analysis is done a priori, we reap a large benefit in advance. Other techniques computing topological knowledge of the dynamical system or qualitatively abstracting the hybrid system model may also inform our utility expectations.

One final point involves a complication in game play which has been glossed over until now: When an adversary makes a move, it's likely that almost all of the agent's game-tree consists of trajectories which will never be traveled. Is all of an agent's deliberation nullified whenever an adversary moves? Yes and no. Yes, in that a new tree must be grown reflecting change in the real world. No, in that the previous game-tree may inform the new game-tree as discussed above. If the agent has examined trajectories where the adversary has taken the same action at a different time, then that subtree may provide much valuable information relevant to the growth of the new game-tree. Similarly, if such a game-playing control system is implemented, then over time we can expect that the model will deviate significantly from the world. The reconstruction of a revised game-tree could be similarly informed from the previous one.

With all of these sources of heuristic information, it is of course essential to underscore the point that a heuristic should always pay for its computational cost in improved performance. Finding efficient ways to draw from these and other sources of information are key to the feasibility of this approach.

7. REFERENCES

Başar, Tamar and Geert Jan Olsder (1995). *Dynamic Noncooperative Game Theory, 2nd Ed.*. Academic Press. London.

Branicky, Michael S. (1995). Studies in Hybrid Systems: modeling, analysis, and control. PhD thesis. Massachusetts Institute of Technology. Cambridge, MA, USA.

Henzinger, Thomas A. and Sastry, Shankar, Eds.) (1998). *LNCS 1386: Hybrid Systems: computation and control, First International Workshop, HSCC'98, Proceedings*. Springer. Berlin.

Neller, Todd W. (1998). Information-based optimization approaches to dynamical system safety verification. pp. 346–359. In: Henzinger and Sastry (1998).

Russell, Stuart and Eric Wefald (1991). *Do the Right Thing: studies in limited rationality.* MIT Press. Cambridge, MA, USA.

Tomlin, Claire, John Lygeros and Shankar Sastry (1998). Synthesizing controllers for nonlinear hybrid systems. pp. 360–373. In: Henzinger and Sastry (1998).