Plays as Team Plans for Coordination and Adaptation

Michael Bowling, Brett Browning, Allen Chang and Manuela Veloso

Computer Science Department, Carnegie Mellon University, Pittsburgh PA, 15213-3891, USA,
{mhb,brettb,mmv}@cs.cmu.edu, allenc@andrew.cmu.edu

Abstract. Coordinated action for a team of robots is a challenging problem, especially in dynamic, unpredictable environments. In the context of robot soccer, a complex domain with teams of robots in an adversarial setting, there is a great deal of uncertainty in the opponent's behavior and capabilities. We introduce the concept of a *play* as a team plan, which combines both reactive principles, which are the focus of traditional approaches for coordinating actions, and deliberative principles. We introduce the concept of a *playbook* as a method for seamlessly combining multiple team plans. The playbook provides a set of alternative team behaviors which form the basis for our third contribution of *play adaptation*. We describe how these concepts were concretely implemented in the CMDragons robot soccer team. We also show empirical results indicating the importance of adaptation in adversarial or other unpredictable environments.

1 Introduction

Coordination and adaptation are two of the most critical challenges for deploying teams of robots to perform useful tasks. These challenges become especially difficult in environments involving other agents, particularly adversarial ones, not under the team's control. In this paper, we examine these challenges within the context of robot soccer [6], a multi-robot goal-driven task in an adversarial dynamic environment. The presence of adversaries creates significant uncertainty for predicting the outcome of interactions particularly if the opponent's behavior and capabilities are unknown a priori, as is the case in a robot soccer competition. As such, this task encapsulates many of the issues found in realistic multi-robot settings.

Despite this unpredictability, most robot soccer approaches involve single, static, monolithic team strategies (e.g., see robot team descriptions in [1].) Although these strategies entail complex combinations of reactive and deliberative approaches, they can still perform poorly against unknown opponents or in unexpected situations. With the uncertainty present in the task, such situations are common. An alternative approach uses models of opponent behavior, constructed either before or during the competition [5], which are used then to determine the best team response. A model may be used in a reactive fashion to trigger a pre-coded static strategy, or in a deliberative fashion through the use of a planner [7]. Although these techniques have had success, they have limitations such as the requirement for an adequate representation of opponent behavior. For a completely unknown opponent team, constructing an a prior model of their strategy is impractical.

Here, we take a novel approach based on observing our own team's effectiveness rather than observing the opponent's behavior. We replace a single monolithic team

strategy, with multiple team plans that are appropriate for different opponents and situations, which we call *plays*. Each play defines a coordinated sequence of team behavior, and is explicit enough to facilitate evaluation of that play's execution. A *playbook* encapsulates the plays that a team can use. Each execution of a play from the playbook can then be evaluated and this information collected for future play selection. Successful plays, whose successes may be attributed to weaknesses in the opponent or particular strengths of our team, are selected more often, while unsuccessful plays are ignored.

2 Overview

The work described in this paper was fully implemented on our CMDragons'02 small size league (SSL) robot team. We competed with our robots at the RoboCup 2002 International competition in Fukuoka, Japan. As with other SSL teams, our small robots utilize perceptual information from an overhead color camera, and an off-field computer for centralized team processing. Hence, our approach does not yet address issues of distributed coordination per se. Due to space limitations, we do not go into the details of the larger architecture. Instead, we refer the reader to [3] and [4].

From the perspective of strategy, each robot can perform a range of individual skills. Each individual skill is encapsulated as a tactic, and all tactics are heavily parameterized to provide a wide range of behavior. The role of strategy is to assign tactics, with suitable parameters, to each robot. The robots then execute the tactic actions each and every frame. Hence, the strategy layer provides the coordination mechanism and executes one instance for the entire team and must meld individual robot skills into powerful and adaptable team behavior. Tactics can be classified as either active or non-active. An active tactic is one that attempt to manipulate the ball in some manner. Active tactics include `shoot`, `steal`, and `clear`, while example non-active tactics include `position_for_loose_ball`, `defend_line`, and `block`. Parameters are tactic specific, but example parameters often include target points, regions, or information affective behavior such as whether to aim or include deflections etc. Each is itself a complex interaction between the robot control layer that maintains robot-specific information, navigation, and motion control.

3 Play-Based Strategy

The main question addressed in this work is: "Given a set of effective and parameterized individual robot behaviors, how do we select each robot's behavior to achieve the team's goals?" This is the problem addressed by our strategy component.

3.1 Goals

The main criterion for team strategy is performance. However, a single, static, monolithic team strategy that maximizes performance is impractical. Indeed, in adversarial domains with unknown opponents, optimal static strategies are unlikely to exist. Therefore we break down the performance criteria into more achievable subgoals. The subgoals are to *(i)* Coordinates team behavior, *(ii)* Executes temporally extended sequences

of action, *(iii)* Allow for special behavior for certain circumstances, *(iv)* Allow ease of human design and augmentation, *(v)* Enable exploitation of short-lived opportunities, and *(vi)* Allow on-line adaptation to the specific opponent.

The first four goals require plays to be able to express complex, coordinated, and sequenced behavior among teammates. In addition, plays must be human readable to make strategy design and modification simple (a must at competitions!). These goals also requires a capable of executing the complex behaviors the play describes. The fifth goal requires the execution system to recognize and exploit fortuitous opportunities not explicitly described by the play. Finally, the last goal requires the system to improve its behavior over time. These goals, although critical to robot soccer, are also of general importance for coordinated agent teams in other unpredictable or adversarial environments. We have developed a play-based team strategy, using a specialized play language, to meet these goals. We describe the major components of this system, specifically play specification, execution, and adaptation, in the following sections.

3.2 Play Specification

Plays are specified using the play language, which is in an easy-to-read text format (e.g., Table 1, Plays use keywords, denoted by all capital letters, to mark different pieces of information. Each play has two components: *basic information* and *role information*. The basic information describes when a play can be executed ("`APPLICABLE`"), when execution of the play should stop ("`DONE`"), and some execution details (e.g., "`FIXEDROLES`", "`TIMEOUT`", and "`OROLE`"). The role information ("`ROLE`") describes how the play is executed, making use of the tactics described above (see Section 2). We describe these keywords below.

The `APPLICABLE` keyword denotes the state of the world under which a play can be executed. It defines the state of the world through a conjunction of high-level predicates following the keyword. Multiple keywords, on separate lines, define a logical DNF where the result of each line forms a disjunction. Examples of common predicates include `offense`, `defense`, `their_ball`, where the meaning of the predicate should be apparent. The ability to form logical DNF's means that we can choose exactly which conditions a play can be operate under.

Unlike classical planning, the level of uncertainty when running real robots makes it difficult to predict the outcome of a particular plan. Although, a play does not have effects, it does have termination conditions. Termination conditions are specified by the keyword `DONE` followed by a result (e.g., `aborted`) and a conjunctive list of high-level predicates similar to the applicability conditions. Plays may have multiple `DONE` conditions, each with a different result, and a different conjunction of predicates. Whenever *any* `DONE` condition is satisfied, the play terminates. In the example play in Table 1, the only terminating condition is if the team is no longer on offense. In this case the play's result is considered to have been `aborted`. In addition to the termination conditions, a play may be terminated by a timeout or by completing the sequence of tactics for each role. Timeouts, the length of time which can be overridden with the `TIMEOUT` keyword, are necessary to prevent the team becoming irretrievably stuck attempting an action that is not succeeding. Completions, defined by the keyword `completed`, means the play terminated correctly but did not lead to a goal score. Finally, a play is

considered to have `succeeded` or `failed` whenever a goal is scored during the play for, or against, the team. These play results form the basis for evaluating the success of the play for the purposes of adaptation.

```
PLAY Two Attackers, Corner Dribble 1
APPLICABLE offense in_their_corner
DONE aborted !offense
TIMEOUT 15

ROLE 1
  dribble_to_shoot {R {B 1100 800} ...
  shoot A
  none
ROLE 2
  block 320 900 -1
  none
ROLE 3
  position_for_pass { R {B 1000 0}...
  none
ROLE 4
  defend_line {-1400 1150} ...
  none
```

```
PLAY Two Attackers, Pass
APPLICABLE offense
DONE aborted !offense
OROLE 0 closest_to_ball

ROLE 1
  pass 3
  mark 0 from_shot
  none
ROLE 2
  block 320 900 -1
  none
ROLE 3
  position_for_pass {R {B 1000 0}...
  receive_pass
  shoot A
  none
ROLE 4
  defend_line {-1400 1150}...
  none
```

Table 1. Two example plays involving sequencing of behaviors. The left play is a special purpose play that only executes when the ball is in an offensive corner of the field.

Roles are the active component of each play, and each play has four roles corresponding to each non-goalie robot on the field. Each role contains a list of tactics with associated parameters for the robot to perform in sequence. As tactics are heavily parameterized, the range of tactics can be combined into nearly an infinite number of play possibilities. Table 1 shows an example play where the first role executes two sequenced tactics. First the robot dribbles the ball out of the corner and then switches to the shooting behavior. Meanwhile the other roles execute a single behavior for the play's duration. Sequencing implies an enforced synchronization, or coordination between roles. Once a tactic completes, all roles move to their next behavior in their sequence (if one is defined). Thus, in the example in Table 1, when the player assigned to pass the ball completes the pass, then it will switch to the mark behavior. The receiver of the pass will simultaneously switch to receive the pass, after which it will try to execute the shooting tactic.

3.3 Play Execution

The play execution module is responsible for instantiating the active play into actual robot behavior. Instantiation consists of many key decisions: role assignment, role switching, sequencing tactics, opportunistic behavior, and termination. Role assignment is dynamic, rather than being fixed, and is determined by uses tactic-specific methods. To prevent conflicts, assignment is prioritized by the order in which roles appear. Thus, the first role, which usually involves ball manipulation, is assigned first and considers all four field robots. The next role is assigned to one of the remaining robots, and so on. The prioritization provides the execution system the knowledge to select the best

robots to perform each role and also provides the basis for role switching. Role switching is a very effective technique for exploiting changes in the environment that alter the effectiveness of robots fulfilling roles. The executor continuously reexamines the role assignment for possible opportunities to improve it as the environment changes. Although, it has a strong bias toward maintaining the current assignment to avoid oscillation.

Sequencing is needed to move the entire team through the list of tactics in sequence. When the tactic executed by the *active player*, the robot whose role specifies a tactic related to the ball, succeeds then the play transitions *each* role to the next tactic in their relative sequence. Finally, opportunistic behavior accounts for unexpected fortuitous situations where a very basic action would have a valuable outcome ie. when an opportunity to shoot directly on goal presents itself. Thus, opportunistic behavior enables plays to have behavior beyond that specified explicitly. As a result, a play can encode a long sequence of complex behavior without encumbering its ability to respond to unexpected short-lived opportunities. Finally, the play executor checks the play's termination criteria, the completion status of the tactics, and the incoming information from the referee to determine if the play has completed, and with what result.

3.4 Play Selection

The final facet of the playbook strategy system is the mechanism for play selection and adaptation of play selections given experience. Our basic selection scheme uses the applicability conditions for each play to form a candidate list from which one play is selected at random. To adapt play selection, we modify the probability of selecting a play using a weighting scheme. We describe this mechanism, along with experimental results, in more detail below.

4 Playbook Adaptation

Playbook adaptation is the problem of adapting play selection based on past execution to find the dominant play, or plays, for the given opponent and the history of execution. In order to facilitate the compiling of past outcomes into the selection process, we associate with each play a weight, $w_{p_i} \in [0, \infty)$. For a given set of applicable plays, A, the weights are normalized to define a probability mass distribution for selecting each play as,

$$Pr(\text{selecting } p_i) = \frac{w_{p_i}}{\sum_{p_j \in A} w_{p_j}}.$$

Playbook adaptation involves adjusting the selection weights given the outcome of a play's execution. An adaptation rule is a mapping, $W(\mathbf{w}, p_i, o) \rightarrow [0, \infty)$, from a weight vector, a selected play, and its outcome, to a new weight for that play. These new weights are then used to select the next play.

There are a number of obvious, desirable properties for an adaptation rule. All things being equal, more successes or completions should increase the play's weight. Similarly, aborts and failures should decrease the weight. In order for adaptation to have any effect, it also must change weights drastically to make an impact within the short

time-span of a single game. This leads us to the basic rule that we implemented for the RoboCup 2002 competition uses a weight multiplication rule, where each outcome multiplies the play's previous weight by a constant. Specifically, the rule is,

$$W(\mathbf{w}, p_i, o) = C_o w_{p_i},$$

With C_o fixed to $C_{\texttt{succeeded}} = 4$, $C_{\texttt{completed}} = 4/3$, $C_{\texttt{aborted}} = 3/4$, $C_{\texttt{failed}} = 1/4$.

4.1 Evaluation

Our strategy system was used effectively during RoboCup 2002 against a wide range of opponents with vastly different strategies and capabilities. Throughout, we observed that our team quickly honed in on the plays that worked within a few minutes. We predominantly began each game with a uniform prior. That is, with $w_i = 1$ for each play i. As the competition progressed, we developed more capable plays, in a reasonably efficient manner helped by the readability of the play language (our final playbook contained around 20 plays, including specialized plays for penalties, free kicks etc). Although we have no specific results, anecdotally, adaptation appears to make the team work as good as the performance of the best play given the underlying tactics. Thus, in situations where the tactics cannot perform their assigned objective, say when playing a very good team, play adaptation does not improve the performance of the team. However, it does not hinder performance either.

In order to more scientifically understand the capabilities and limitations of the play approach, we constructed a number of simplified scenarios to evaluate adaptation performance. These scenarios compare whether multiple plays are actually necessary, and also examine the usefulness of playbook adaptation. We compared four simple offensive plays paired against three defensive tactics. Only two offensive robots were used against one defensive robot, where the offensive plays consist of the various combinations of `shoot` with and without aiming for the active role, and `position_for_rebound` or `screen` for the supporting role. The defensive robot executed a single tactic, which was one of `block`, `active_def`, or `brick` where the robot did not move. In all cases, the robots start in the usual "kick off" position in the center of the field. For each scenario 750 trials were performed in our UberSim SSL simulator [2]. A trial was considered a success if the offense scored a goal within a 20s time limit.

Table 2 shows the play comparison results. Each trial is independent, and so the maximum likelihood estimate of each play's success probability is the ratio of successes to trials. Note that there is no static strategy that is optimal against every possible opponent even in this simplified scenario. Our results support the notion that play-based strategies are capable of exhibiting many different behaviors with varying degrees of effectiveness. For instance, the screen plays, one of which was shown in the example trace, are effective against an "active" defender which tries to steal the ball from the offense, because the non-shooting attacker is capable of setting screens for the non-shooting attacker. On the other hand, the screen plays are less effective against a "blocking" defender which guards the goal.

To explore playbook adaptation we use a playbook containing all four offensive plays against a fixed defender running either `block` or `active_def`. We initially used

Table 2. Play comparison results. For each scenario, the percentage of successes for the 750 trials is shown. The boldfaced number corresponds to the play with the highest percentage of success for each defensive behavior.

Play tactic 1	tactic 2	`block`	`active_def`	`brick`
shoot(aim)	position_for_rebound	**72.3%**	49.7%	**99.5%**
shoot($no\ aim$)	position_for_rebound	66.7%	57.3%	43.1%
shoot(aim)	screen	40.8%	59.0%	92.4%
shoot($no\ aim$)	screen	49.2%	**66.0%**	72.0%

the algorithm described above, but discovered an imperfection in the approach. Due to the strength of the reinforcement for a completed play, it is possible for a moderately successful but non-dominant play to quickly gain reward and dominate selection. This phenomenon did not occur in competition due to the larger disparity in plays against a given opponent and lower success probabilities. The issue is a lack of normalization in the weight adjustment to account for play selection probabilities. Therefore, we included a normalization factor in the weight updates. Specifically, we used the following rule,

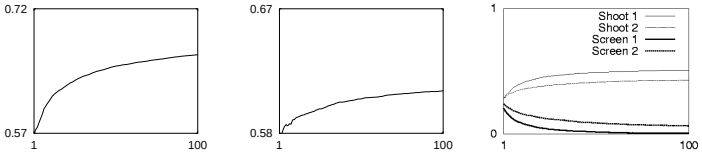
$$W(\mathbf{w}, p_i, o) = \begin{cases} w_{p_i} 2/\Pr(p_i) \text{ if } o = \texttt{Succeeded} \\ w_{p_i} \Pr(p_i)/2 \text{ if } o = \texttt{Failed} \end{cases},$$

where $\Pr(p_i)$ is the probability assigned to p_i according to $\mathbf{w}$.

To evaluate the performance of the algorithm, we compare the expected success rate (ESR) of using this adaptation rule against a fixed defensive behavior. We used the results in Table 2 to simulate the outcomes of the various play combinations. All the weights are initialized to 1. Figure 1(a) and (b) show the ESR for play adaptation over 100 trials, which is comparable to the length of a competition (approximately 20 minutes). The lower bound on the y-axis corresponds to the ESR of randomly selecting plays and the upper bound corresponds to the ESR of the playbook's best play for the particular defense. Figure 1(c) shows the probabilities of selecting each play over time when running the adaptation algorithm.

As graphs (a) and (b) indicate in Figure 1, against each defense the overall success rate of the offense quickly grows towards the optimal success rate within a small number of trials. Likewise graph (c) shows that against the `block` defense, the probability of selecting either of two plays with comparatively high individual success rates quickly dominates the probability of selecting the two less successful plays. Clearly, the algorithm very quickly favors the more successful plays.

These results, combined with the RoboCup performances, demonstrate that adaptation can be a powerful tool for identifying successful plays against unknown opponents. Note the contrast here between the use of adaptation to more common machine learning approaches. We are not interested in convergence to an optimal control policy. Rather, given the short time limit of a game, we desire adaptation that achieves good results quickly enough to impact the game. Hence a fast, but non-optimal response is desired over a more optimal but longer acting approach.



(a) (b) (c)

Fig. 1. (a), (b) show ESR against block and active_def, (c) shows expected play success probabilities against block. These results have all been averaged over 50000 runs of 100 trials.

5 Conclusion

We have introduced a novel team strategy engine based on the concept of a play as a team plan, which can be easily defined by a play language. Multiple, distinct plays can be collected into a playbook where mechanisms for adapting play selection can enable the system to improve the team response to an opponent without prior knowledge of the opponent. The system was fully implemented for our CMDragons robot soccer system and tested at RoboCup 2002, and in the controlled experiments reported here. Possible future directions of research include extending the presented play language, enhancing the play adaptation algorithm.

6 Acknowledgements

This research was sponsored by Grants Nos. DABT63-99-1-0013, F30602-98-2-013 and F30602-97-2-020. The information in this publication does not necessarily reflect the position of the funding agencies and no official endorsement should be inferred.

References

1. Andreas Birk, Silvia Coradeschi, and Satoshi Tadokoro, editors. *RoboCup 2001: Robot Soccer World Cup V*. Springer Verlag, Berlin, 2002.
2. Brett Browning and Erick Tryzelaar. Ubersim: A multi-robot simulator for robot soccer. In *Proceedings of AAMAS*, 2003.
3. James Bruce, Michael Bolwing, Brett Browning, and Manuela Veloso. Multi-robot team response to a multi-robot opponent team. In *ICRA Workshop on Multi-Robot Systems*, 2002.
4. James Bruce and Manuela Veloso. Real-time randomized path planning for robot navigation. In *Proceedings of IROS-2002*, pages 2383–2388, Switzerland, October 2002.
5. S.S. Intille and A.F. Bobick. A framework for recognizing multi-agent action from visual evidence. In *AAAI-99*, pages 518–525. AAAI Press, 1999.
6. Itsuki Noda, Shóji Suzuki, Hitoshi Matsubara, Minoru Asada, and Hiroaki Kitano. RoboCup-97: The first robot world cup soccer games and conferences. *AI Magazine*, 19(3):49–59, Fall 1998.
7. Patrick Riley and Manuela Veloso. Planning for distributed execution through use of probabilistic opponent models. In *ICAPS-02, Best Paper Award*, Toulouse, France, April 2002.