Distributed, Play-Based Role Assignment for Robot Teams in Dynamic Environments

Colin McMillen and Manuela Veloso

School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, U.S.A.
{mcmillen,veloso}@cs.cmu.edu

Summary. The design of a coordination strategy for a distributed robotic team is challenging in domains with high uncertainty and dynamic environments. We present a distributed, play-based role assignment algorithm that has been implemented on real robots in the RoboCup four-legged league. The algorithm allows the robots to adapt their strategy based on the current state of the environment, the game, and the behavior of opponents. The distributed play-based approach also enables the robots to reason about task-based temporal constraints and has been designed to be resistant to the problem of role oscillation.

1 Introduction

A common goal of distributed autonomous robotic systems is the development of teamwork and coordination strategies. The benefits of adding multiple robots to a system, such as increased performance and reliability, have been demonstrated in many different situations. However, depending on the domain and the task, different sorts of approaches might be needed. We are interested in the implementation of multi-robot coordination in domains with high uncertainty and dynamic environments.

In this paper, we present our approach to *role assignment* in the RoboCup four-legged league [2], in which two teams of four Sony AIBO robots compete in a robot soccer game. Figure 1 shows a snapshot of a recent AIBO game. This domain presents many challenges, including: full robot autonomy, distributed robot team control, limited individual robot perception, the presence of robot adversaries, task-dependent temporal constraints, and high communication latency. In this paper, we contribute a new distributed *play-based* system that equips the robots with *plays* – alternative teamwork strategies. This method was developed to overcome limitations of previous approaches. In particular, it assigns roles to robots in a fault-tolerant manner that minimizes role switching and synchronization problems. Our approach has been fully implemented within robot soccer, but is designed to be relevant to general multi-robot domains that share some of the challenging features of robot soccer, as identified in this paper.

Fig. 1. A RoboCup four-legged league soccer match.

We discuss the RoboCup legged league in section 2, identifying the technical features that we address as especially challenging for multi-robot coordination. Section 3 introduces our approach to teamwork. Section 4 presents experimental results that highlight some of the advantages of our approach. We present our conclusions in section 5. Related work is discussed throughout the paper as needed.

2 Challenges of the RoboCup Legged League

In the RoboCup four-legged league, two teams of four Sony AIBO robots play each other in a time-limited and space-limited setting (currently two game halves of ten minutes each and a field of 6m×4m). The settings and the rules of the game change every year to create new research challenges. The current complete rules of the domain are available at [2]. We focus on the discussion of the general features of the game that are of relevance to team coordination. These features include:

1. **Full autonomy:** each team of robots operates completely without human supervision. However, teams are allowed to change the robots' programming at halftime or during a timeout. Each team is granted one timeout per game.
2. **Distributed teams:** all perception, computation, and action is done onboard the robots. The robots are equipped with 802.11b wireless networking, which enables communication among team members; however, the robots are not allowed to communicate with any off-board computers.

3. **Limited perception:** each robot's primary sensor is a low-resolution camera with a very narrow field of view (under 60 degrees). A single robot therefore has a very limited view of the world, so teams can benefit greatly from communication strategies that build a shared world model.
4. **Dynamic, adversarial environment:** the presence of adversaries in the environment is a significant challenge. Opponents ensure that the environment is extremely dynamic: within a few seconds, the state of the world may change significantly.
5. **Temporal constraints:** there are two temporal constraints that arise due to the presence of adversaries. First, all team decisions must occur in real time. A team that takes too long to coordinate will have robots that display hesitation in carrying out their tasks, which gives the opponents a significant advantage. Second, soccer is a finite-horizon zero-sum game. A game of soccer has a winning team, a losing team, and a defined ending point. Playing a conservative strategy – which might work well over a long period of time – is of no use to a team that is losing and only has a few seconds remaining in the game. A team in this situation must choose a strategy that can score a goal quickly, even if such a strategy has other weaknesses. Multiple team coordination strategies are needed. The selection algorithm faces complex multi-objective optimization criteria.
6. **High network latency:** the presence of dozens of robots in the competition environment leads to very unpredictable quality of the robots' wireless network. Teams may experience periods of high network latency and collisions; latencies of over a second have been observed. To achieve consistent performance, a team needs to ensure that the coordination strategies employed are robust to disruptions in communication.

The issues of limited perception in the four-legged league have already been addressed in previous work. Our specific implementation of a shared world model [7] is not discussed at length here, but it is important to note that some of the decisions made by individual robots on our team are influenced by information that has been communicated by teammates.

In this paper, we focus on the challenges of role assignment (also known as *task allocation*) in the RoboCup legged league. Due to the constraints of our domain, it is a requirement that the solution be implemented on fully autonomous, distributed robots. The work presented in this paper specifically addresses the issues of adversarial environments, temporal constraints, and robustness under high network latency.

Gerkey and Mataric [4] discuss role assignment in RoboCup, giving an overview of the strategies used by teams in each of the RoboCup leagues. In the four-legged league, the predominant approach involves allocating three roles: an *attacker*, a *defender*, and some sort of *supporter*. According to Gerkey's survey, nearly all RoboCup teams assign roles to robots in a greedy fashion. For example, in previous years, our own team (CMPack) assigned the roles in a fixed order using a well-defined objective function, namely first the *attacker* role to the robot that could reach the ball most quickly, then the *defender* role to whichever of the other two robots was closest to our own goal, and finally the *supporter* role to the remaining robot [8]. To prevent robots from interfering with one another, the attacker was the only robot allowed to actually approach the ball for a kick; the other robots would position themselves in useful supporting locations. If the ball came near to another robot, the team members would negotiate a role switch. After the role switch, the closest robot to the ball would become the new attacker, and the attack would continue. This was a very effective strategy that contributed strongly to our world championship in 2002. However, this strategy has some limitations, particularly in terms of role switching and oscillation. Two robots that are equally suited to the attacker role might fight over it. The potential outcome is an undesirable role oscillation between the robots – a period of hesitation where neither robot makes significant progress toward completing the task. A standard solution to this problem, which has previously been implemented in our own team and other RoboCup teams, is to add some hysteresis to the role assignment: either allowing role assignments to occur infrequently (e.g., every few seconds) or ensuring that a role switch will not occur unless one robot is significantly more suited to the task than another. However, adding hysteresis to a system can be difficult: if too much hysteresis is added, the robots will miss out on opportunities because they are no longer reacting as quickly to changes in their dynamic environment. In practice, finding a solution that balances these two constraints (lack of role oscillation and immediate response to environmental changes) can be difficult. This problem is exacerbated under the presence of communication failures or high network latency, where the negotiation of a role switch may take several seconds to complete. Another limitation of this design is that there can be a significant cost to role switching, as the robots reconfigure themselves for their new roles. The work presented in this paper attempts to minimize the effect of these limitations.

Dylla *et al.* [3] propose a soccer strategy language that formalizes the strategies and tactics used by human soccer teams. Their goal is to be able to specify soccer strategies in an abstract way that does not depend strongly on the specific robot hardware used in the competition. However, to our knowledge, the authors do not yet plan to use this language in the implementation of any real robot soccer team. In this paper, we present a strategy language that could be used in any multi-robot system and that has been implemented in the four-legged league of the 2005 RoboCup competition.

3 Distributed Play-Based Role Assignment

We can say that teamwork in general consists of a team control policy, i.e., a selection of a joint action by teammates given a perceived state of the environment [6]. It is our experience that it is rather challenging to generate or learn a team control policy in complex, highly dynamic (in particular adversarial), multi-robot domains. Therefore, instead of approaching teamwork in terms of a mapping between state and joint actions, we follow a *play-based* approach, as introduced by Bowling *et al.* [1]. We introduce a distributed play-based approach to teamwork, which allows us to handle the domain challenges introduced in section 2. A play specifies a *plan* for the team; i.e., under some applicability conditions, a play provides a sequence of steps for the team to execute. Multiple plays can capture different teamwork strategies, as explicit responses to different types of opponents. Bowling showed that play selection weights could be adapted to match an opponent. Plays also allow the team to reason about the zero-sum, finite-horizon aspects of a game-playing domain: the team can change plays as a function of the score and time left in the game. Our play-based teamwork approach ensures that robots do not suffer from hesitation nor oscillation, and that team performance is not significantly degraded by possible periods of high network latency. We believe that ours is the first implemented distributed play-based teamwork approach, at least in the context of the RoboCup four-legged league.

3.1 Plays

A *play* is a team plan that provides a set of roles, which are assigned to the robots upon initiation of the play. Bowling [1] introduced a play-based method for team coordination in the RoboCup small-size league. However, the small-size league has centralized control of the robots. One of the significant contributions of our work is the development of a play system that works in a distributed team. The play language described by Bowling assumes that the number of robots is fixed, and therefore always provides exactly four different roles for the robots. In another extension to Bowling's work, our plays also specify which roles are to be used if the team loses some number of robots due to penalties or crashes. This extension to the role-assignment aspects of Bowling's play language allows the team to robustly adapt to the loss of team members without the need for additional communication. This is a particularly important extension for domains where limited or high-latency communication is the norm.

Our play language itself is also strongly inspired by the work of Bowling. Our language allows us to define *applicability conditions*, which denote when a play is suitable for execution; what *roles* should be assigned when we have a specific number of active robots on the team; and a *weight*, which is used to decide which play to run when multiple plays are applicable.

Applicability. An applicability condition denotes when a play is suitable for execution. Each applicability condition is a conjunction of binary predicates. A play may specify multiple applicability conditions; in this case, the play is considered executable if any of the separate applicability conditions are satisfied.

Roles. Each play specifies which roles should be assigned to a team with a variable number of robots by defining different `ROLES` directives. A directive applies when a team has k active robots, and specifies the corresponding k roles to be assigned. If a robot team has n members, each play has a maximum of n `ROLES` directives. Since our AIBO teams are composed of four robots, our plays have four `ROLES` directives.

Weight. Weight is used to decide which play to run when multiple plays are applicable. In our current implementation, the play selector always chooses the applicable play with greatest weight. Future work could include choosing plays probabilistically based on the weight values or updating the weights at execution time to automatically improve team performance. *Playbook adaptation* of this sort has been implemented by Bowling for the small-size league [1]. In the work presented in this paper, adaptation was not used – the weights were set manually.

Unlike the work of Bowling, we do not have `DONE` or `TIMEOUT` keywords that specify when a play is complete. Rather, a play is considered to be complete as soon as the play selector chooses a different play, which may happen because the current play is no longer applicable or because another play with greater weight has recently become applicable. Each predicate used in an applicability condition is designed with some hysteresis, such that it is not possible for the predicate to rapidly oscillate between true and false. The predicates used in our approach depend on features of the environment – such as the time left in game, the number of goals scored by each team, and the number of robots available to each team – that by their nature cannot rapidly oscillate. This ensures that the play choice also cannot rapidly oscillate.

Figure 2 shows an example of a defensive play. Its applicability conditions specify that this play is applicable 1) when our team has fewer active players than the opponents or 2) when the game is in the second half and our team is winning by at least two points. If we have only one active robot on our team, we will assign it the Goalkeeper role; if we have two robots, one is assigned the Goalkeeper role and the other is assigned the Defender role; and so on.

```
PLAY StrongDefense
APPLICABLE fewerPlayers
APPLICABLE secondHalf winningBy2OrMoreGoals
ROLES 1 Goalkeeper
ROLES 2 Goalkeeper Defender
ROLES 3 Goalkeeper Defender Independent
ROLES 4 Goalkeeper Defender Midfielder Independent
WEIGHT 3
```

Fig. 2. An example play with multiple applicability conditions.

3.2 Play Selector

The *play selector* runs continuously, on one robot that is arbitrarily chosen to be the leader. The play selector chooses which play the team should be running. The leader periodically broadcasts the current play (and role assignments) to its teammates. Distributed play-based coordination is achieved through a predefined agreement among the team members to resort to a *default play* if a robot doesn't hear a play broadcast within a *communication time limit*. A failure of the leader or a network problem may trigger this default coordination plan. The algorithm used by the play selector is presented in Figure 3.

```
SELECT_PLAY(S: world state, P: playbook, D: default play):
  BEST_PLAY <- D
  BEST_WEIGHT <- WEIGHT(D)
  for each PLAY in P:
    if WEIGHT(PLAY) > BEST_WEIGHT:
      for each CONDITIONS in APPLICABLE(PLAY):
        if all CONDITIONS are satisfied in STATE:
          BEST_PLAY <- PLAY
          BEST_WEIGHT <- WEIGHT(PLAY)
  return BEST_PLAY
```

Fig. 3. Algorithm used by the play selector.

3.3 Roles

The *role* assigned to each robot determines what behaviors the robot actually runs. Our approach, used in RoboCup 2005, is unique in that it is *region-based*: each robot is assigned to a region of the field. A robot is primarily responsible for going after the ball whenever the ball is in that robot's region. When the ball is not in its region, the robot will position itself at a good position within its region (defined by a role-dependent objective function). Unlike previous approaches, robots no longer need to negotiate with one another in order to gain the *attacker* role that allows them to approach the ball. In this way, the performance of the team does not degrade significantly under high network latency. To ensure that one or more robots are always chasing after the ball, these regions typically overlap significantly. We have developed algorithms that prevent the robots from interfering with one another even when they are playing in overlapping regions. To provide robustness against communication failure, these algorithms are designed to operate without the need for communication, using local information such as a robot's vision of its own teammates. However, if communication is available, we can use additional features (such as reported teammate positions and ball positions) that provide additional confidence that our robots will not interfere with one another.

3.4 Role Allocation

The selection of a play determines which roles need to be allocated to the robots. However, it does not specify which robots should be assigned to each role. Therefore, a role allocation algorithm is still needed to assign the roles. This algorithm also runs on the leader robot, which broadcasts the assignment along with the selected play. Our role allocator has two features that differentiate it from those used by most other RoboCup legged-league teams [4]. First, it only runs when a play is initially selected, as opposed to continuously. Second, it allocates roles in a *role-preserving* manner – minimizing role switching. Formally, if a new play P_t is selected at time t, and P_t specifies n roles $\{R_{1..n}\}$ for the n robots $r_{1..n}$, and r_i was already assigned to R_j in P_{t-1}, r_i is guaranteed to still be assigned to R_j in P_t. (Any remaining roles can be allocated in a greedy fashion.) The region-based nature of our plays enables this feature, as r_i is already guaranteed to be in the region for R_j since it was already in that role's region in the previous play. Therefore, it can assume the new role without any transitional cost. These features provide additional resistance to oscillation in cases in which two plays share common roles.

4 Experimental Results

We have previously presented empirical results that support the feasibility and effectiveness of multiple plays in the RoboCup four-legged league [5]. In this paper, we contribute a role allocation strategy, claiming that it addresses hesitation due to role oscillation by preserving a robot's role when possible. We show experimental evidence that supports this particular claim.

In each experimental trial, three robots work together in a robot soccer task, namely *ball advancement* – moving the ball towards the opposing goal as quickly as possible. Figure 4 shows the initial position of the robots, from which the team advances the ball down the field towards the goal. A trial is considered complete when either a goal is scored, the ball advances past the opponents' back line, or the ball hits one of the goal posts. The time of each completed trial is measured.

We test the robots' teamwork in three different team play configurations: (i) a single *Defender-Striker-Independent* play; (ii) a single *Defender-Midfielder-Independent* play; and (iii) switching every five seconds between the two plays in (i) and (ii). Since these two plays share two roles (defender and independent), we expect that, even with frequent play switching, our role assignment algorithm will not adversely affect the performance of the team.

Each configuration was tested for 40 completed trials, for a total of 120 experiments. Figure 5 summarizes the results. The fastest and slowest times achieved in any trial were 17.18 and 70.15 seconds, respectively. The *Defender-Striker-Independent* play performs best at this task, completing each trial in a mean time of 31.06 seconds. The *Defender-Midfielder-Independent* play

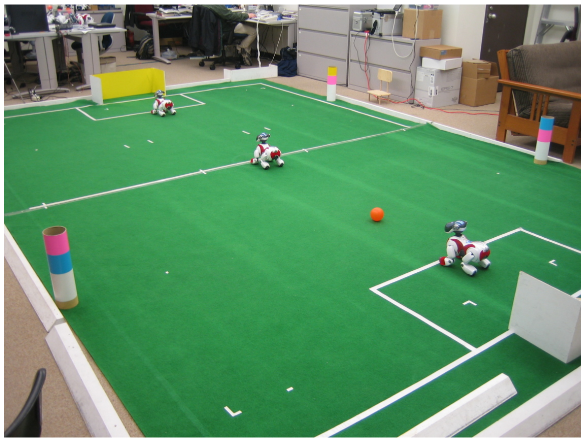

Fig. 4. Initial position for each experimental trial. The three robots are placed in three positions on the field, with the ball in the defense area. The experiment proceeds until the robots advance the ball past the end line of the opposite half of the field.

performs more slowly, completing each trial in a mean time of 35.05 seconds. The difference between these times is significant (determined by Student's two-tailed t-test, with $p = 0.048$). When the robots oscillate between these two plays, their performance remains good, with the mean time of the switching case (33.29 seconds) between the mean times of the other two cases. Since the play-switching case still performs better than the worse of the two plays, we note that there is no significant detrimental effect on performance.

5 Conclusion

In this paper, we have presented the details of a distributed play-based role assignment algorithm, which has been implemented on a distributed team of robots for the RoboCup four-legged league. The algorithm aims to solve several important general distributed multi-robot challenges, including the presence of adversaries, task-based temporal constraints, and robustness to network failure. We have presented experimental results that show that our role-preserving assignment algorithm allows a team to perform well even when plays are rapidly changed.

The presented role-assignment algorithm and plays have been tested in the RoboCup 2005 competition. Our team came in fourth place in a challenging competition of twenty-four teams. Our team typically rotated through three well-balanced plays in the first minutes of each game, which allowed us to see the performance of each play against the specific opponent. As a form of adjustable autonomy, we could manually change the team's strategy at halftime or during a timeout. Our future work includes the investigation

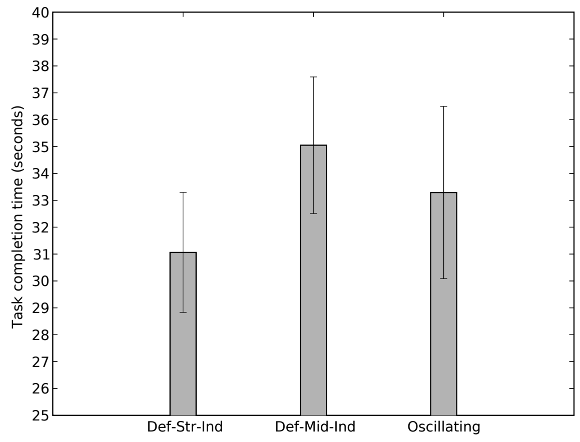

Fig. 5. Experimental results for the *Defender-Striker-Independent* play, *Defender-Midfielder-Independent* play, and switching between the two plays. The figure shows the means and 90% confidence intervals for each case.

of automatic play adaptation in distributed domains and recognition of the opponents' state and strategy.

References

1. M. Bowling, B. Browning, A. Chang, and M. Veloso. Plays as team plans for coordination and adaptation. In D. Polani, B. Browning, A. Bonarini, and K. Yoshida, editors, *RoboCup 2003: Robot Soccer World Cup VII*. 2004.
2. RoboCup Tech. Committee. Four legged robot football league rule book, 2006.
3. F. Dylla, A. Ferrein, G. Lakemeyer, J. Murray, O. Obst, T. Röfer, F. Stolzenburg, U. Visser, and T. Wagner. Towards a league-independent qualitative soccer theory for RoboCup. In *RoboCup 2004: Robot Soccer World Cup VIII*. 2005.
4. B. Gerkey and M. Mataric. On role allocation in RoboCup. In D. Polani, B. Browning, A. Bonarini, and K. Yoshida, editors, *RoboCup 2003: Robot Soccer World Cup VII*. 2004.
5. C. McMillen, P. Rybski, and M. Veloso. Levels of multi-robot coordination for dynamic environments. In *Multi-Robot Systems: From Swarms to Intelligent Automata, Volume III*. 2005.
6. D. Pynadath and M. Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of AI Research*, 2002.
7. M. Roth, D. Vail, and M. Veloso. A real-time world model for multi-robot teams with high-latency communication. In *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2003.
8. D. Vail and M. Veloso. Dynamic multi-robot coordination. In *Multi-Robot Systems: From Swarms to Intelligent Automata, Volume II*. 2003.