Workshop 9

Agents in Real-Time and Dynamic Environments

Editors:
Wiebe van der Hoek
Gal A. Kaminka
Yves Lespérance
Michael Luck
Sandip Sen
Proceedings of the AAMAS 2010 Workshop on Agents in Real-time and Dynamic Environments

Ubbo Visser
Sahar Asadi
Tim Laue
N. Michael Mayer
(editors)

Held on May 10th, 2010 in conjunction with the ninth International Conference on Autonomous Agents and Multi Agent Systems, Toronto, Canada
Recent developments in Multi Agent systems (MAS) have been promising in achieving autonomous, collaborative behaviour between agents in various environments. The progress can be seen in competitions that serve as benchmarks in the respective fields (e.g. RoboCup, DARPA challenge). Still, most of the approaches, both in software simulations and real world robots, have problems if the environment is dynamic and the agents have to act in real time. Examples are obstacle avoidance with moving obstacles or world models which are composed from egocentric views of numerous agents. Another aspect is the need for quick responses. In an environment where a number of agents build a team and both single agent decisions and team collaborative decisions have to be made methods have to be fast and precise. This workshop addresses various problems that occur with respect to these issues.

The main focus of this workshop will be methods from various areas such as world modeling, planning, learning, and communicating with agents in real-time and dynamic environments. Within this general framework we particularly aim to bring together researchers to discuss the following topics: World modeling (quantitative, qualitative), coaching (one agent gives advice to a group of agents), planning with resources (especially time), cooperation between agents (robot and/or humans), communication between agents (implicit, non-verbal, or verbal one), real-time systems software issues (often ignored but important if serious about real-time issues in robotics), scalability and robotics interfacing issues (demands a great deal of support from the initial design of the system).

We selected eight contributions that fitted best to the subject of the workshop. The contributions focused on team coordination, joint exploration, recognition of intentions. In addition, we also had contributions that show up very interesting new directions which may turn into "hot topics" in the following years: "Analyzing the Influence and Effectiveness of One-man Agent on its Teams Performance" by Kawarabayashi and Kubo and "Avatars in a Modern Soccer Manager" by Visser and Lisetti are among those. Kawarabayashi and Kubo analyze if a single agent shares a common goal without sharing the common strategy how this affects the performance of the whole team. The authors get a better result in some cases if a singleton agent is deployed. The second paper introduces an approach to realistically simulated national league and international soccer games based on player profiles and in this way come to realistic results.

The work titled "Towards An Integrated Approach of Real-time Coordination for Multi-Agent Systems" by G. Mahdi et al. complements this section and gives an outlook on possible further developments. The cooperative method for agents behavior in performing a specified task is represented by "Selection of Rendezvous Points for Multi-Robot Exploration in Dynamic Environments" by J. de Hoog and "Decentralised Coordination of Unmanned Aerial Vehicles for Target Search using the Max Sum Algorithm"
by D. F. Francesco Maria et al.. In the first contribution, an advanced algorithm is presented that selects rendezvous-points for multi-robot exploration in static environment under limited communication condition. The second paper complements the subject by discussing strategies for flying agents and their coordination.

Another important field is the recognition of intentions represented by Felipe and Veloso with the contribution "Learning Opponent’s Strategies in the Small-Size League of RoboCup", which gives a concise overview over recent efforts in this field. A more technical approach follow the two remaining contributions that are "Hypotheses based Multi-Object Tracking in the RoboCup MidSize League" by Janssen and Molengraft and "Human-Guided Real-Time Multi-Agent Coordination in Dynamic Uncertain Domains" by M. Rajiv et al..
Program Committee and Reviewers

We are grateful to the following members of the international program committee for helping us to make this workshop:

- Luis Almeida, Porto, Portugal
- Minoru Asada, Osaka, Japan
- Sahar Asadi, Oerebro, Sweden
- Tucker Balch, Atlanta, USA
- Xiaoping Chen, Beijing, China
- Amy Eguchi, New York, USA
- Udo Frese, Bremen, Germany
- Uli Furbach, Koblenz, Germany
- Adam Jacoff, Gaithersburg, USA
- Todd Hester, Austin, USA
- Gerhard Kraetzschmar, Bonn, Germany
- Gerhard Lakemeyer, Aachen, Germany
- Tim Laue, Bremen, Germany
- Pedro Lima, Lisboa, Portugal
- N. Michael Mayer, Chia-Yi, Taiwan
- Daniele Nardi, Rome, Italy
- Itsuki Noda, Tsukuba, Japan
- Daniel Polani, Hatfield, UK
- Martin Riedmiller, Freiburg, Germany
- Peter Stone, Austin, USA
- Satoshi Tadokoro, Sendai, Japan
- Manuela Veloso, Pittsburgh, USA
- Ubbo Visser, Miami, USA
- Stefan Zickler, Pittsburgh, USA
Contents

6   Analyzing the Influence and Effectiveness of One-man Agent on its Teams Performance .........................................................
    Tomomi KAWARABAYASHI-KUBO and Tatsuya YAMADA

12  Avatars in a Modern Soccer Manager ........................................
    Ubbo Visser and Christine Lisetti

20  Towards An Integrated Approach of Real-time Coordination for Multi-Agent Systems .................................................................
    Ghulam Mahdi, Abdelkader Gouaich, and Fabien Michel

25  Selection of Rendezvous Points for Multi-Robot Exploration in Dynamic Environments ..........................................................
    Julian de Hoog, Stephen Cameron and Arnoud Visser

35  Decentralised Coordination of Unmanned Aerial Vehicles for Target Search using the Max Sum Algorithm ........................................
    Delle Fave Francesco Maria, Zhe Xu, Alex Rogers, and Nick Jennings

45  Learning Opponent’s Strategies in the Small-Size League of RoboCup .................................................................
    Trevizan Felipe and Manuela Veloso

53  Hypotheses based Multi-Object Tracking in the RoboCup MidSize League .............................................................
    Rob Janssen and Ren van de Molengraft

55  Human-Guided Real-Time Multi-Agent Coordination in Dynamic Uncertain Domains ..............................................................
    Maheswaran Rajiv, Craig Rogers, Romeo Sanchez, and Pedro Szekely
Analyzing the Influence and Effectiveness of One-man Agent on its Team’s Performance – on the RoboCup Soccer Simulation 2D –

Tomomi KAWARABAYASHI-KUBO
Fukui National College of Technology
Geshi, Sabae City, Fukui Pref.
916-8507, Japan
tomomi@fukui-nct.ac.jp

Tatsuya YAMADA
Fukui National College of Technology
Geshi, Sabae City, Fukui Pref.
916-8507, Japan
tatsuya.wa@gmail.com

ABSTRACT
The purpose of our study is to examine the influence and effectiveness of a one-man agent on its simulated soccer team based on the coordination, from the standpoint of its team’s performance.

The one-man agent is defined as an agent which behaves by itself without others’ cooperation while the agent shares the goal of the team – to win against an opponent team. Compared to the base team, the average scores of two experimental teams, which has the one-man agent of a center forward and a right forward respectively, were increased from 0.04 to 0.48 and from 0.04 to 0.50, respectively. Additionally, 67% and 96% of average scores of these teams were scored by the one-man agent, respectively. The results suggest the one-man agent may contribute to the improvement in its team’s performance.

Categories and Subject Descriptors
I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence—Multiagent systems

General Terms
Performance, Experimentation, Verification

Keywords
Agent Cooperation::Teamwork, coalition formation, coordination

1. INTRODUCTION
Many researchers on the RoboCup[1,2] Soccer Agents has been conducted as a study of multi-agent system(MAS). In these studies, to achieve the goal as a soccer team, these simulated soccer teams are made up based on the premise of cooperative relation by a formation and a position of each agent[3, 4]. Furthermore, there have been a lot of researches on how each agent collaborates with its team mates to make each agent accomplish the team’s goal[5, 6, 7], because generally it should have a big advantage that the agent behaves cooperatively with the team mates by making a use of the cooperative relation. Realizing a team play(an organized play) as mentioned above is occupied an important place in real human soccer.

However, getting back to completing the team’s goal – to win against an opponent team –, it is not always better for the team that an agent behaves with the team mates cooperatively. It is also concerned that the agent which behaves by itself may produce good results.

From this perspective, an agent which behaves by itself while the agent shares the goal of the team is focused on. The purpose of our study is to examine the influence and effectiveness of such an agent in a simulated soccer team from the standpoint of its team’s performance such as the number of wins and losses, scores, the number of shots, considering whether the agent could yield a good result. In this paper, the one-man agent is defined as an agent which behaves by itself without others’ cooperation while the agent shares the goal of its team. "rcsserver[8]"(RoboCup 2D Soccer Simulator) is used as a soccer simulator.

In this paper, the rosserver as a platform is explained in section 2. The definition of the one-man agent is given in section 3. This is the key word of our study. Section 4 describes the experiment and the results. The effectiveness of the one-man agent is evaluated. The final conclusion and future work are presented in section 5. In Appendix the analytical indicators are illustrated.

2. RCSERVER
The rosserver, soccer simulator provided by the RoboCup Soccer Simulation league 2D. The soccer simulation system is realized as a client-server system as shown in Figure 1: rosserver as a server, each agent as a client[9].

An agent(player) on the RoboCup Soccer Simulation 2D makes a decision based on information of visual and auditory perception received from server and acts. Objects information(name, relative distance, relative velocity and relative angle) are received as visual perception. However, the agent’s view angle is restricted and accuracy of these information reduces depending on an distance to the object. The agent behavior is performed by sending a low level command such as "kick", "dash" and "turn" to the server. The kick command is executable when the ball is in the kick-able area. Also, behavior such as "dribble" or "pass" is performed by combination of these low level command.
Table 1: Overview of Behavior Rules (the agent2d and the one-man agent)

<table>
<thead>
<tr>
<th>Conditions</th>
<th>agent2d behavior</th>
<th>one-man agent behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>decision-making</td>
<td>agent2d behavior</td>
<td>one-man agent behavior</td>
</tr>
<tr>
<td>Should I make a shot?</td>
<td>shoot</td>
<td>shoot</td>
</tr>
<tr>
<td>Should I pass?</td>
<td>pass</td>
<td>(*)</td>
</tr>
<tr>
<td>else</td>
<td>dribble</td>
<td>dribble</td>
</tr>
<tr>
<td>Should I go to the ball?</td>
<td>go to the ball</td>
<td>go to the ball</td>
</tr>
<tr>
<td>else</td>
<td>take a position</td>
<td>take a position</td>
</tr>
</tbody>
</table>

3. ONE-MAN AGENT

In our study, the one-man agent is defined as an agent which always takes the one-man approach and simultaneously shares the goal of its team — to win against an opponent team —. Then, the one-man approach is realized as the behavior that the agent dribbles the ball toward the opponent goal and then makes a shot without passing to its team mate.

Specifically, the one-man agent is implemented by removing "pass"(*) from its behavior rules of the team agent2d[10]. Table 1 shows the overview of behavior rules that compares the agent2d with the one-man agent.

4. EXPERIMENTS AND RESULTS

To examine the influence and effectiveness of the one-man agent in a simulated soccer team, experimentations were performed through simulated soccer games. Then, the experimental results of the teams including the one-man agent were compared with the results of the team not including the one-man agent. The number of wins and losses, scores, the number of shots and trajectories of dribbling were used as analytical indicators.

Figure 2: Agents' positions in an experimental team (Each agent shows an agent, and inner-number is its uniform number.)

4.1 Experiment Description

The experiments were carried out through simulated soccer games (eleven-on-eleven). There are 11 experimental teams below. Each experimental team played 50 games against agent2d. One game has 3000 simulation steps.

An experimental team is a team that one agent of the team agent2d is replaced with the one-man agent. Figure 2 shows the formation of the experimental teams. Each circle represents an agent. The digit in each circle is an uniform number of agent. The goal keeper wears uniform number 1. The formation and agents' positions in the experimental teams are defined as follows.

\[ T_i = \{p_j | 1 \leq j \leq 11 \} \]  \hspace{1cm} (1)

Where \( T_i \) is an experimental team, \( i \) is a number of experimental team, \( p \) is an agent(player) and \( j \) is an uniform number of an agent. Also, the set of all experimental teams is described as follows.

\[ \{T_i|0 \leq i \leq 11, i \neq 1\} \]  \hspace{1cm} (2)

Where \( i \) is a team's identification number and is also the uniform number of the one-man agent of the team. There is no team \( T_{11} \), because the goal keeper(uniform number 1) is not replaced with the one-man agent. When \( i = 0, T_{10} \) is the same team as the team agent2d. The experimental results of team \( T_{10} \) were utilized as the basis of analysis.

4.2 Experimental Results

The bar chart in Figure 3 shows the results of win-lose through 50 games for each experimental team. The x axis is the number of wins and losses. A white bar shows the number of wins. A black bar shows the number of losses. The y axis represents each experimental team. Team \( T_{10} \) had 19 wins, 0 loss and 31 draws and team \( T_{1011} \) had 21 wins, 0 loss and 29 draws. These teams won better than the others. Compared to team \( T_{10} \) which has no one-man agent, the numbers of win games of team \( T_{10} \) and team \( T_{1011} \) are 17 and 19 greater than those of team \( T_{10} \), respectively. The results have great advantage of the number of win games.

The bar chart in Figure 4 shows the total scores of each experimental team and the scores obtained by each agent. The x axis is the total scores. The y axis represents each experimental team. The graph legends show which score. Team \( T_{10} \) and \( T_{1011} \) score 24 and 25, respectively. It shows the great advantage of scoring from other teams.

Table 2 is the comparative table of the results of teams \( T_{10} \) and \( T_{10} (=\text{agent2d}) \). Avg.(average) and SD.(standard
Figure 3: Win-lose results of experimental teams against the team agent2d (50 games per an experimental team)

Figure 4: Total scores of experimental teams and agents against the team agent2d (50 games per an experimental team)
Table 2: Comparison of Experimental Results (Number of Shots and Scores) between teams $T_{b0}$ and $T_{b9}$

<table>
<thead>
<tr>
<th></th>
<th>Number of Shots</th>
<th>Scores</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Avg.</td>
<td>SD</td>
</tr>
<tr>
<td>$T_{b9}$</td>
<td>0.04</td>
<td>0.27</td>
</tr>
<tr>
<td>$T_{b9}$</td>
<td>0.04</td>
<td>0.27</td>
</tr>
<tr>
<td>t-value</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Values in parentheses are the results of the agent wearing uniform number 9.

Table 3: Comparison of Experimental Results (Number of Shots and Scores) between teams $T_{b0}$ and $T_{b11}$

<table>
<thead>
<tr>
<th></th>
<th>Number of Shots</th>
<th>Scores</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Avg.</td>
<td>SD</td>
</tr>
<tr>
<td>$T_{b0}$</td>
<td>0.08</td>
<td>0.27</td>
</tr>
<tr>
<td>$T_{b0}$</td>
<td>0.04</td>
<td>0.27</td>
</tr>
<tr>
<td>t-value</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Values in parentheses are the results of the agent wearing uniform number 11.

The average number of shots and the average scores of team $T_{b9}$ were better than them of team $T_{b0}$. To clarify if the difference of number of shots and scores have statistical significance, team $T_{b9}$ is tested based on team $T_{b0}$. The t-values of number of shots and Scores were 0.00, respectively, in Table 2 and had statistical significance. These results indicate that the team performance of team $T_{b9}$ was better than it of team $T_{b0}$. The one-man agent (the uniform number 9 of team $T_{b9}$) obtained the average number of Shots 0.40 and the average scores 0.32. The one-man agent gained more than the agent in the same position of the team agent2d (the uniform number 9 of team $T_{b0}$). The one-man agent wearing uniform number 9 scored 16 goals out of 24 through 50 games in $T_{b9}$. Thus, 67% of total scores of team $T_{b9}$ was scored by the one-man agent.

Table 3 is the comparative table of the results of teams $T_{b11}$ and $T_{b0}$ (=agent2d). Avg. (average) and SD (standard deviation) of number of shots, Avg. and SD of scores, and t-values were calculated in Table 3. Values in parentheses are the results of the agent wearing uniform number 11. The average number of shots and the average scores of team $T_{b11}$ were better than them of team $T_{b0}$. To clarify if the difference of number of shots and scores have statistical significance, team $T_{b11}$ is tested based on team $T_{b0}$. The t-values of number of shots and scores were 0.00, respectively, in Table 3, and had statistical significance. These results indicate that the team performance of team $T_{b11}$ was better than it of team $T_{b0}$. The one-man agent (the uniform number 11 of team $T_{b11}$) obtained the average Number of Shots 0.56 and the average scores 0.48. The one-man agent gained more than the agent in the same position of team agent2d (the uniform number 11 of team $T_{b0}$). The one-man agent wearing uniform number 11 scored 24 goals out of 25 through 50 games in $T_{b11}$. Thus, 96% of total scores of team $T_{b11}$ was scored by the one-man agent. These results suggest that the one-man agent agents in teams $T_{b9}$ and $T_{b11}$ contribute to the improvement of the teams’ performance, respectively.

At the end, these teams’ trajectories of the ball are shown in Figures 5, 6, 7 and 8. Each figure represents a soccer field. The right side is the opponent side in each figure. Compared to team $T_{b0}$ in Figure 5, the whole trajectory of the ball (light gray lines) of team $T_{b9}$ in Figure 6 was changed clearly. Also, compared to the dribbling trajectory (black line in Figure 5) of the agent (the uniform number 9) of team $T_{b0}$ (=agent2d), the trajectory (black lines in Figure 6) of the one-man agent (the uniform number 9) of team $T_{b9}$ shows that the one-man agent headed to the opponent goal rather than the agent of team $T_{b0}$. It leads that the team $T_{b9}$’s chances of shots (dark gray lines) increased rather than team $T_{b0}$ of them. The same tendency can be seen between the one-man agent of team $T_{b11}$ in Figure 8 and the agent (the uniform number 11) of team $T_{b0}$.

4.3 Discussion

The experimental results suggest that the one-man agent may contribute to the improvement of its team’s performance. This means that the behavior by itself without other’s cooperation gives positive results such as achieving the goal of its team and improvement in its team’s performance.

Some of the experimental teams have not significantly influence on its teams’ performance. The reason is assumed to the positions of the one-man agent. That is the result that a positions in the soccer represents coordination in a team and a role of agent such as Forward (FW), Midfielder (MF) and Defender (DF) implicitly.

The one-man agents of teams $T_{b0}$ and $T_{b11}$ are FW players. The experimental team $T_{b10}$’s one-man agent is also located a FW position, but the experimental result of team $T_{b10}$ is not significantly improved(Table 4).

Further analysis is needed, however, it could be caused by some behavior of the team agent2d (e.g. the one-man agent of team $T_{b10}$ may tend not to be passed because of the team mates’ pass priorities ).
Figure 5: Ball trajectory (team $T_{E0}$ vs. the team agent2d, 50 games total). Light gray lines is the whole trajectory of the ball. Black lines is the dribbling trajectory by the agent wearing uniform number 9 of team $T_{E0}$. Only dribbling play was extracted from the other behavior including a passing. Dark gray line is the shot trajectory by the agent wearing uniform number 9 of team $T_{E0}$.

Figure 7: Ball trajectory (team $T_{E0}$ vs. the team agent2d, 50 games total). Light gray lines is the whole trajectory of the ball. Black lines is the dribbling trajectory by the agent wearing uniform number 11 of team $T_{E0}$. Only dribbling play was extracted from the other behavior including a passing. Dark gray line is the shot trajectory by the agent wearing uniform number 11 of team $T_{E0}$.

Figure 6: Ball trajectory (team $T_{E9}$ vs. the team agent2d, 50 games total). Light gray lines is the whole trajectory of the ball. Black lines is the dribbling trajectory by the agent wearing uniform number 9 of team $T_{E9}$. Dark gray line is the shot trajectory by the agent wearing uniform number 9 of team $T_{E9}$.

Figure 8: Ball trajectory (team $T_{E11}$ vs. the team agent2d, 50 games total). Light gray lines is the whole trajectory of the ball. Black lines is the dribbling trajectory by the agent wearing uniform number 11 of team $T_{E11}$. Dark gray line is the shot trajectory by the agent wearing uniform number 11 of team $T_{E11}$.
5. CONCLUSIONS

In our study, the one-man agent is defined as an agent which behaves by itself without others' cooperation while the agent shares the goal of the team with the team mates. Then, the influence and effectiveness of the one-man agent in a team based on coordination was analysed from the standpoint of its team's performance through simulated soccer games. The situations that the one-man agent plays effectively will be analyzed. After that, our research will be advanced on the realization of how an agent switches behaviors that the agent behaves by itself or cooperatively.

6. REFERENCES


http://sourceforge.net/apps/mediawiki/robocup/.


APPENDIX

A. ANALYTICAL INDICATORS

The analytical indicators of "shot" and "dribble" are explained here.

The script parsed the log data (including the ball position and velocity, agents’ positions and velocities, and etc. during a game) and judged "shot" and "dribble" automatically according to the definitions of "shot" and "dribble" below.

Figure 9: Situation of an agent’s making a shot

![Figure 9: Situation of an agent’s making a shot](image1)

Therefore, the values shown in the experimental results can be not perfectly matched with the values judged by human.

A.1 Shot

The shot is defined as a kick that the ball is moved toward the opponent’s goal area (Figure 9). In other words, after the ball is kicked if the angle of the ball velocity vector ($\text{ang}$ in Figure 9) is between the Poles A ($\text{angA}$ in Figure 9) and B($\text{angB}$ in Figure 9) (Equation 3), the kick is regarded as a shot.

\[
\text{angA < ang < angB}
\]

A.2 Dribble

The state of dribble is defined as a situation that a ball possession situation is continued by an agent. The ball possession situation is defined as a situation that the distance $D$(Figure 10) between the ball and the agent is less than the threshold and no other agent exists within distance $D$ from the center of the ball. $D$ is set to 1.0. An continuum of the ball possession situation by an agent and the next continuum of the ball possession situation by the same agent are considered as one continuum of the ball possession situation, if there is no ball possession situation by the others between these two continuums. However, it should be during progressing a game (play on mode).
Avatars in a modern soccer manager

Ubbo Visser
Department of Computer Science
University of Miami
1365 Memorial Drive
Coral Gables, FL, 33146 USA
visser@cs.miami.edu

Christine Laetitia Lisetti
School of Computing and Information Sciences
Florida International University
University Park, Miami, FL 33199 USA
lisetti@cis.fiu.edu

ABSTRACT
This paper describes recent research results in the area of agents in dynamic environments and emotional avatars. A online soccer manager has been developed over the past few years that is subject to an online game carried out by the German Bundesliga. More than 200,000 users operate the Official Bundesliga Manager and TopLeague, complex real-time soccer simulators that are based on actual data of real professional soccer players. The underlying technology is a hierarchical three-tier multiagent system that consists of autonomous BDI agents and allows dynamic group structures (e.g. an emergent situation for a wing attack). The online games run seamlessly in a web browser with a state-of-the-art 3D visualization engine. We describe the underlying technologies of the game and focus on recent advances in motion capture techniques for graphical body animation and facial expressions.

1. INTRODUCTION
A vast amount of games have been developed for the mass market over the past decade. Not only can we observe more than ten types of games (e.g., adventure games, casual games, sports games [1, 31]), we can also state that most of the games have been produced for specific hardware (e.g., consoles, game consoles, Wii consoles). Online games have also been developed in great numbers [17], however, web browser restrictions prevented them from being as complex - especially in their graphical animation - as their console counterparts. One reason for this is the computers configuration (e.g., real time rendering, older computers have limited rendering capabilities). Some hardware related features (e.g. vertex shaders) may not work on older machines. Another reason is security (applets for instance cannot change the users configuration without prior permission). Our research reveals that a significant number of users (approx. 10%) have not installed their graphic drivers properly.

Nevertheless, online games have become more and more complex, especially in the last few years. We see a merge between the technologies, e.g. consoles have Internet access now and can download the latest updates while still having an advantage with a defined hardware. On the other hand, the market share of online games (and mobile applications) are increasing [22]. We can even argue for a paradigm shift as nowadays computers and network bandwidth reach a point where complex real-time online games are possible for the mass market. The RoboCup simulation league offers a number of research targets. Among them are complex simulation [24, 23], real-time situation estimation [5, 39, 47, 19] and behavior of agents [42, 18, 48, 29]. Starting off as a simulation team in the early years of this decade we generated the idea to combine the new game paradigm shift with autonomous robots and sports data from real professional soccer players. We have developed a simulation engine where the agents comply to weekly updated performance indicators of real players (e.g. accuracy of shooting, tackle performance) and act according to tactical preferences of the user who operates the game.

Game play:
The general idea for the user is to pick a virtual version of a professional soccer team and act as a coach. Games follow the regular season and the official game days follow the game days of the professional leagues. The user adds soccer knowledge for the next game while choosing formation, tactics, and training sessions for his team. Each game is simulated and a newly developed and high quality 3D graphics engine allows the user to see the match in a web browser thereafter (cf. figure 1). The user coaches a real team (e.g. Bayern Munich) and has the same problems to solve than

Figure 1: Emotion in autonomous agents in TopLeague
the real coach in the real world (e.g., players that are suspended in the real world are also suspended in the virtual world). The games have the same rhythm than the reality, however, TopLeague\(^1\) games will be played one day earlier.

The game is significantly different from the known PC/Console games such as FIFA Soccer 10 or Pro Evolution Soccer. The major differences lay in user interaction (our user is the coach not a player on the field), and visualization (our game is an applet in a web browser).

**Contribution:**

This paper has at least three distinct contributions: 1.) The existence and availability of an online soccer game on free market with large group of users that giving exclusively positive comments. 2.) A novel paradigm of allowing same player skills in virtual world than in the real world. 3.) The 3D-Visualization in a web browser demonstrating that the latest research results in the area of motion capturing for body motions are promising and necessary.

This paper introduces the most important aspects and technologies of the game. The focus, however, is drawn on recent research in the area of subtle facial expression for avatars and characters in multiagent systems. We address the need for virtual characters to be able to display a wide range and subtleties of facial expressions under the given restrictions of an applet and under real-time conditions.

The paper is organized as follows: the architecture including simulator and graphics engine of the game is presented in the next section. We then describe our animation process discussing our latest research results including emotions on avatars.

### 2. ARCHITECTURE

All data are held in a relational database (MySQL). A number of modules as middleware have been developed ensuring the scalability of our game. A scheduler is responsible for the list of games that need to be evaluated (see figure 2).

![Figure 2: Setup of MAS with MySQL DB, scheduler, and team structure](http://www.topleague.de)

### 2.1 Simulator

The simulator is developed for a sports game in general. In this instance, however, it is used to simulate a soccer game. It consists of a soccer environment with the necessary agents (two teams, referee, substitute players). There is no physics involved in the game with one exception: the ball has an accurate model for gravity, air friction as well as bouncing effects on the ground and for the posts. The friction varies with weather conditions, a rainy playground has less friction for example. The multiagent system should consist of $n$ software agents and $m$ dynamically generated sub-structure agents that take over the coordination of specific play situations. Each team has optimal vision and accurate information about the world. They can communicate to each other and have a 100% access to each others world model. Each team is organized in a hierarchy.

Within the simulation, the team decides about the general strategy (e.g., offense play, wing attacks in general). The agents in the next level (stable team structures such as defense, middle field, forwards) balance out the given general strategy with their own goals (e.g., defense within an attack or distribution of the players within an attack). The lowest level consists of agents that act and react according to the given overall goals and the concrete play situation based on their personal skills.

![Figure 3: Agent architecture](http://www.topleague.de)

All agents 'exist' within an agent environment (cf. figure 3). The environment’s changes are sensed by each agent with the help of sensors leading to a world model. A perception of the world induces the reflection model that updates the internal world model of the agent with the actual perceptions and that validates the current action sequence taken by the agent (current commitment). A new deliberation cycle will be induced if the current action sequence is no longer valid (e.g., change of game situation). Within the deliberation cycle the various roles of an agent will be weighted based on the current situation evaluation. A role of an agent is directly linked to a list of behaviors, which can be used for the calculation of the action selection. A planning process is carried out that calculates the action sequence for the agent.
2.1.1 Input parameters:

All real professional soccer players are grouped into five skill groups. The skills are calculated for all players of same position in same group based on Opta Sports Data statistics. After this, we are ranking players within one group. If players don’t play due to sick leave or suspension, minimum skills will be taken for those players. Often, bad player skills can be observed (e.g. player haven’t played or played but did not score or perform well). In order to rectify this situation we perform small changes in skills randomly.

A general input factor is a random seed for the pseudo random number generator. Team settings include but do not exhaust home team/away team, formation, and heterogeneous players. Tactics include team, shooting and passing behavior, side of attack, rules for ball holding and rules for tactical changes throughout the match. The user is also able to define certain roles such as captain, penalty scorer, free-kick and corner-kick player. Player settings includes the level of aggressiveness and deployment.

The player attributes/skills (cf. figure 4) are based on Opta Sports Data. Thirteen skills are used either internally or externally. An example for internal skills that are only used inside our simulator is penalty.kick. An example for the an external skill that is also shown to the end user via web front-end is duel_strength. Examples for equations can be found in [45].

2.2 3D Visualization Engine

The user can interact with the game in various ways. Among setting formation and tactics, substitution players, substitution rules, and modern training sessions can be defined.

The simulator produces a < 250K log file that contains the positions of the players, the referees and the ball at each given time slot. The simulator runs with 10Hz. The log-file is then stored on one of our servers and can be requested from each user via web service. Technically, the log-file is requested by an applet on the client side that shows the game and various resolutions/options that are tailored for a variety of graphic cards.

All animation scenes are visualized in this graphics engine. It has been developed from scratch and guarantees high quality pictures in a web browser. The kernel is also domain independent and carries most of the features. Another module contains the 3D model for the stadium, scenes, textures, animations and interaction GUI for the soccer stadium. FaceIntegrator, developed in-house, is a tool that contains scenes and textures among 3D models. It also contains an implementation for the generation and integration of real soccer player images for our autonomous agents (cf. figure 11 on page ).

The graphics engine contains features that include static and dynamic models, (hand-made) animations, skeletons (soon also bipeds), terrain description, object motions for animations, motion differences for animations, bezier curves, vertex-, fragment-, and geometry shader, potential fields, and some configuration files. As an example, we describe the bone/joint system and dynamic scene generation.

2.3 Quaternion based bone/joint system

A quaternion-based bone/joint system allows a biquine interpolation of object rotations in 3D space. This is not possible within the Euler space because different object rotations can be combined in arbitrary orders and may end up in the same end-rotation position. Quaternions and their biquine interpolation characteristics allow us to fluently animate bone systems of dynamic objects with the help of a few key frames. An advantage is that we can animate objects at any given time independently from the frequency rates. The animated objects can be rendered in real time with software and hardware (shaders). Figure 5 shows our level-of-detail dependent use of bone systems. Objects in the foreground show more bones as objects in the background. In this scene, players and banners are animated objects. Note that the level-of-detail does not show the hands of players in the background.

2.4 Dynamic scenes

Users have the option to differentiate between various camera settings. All cameras can be used with a zoom. The cameras focus on an area close to the ball. In addition to the ball position in the game, more ball positions are estimated for potential future ball positions. This gives an illusion that the ‘camera man’ would take the scene live and would not know future ball positions. All ball positions are smoothened by cubic curves (Bezier). Using curves allows us a 3rd degree interpolation, which is significantly more dynamic than a 1st degree interpolation (linear interpolation). The soccer simulator marks special events such as goals, fouls, offside situation etc. These marks are the basis for a game situation evaluation which creates a suitable camera path. Figure 6 shows two different camera paths. The upper row shows a
foul situation, the lower row a goal situation. Each game situation is unique and creates a unique camera path.

The actual hosting architecture (OS, Web Server(s), Application, Server(s), DB Server(s), firewalls etc.) is dimensioned for the existing number of users (200K in total, approx. 30K unique user per month). The two web servers (DualCore Xeon 5130@2GHz, 2GB) should have a distributed load for static content. The database server (2x DualCore Xeon 5130@2Ghz, 16GB) is adequate for up to ca. 200K users. Two simulation servers (each 2x QuadCore Xeon E5535@2GHz, 2GB) are adequate to handle up to 60K games per game day. The OS is RedHat Enterprise Linux 5.

This has an impact on the rendering capabilities and level of detail of our player models including facial expressions since it is eminent to distinguish between high-quality agents in a stand-alone application and a game that runs in an applet within a browser. Multiple agents in a browser game have 1,000-5,000 vertices these days whereas professional games engines that are tailored for the movie industry may have more than 10 million vertices.

3. MOTION CAPTURE FOR BODY MOTIONS AND FACIAL EXPRESSIONS

Character animation is an important part of a graphical game engine. The restriction of an applet (or better: the client machine and the applet) do not allow us to create high resolution models that we can see in console based games. Motion capture techniques are well-known and can be used for character animation.

There are many different approaches in the area of motion capture. Approaches contain non-optical systems such as inertial systems (inertial sensors, biomechanical models and sensor fusion) [46], electromechanical motion [43], and electromagnetic systems [25]. Other techniques are optical. Among these are semi-passive imperceptible markers [40], pure vision-based, markerless techniques [26, 16] as well as active markers [50].

The most popular approach to motion capture today is to use reflective markers to the body and/or face of an actor, and track these markers in images acquired by multiple calibrated video cameras [44, 20, 41]. The marker tracks are then matched, and triangulation is used to reconstruct the corresponding position and velocity information.

Furukawa & Ponce [15] state that any motion capture system is limited by the temporal and spatial resolution of the cameras, and the number of reflective markers to be tracked. Matching becomes difficult with too many markers that all look alike or are getting lost while labeling. They state further that on the other hand, although relatively few (say, 50) markers may be sufficient to recover skeletal body configurations, thousands of markers may be needed to accurately recover the complex changes in the fold structure of cloth during body motions, or model subtle facial motions and skin deformations (see also [49]).

Our system is a mixture of hand-made animations and those that are made with the help of motion capture techniques (reflective markers). The Motion Capture Lab of the University of Miami is equipped with a modern Vicon System. It consist of ten 4-megapixel cameras with 120 Hz, two video cameras for ground truth, four force plates to measure forces on ground, facilities for facial expressions, and an electromagnetic device capture voltage in max 16 muscle groups. Figure 7 outlines the various steps (workflow) that have been performed to get a motion onto animated characters that can be used for a web browser game. One important step for both body and facial motions/expressions is the choice of numbers of passive markers. This is crucial and subject of a tradeoff at the same time. On one hand, the more markers we capture, the better and more precise the motions can be. On the other hand, the more markers we have the more difficult and cumbersome the labeling process will be. This process defines and assigns each point in 3D to a bone or joint of the character model.

We conducted various experiments with different marker sets for the body motion capture. We have tailored with process to a biped model that we use for our game engine. We started with nearly 50 markers, which was optimized for the real soccer player who acted as a model. After a few experiments we discovered that 33 markers are enough to cover even complex body motions as long as we model a single player. This reduces the amount of points in 3D space significantly. For a 1 second animation trial with 55 markers on 10 cameras that run on 120Hz, for example, 66,000 data points have to be considered. If we have 33 markers, we end up with 39,600 data points, which is a decrease of 40%. The labeling process is a tedious part, especially if we are confronted with markers that are lost in certain frames. This can occur if either the markers are falling of the real model during the motion capture trial or the same markers are occluded. The latter problem occurs if some markers are occluded by body parts that move around (e.g. the torso occludes an arm). Usually, this problem occurs less the more cameras and camera angles are used. 10 cameras are enough to cover this problem in general. This situation occurs even more if more than one body needs to be captured. Not only doubles the amount of data points if there is another person in the scene, the number of occluded data points increases as well. The tedious part is to decide to which body part of which body the lost and then re-occuring data point belongs. The next step is the first cleanup process. Here, the noise in the scene needs to be eliminated up and the data are mapped onto a bone model. Usually, the real
soccer model has different proportions than the character model. An example for the translation transformation is that the shoulders of the character are hanging because of the proportion problem. This needs to be rectified on the model (figure 8b).

A biped (figure 8c) with a mesh for the body style is then introduced and adjusted in every animation. The mesh is important because we can identify some errors or flaws that could not be detected on the bone system alone. An example for this is an elbow that is too close to the body. Once this is done, the full animation can be rendered into a video with the appropriate and available software (figure 8d).

Another big step is the transformation from a single animation that is on a character model into the game engine. This step depends heavily on the game engine. We use our own game engine that has been developed in-house. More small mistakes can be detected such as feet that do not touch the ground completely or go into the ground in some frames. This usually includes re-iteration of the former steps.

A large and for real-time browser games unsolved problem - to our knowledge - is a smooth concatenation of complex animation scenes. The problem lays in the complexity where n single animations need to be calculated for concatenation for m agents in real-time. Motion composes across the agent's body and across time which makes the number of available motions huge. Agents might also interact with objects and other agents. There is thus a need for the creation for families of motions.

3.1 Modeling Avatars

There have been many recent advances in character animation in terms of body language and gestures [34, 33, 36, 6], quality of synthesized voice, social dialog abilities [4], mark-up languages [37, 33], facial expression animations [35, 32, 36, 6], empathic behavior [38, 21, 30]; to name a few. One of the main challenges still facing character animation research is that whereas most current implementations of avatars can display facial expressions of emotions, most of these approaches are limited to Ekman's six basic emotions of happiness, fear, anger, surprise, disgust, and sadness [9, 12, 11, 14, 13, 8, 10]. However, this makes it difficult to express more subtle expressions such as concern, sympathy, or understanding.

On Dec 18, 2009, Fox launched a new movie called Avatar. Figure 9 shows two of the characters.

This is state-of-the-art in motion pictures, however, fully rendered offline and turned into a video for watching. Even modern cluster computers need a long time for rendering, given all the shaders, lighting sources and character movements. This cannot be compared with the approach that we are proposing since there is no real-time interactivity with a human based on emotion recognition and feedback of conversations.

Modern engines (e.g. CryENGINE3) offer an excellent resolution for avatars, they can have as much as 10 million vertexes. This is a significant difference, however, a conversation between avatars and humans in real-time with non-verbal feedback is not possible.

The final goal therefore is to address the need for virtual characters to be able to display a wide range and subtleties of facial expressions under the given restrictions of an applet and under real-time conditions. We create a first (admittedly rough) model of human facial expressions portrayed by humans during social dialog, by recording the motion capture of facial expressions and bust movements from a series of human-human dialog conducted in the context of social work interviewing. So far, we started from the avatar shown in Figure 10, which is developed in house. The current bone system allows us to do simple expressions such as happiness and anger. Further research will include a much finer granularity of movements and expressions due to an improved bone system.

Figure 10: Bone system of the character that serves as a model for the avatar

There are three main approaches to obtain facial models.

http://mycryengine.com/
Firstly, 3D scanning of faces. Face models can be complete representations of a polyhedron that is obtained with the help of the faces’ surroundings scanning algorithm. These algorithms can use the viewing sphere and the perspective projection concepts (known as the K-M view space model) [28, 27, 3]. Secondly, available software products that come with pre-defined models such as FaceGen\(^4\), Poser\(^5\) or Daz-Studio\(^6\). FaceGen has been created using statistical modeling of the shape and appearance of human faces. FaceGen uses a similar approach to Blanz’s [2], although the implementation differs in the details, and much additional work has been done to add features and allow integration in the character-creation pipeline. Poser offers eight ready-to-pose and textured humans of different ethnicities. Lastly, hand-made models with other software products such as Poly Modeling with 3D Studio Max or ZBrush. The basic idea here is to create characters from scratch using generic spheres from which polygons are used to create the desired shape.

We use existing software tools (e.g. FaceGen) and develop necessary tools (e.g. FaceIntegrator) ourselves (see figure 11 and 12 for the outcome).

![Figure 11: A soccer player's picture projected on a morph with FaceGen](image)

### 3.2 Facial Expression Generation

Much work is being performed to animate believably avatars [34, 33, 36, 6, 37, 33, 53, 52, 36, 6]. To generate facial expressions, the approach taken by many research groups has been to use the well known Facial Action Coding system [9, 12, 11, 14, 13, 8, 10] to extrapolate different facial areas that are then independently activated according to predefined patterns [34, 33, 36]. These approaches address the difficult notion of expressing mixed emotions, but because it is based on Ekman’s static encoding of the face, no specific emphasis is given to the dynamics of the facial expression generation. However, as pointed out in a study on communicating facial affect in computer-mediated communication and in affective computing applications, if there is a tradeoff to be chosen between showing highly realistic facial images (i.e. high spacial resolution, full color) at the expense of motion (e.g. low temporal resolution, lags), the motion should be retained as it represents the most important factor in communicating facial affect [7].

Morph target animation is a method that is used as an alternative to skeletal animation. Morph target animation is stored as a series of vertex positions which are moved to a different position in each keyframe.

FaceGen has the ability to easily generate morphs of multiethtnic avatars, which is an added advantage for our multicultual avatar dialog system. We generated a small number of dynamic facial expressions (see figure 12) using our simple player avatar model, following the dynamic principles for facial expression generation in humans.

### 4. RESULTS

We have carried out the mentioned stages and included body motion, included real soccer player’s faces on our current player model with simple facial animations and made the new agent available in our game. In order to generate natural scenes, the emotions fit to the game/play situation (e.g. figure 12 shows a facial reaction after scoring). We have also included facial animation for pain after a foul and variations of scoring and pain. A large group of users in our game give constant feedback for our latest research results. The comments are exclusively positive about the necessity of having a close appearance between the real soccer player and its virtual counterpart. A survey with more than 1,000 users has revealed that our steps improved the quality of the game and thus generated an overall positive feedback.

We are currently experimenting with more subtle facial expressions in the motion capture lab. Figures 13a and 13b show our experimental setup with our human model who has X facial markers. We used 4 Vicon cameras for this setup and are currently experimenting with the right setup for a facial bone system. We hope to show new results by the time the workshop is being held.

![Figure 12: Avatars with Expressions of Emotions within a browser applet being built.](image)

![Figure 13: Our setup in the motion lab: the human model (a) with the markers and a snapshot from a scene that has been captured (b).](image)

At present, we cannot say whether the number of markers on the face is sufficient for our purpose. There will be a

\(^4\)http://www.facegen.com
\(^5\)http://mysmithmicro.com/
\(^6\)http://www.daz3d.com
tradeoff between what is desirable in terms of subtle facial expressions and what is possible in a browser game in real time.

5. CONCLUSION

We have developed an online game that is available on the free market. Nearly three years after launching we can state that the proof of concept has been successful. The technical key features of our technology can be identified as a unique AI simulation engine with an authenticity module allowing same player skills in virtual world than in the real world. This is a new paradigm on online sports games and can be seen as the closest soccer simulation game to reality. Another key feature is the visualization which is a 3D state-of-the-art graphics engine for a web browser. Latest research results in the area of motion capturing for body motions are promising and necessary for a modern game. Simple facial expressions have been implemented as well and user feedback suggest for more realistic scenes. More subtle facial expressions are currently investigated. A new facial model will have to be created for this matter.

6. ACKNOWLEDGMENTS

We would like to thank the following individuals and organization for their help in order to conduct this R&D project: Dennis Pachur, Dirk Lohwasser, aiment GmbH, Bremen, Germany; Adam McMahon, Justin Stoecker, Armando Locay, AI & Games Group, University of Miami; Lester Melendez, Florida International University; Dr. Asfour, Moatas Eltoukhy, Biomedical Engineering Lab, University of Miami.

7. REFERENCES


Real-time Coordination Of Multi-Agent Systems: An Integrated View

Ghulam Mahdi  
LIRMM, Univ. Montpellier II,  
161 Rue Ada,  
34392 Montpellier Cedex 5 - France  
ghulam.mahdi@lirmm.fr

Abdelkader Gouaïch  
LIRMM, Univ. Montpellier II,  
161 Rue Ada,  
34392 Montpellier Cedex 5 - France  
abdulkader.gouaich@lirmm.fr

Fabien Michel  
LIRMM, Univ. Montpellier II,  
161 Rue Ada,  
34392 Montpellier Cedex 5 - France  
fmichel@lirmm.fr

ABSTRACT

Real-time coordination in multi-agent systems have been studied from different perspectives of computation, communication and other aspects of coordination. Here in this position paper, we present an integrated view of coordination covering agent reasoning, message passing, resource management and negotiation. We argue for an integrated and comprehensive approach of “real-time coordination” in one unified model of coordination. Our position regarding real-time agent coordination would result in overall better understanding of real-time coordination and performance amelioration in MAs. We analyze current approaches and present an outline for an integrated and comprehensive view of “real-time coordination”.

Categories and Subject Descriptors

H.4 [Team coordination methods]: Teamwork and heterogeneous agents

General Terms

Performance, Design, Theory.

Keywords

Multi-agent systems, real-time, coordination, integrated view

1. INTRODUCTION

Coordination has remained a key concern in multi-agent systems studies in order to achieve globally coherent results [19, 1, 4]. Some works even argue on the separation of treatment for computation and coordination as two distinct and orthogonal dimensions of all useful computing inferring explicit treatment of coordination [9].

Recently real-time coordination has been introduced in the fields of robotics [18], traffic management [2] and multimedia systems [23]. As multi-agent systems are being proposed for time-sensitive domains, we propose a similar study to be carried out from the perspective of real-time systems. Here in this position paper, we argue that there is a need of an integrated and comprehensive view of “real-time” phenomenon when suggested for multi-agent systems. The idea is to utilize lessons learned from earlier studies in multi-agent systems carried out from different perspectives of “real-time” processing and suggest an integrated view covering all these aspects. Such an integrated view would address a key challenge for agents in real-time and dynamic environments by keeping real-time at all levels of computation and coordination. Earlier studies on the subject are either limited to suggesting real-time in agent behavior or proposing it for only aspect of coordination. As a result the real spirit of real-time performance is lost. The objective of this article is to shed some light on an integrated and comprehensive view of real-time coordination presenting all aspects of multi-agent systems processing as sub-processes being followed by time constraints. The rest of this article is structured as follows: The next section introduces some motivations from earlier works and real world scenarios for investigating coordination issues; Then the following section is about characterizing some key dimensions of coordination and influence of “real-time” on them; Later we discuss classification to our suggested of coordination then we present some guidelines as our criteria for measuring integrated real-time coordination; Finally a discussion presents overall features of such integrated view and conclusions of our study.

2. MOTIVATIONS

Real-time systems and multi-agents systems have individually contributed to many complex, heterogeneous and diverse real-world applications even before joining hands to be applied in domains particularly known for distributed, time-critical and autonomous features. A transfusion of both disciplines has shown quite interesting results in diverse domains ranging from sensor networks [21, 20] to virtual classrooms [16] and from e-commerce applications [5] to soccer robots [14]. Apart from these applicative studies of multi-agent and real-time systems, many works have tried to define, develop and implement efficient agent models presenting features of both disciplines (like [12, 13]); Some others have focused on the frameworks simulating such real-time agents [17]; Even others align to developing joint architectures for such type of agents [22, 3].

However a fundamental issue of coordinating multi-agent systems at real-time constraints has remained largely unaddressed, even if some of works which have tried to address the agent coordination are limited to focus on only one aspect (sub-process instead of addressing the issue in an integrated manner. Our motivations to address the problem has theoretical as well applicative inspirations. From the perspective of applications, most of real-time multi-agent systems are functioning in resource-constrained environments. In such domains of applications, usually agents share a single precious resource for a limited time to an agent so there is a need of coordinating the system processes for effective
utilization of the resources in given time. Examples of such limited resources include single road, bridge, clean air, energy resource in the domains of virtual emergencies, hospital rooms and virtual rescue systems. In other domains, where it seems that there isn't apparent inadequacy of resources as satellites, printers, servers and network traffic but exorbitant nature of these resources leads for sharing distribution of the tasks and the common resources for a limited time. When there is involvement of multiple entities for sharing a resource or distribution of tasks there is an inherent element of competitiveness. In case of resource management "real-time coordination" may have more critical job to do as all comprising autonomous agents try to get the maximum of the resource share for most of times and in task distribution for common goals- each agent may have a "get the least share of the common task" tendency and above all ensuring due share of resources to all the agents and timely completion of the tasks. Theoretical motivations for our study come from the realization that coordination problem at real-time level has not been viewed in agent studies. This "position" nature of work examines agent coordination studies at different levels and suggests an integrated as well comprehensive framework for efficient "real-time coordination". The time factor is so immersely embedded in the studies of multi-agent systems that most of coordination studies have an intrinsic treatment if not the explicit one as we are mentioning below. On the applications side, we realise that a lack of an efficient coordination mechanism in health services, virtual emergencies, distributed weather forecasting and traffic control may lead to performance degradation if not catastrophic results.

3. THREE DIMENSIONS OF AGENT COORDINATION

Edmund Durfee in [6, 7] characterizes three dimensions of scaling up agent coordination namely agent population, task environment and the solution properties. He argues that an efficient coordination mechanism needs to address all three dimensions of coordination. Here we briefly discuss all three dimensions and present our perspective that how temporal behavior influences all of three dimensions of coordination.

3.1 Agent Population Properties

Agent population properties include number of participating agents; Complexity i.e. how complex they are in their architectures and internal reasoning mechanisms; and heterogeneity of communication languages, ontological and internal architectures. Here coordination without having temporal constraints may lead to serious delays and performance degradation as the increase in number of agents, diversity and complexity would also require an increase in coordination time, if it is left to function on its pace.

3.2 Task Environment Properties

Task environment properties are about defining how an environment influences overall coordination task. Main characteristics of agent environment include the degree of interaction between participating agents, environment dynamics and distributivity of agents in the environment obviously require a coordination solution having "real-time" factor. In other case resulting coordination may turn not only ineffective due to normal pace of interaction but also irrelevant due to environmental dynamics and distributivity of agents.

3.3 Solution Properties

Solution of any proposed coordination mechanism have to consider how the resulting coordination is efficient in terms of quality, how much overhead it is there in terms of communication, computation and spent time along with robustness of the solution. Here real-time requirements are directly involved in terms of time spent and the resulting coordination in due time along with minimum overhead of time resources. Having a brief idea about the implications of "time factor" in agent coordination, now we individually describe the factors involved in amelioration of "time factor" in agent computations. By description these "performance contributors" we argue that these processes may be viewed as constituting sub-processes in global agent coordination process for an integrated view of "real-time coordination" in multi-agent systems. Here we describe different studies on these "performance contributors" and their role in integrated and comprehensive understanding of "real-time coordination" of multi-agent systems.

4. NOTION OF TIME IN AGENT COORDINATION

4.1 Time in Message Passing

Assuming agents process incoming messages atomically as soon as they receive them (or buffered in the message inbox), we need to take care of how much time it takes to deliver a message. Having timing constraints on the delivery of messages may play a substantial role in managing temporal behavior of the overall system. Embracing monitors that ensure timely dispatch of the messages do not have to come in conflict with the timing constraints in message processing in a way that message delivery is not to breach the agent encapsulation of how and when the message is processed, rather it's sole concern would be about timely delivery of the incoming messages. Such message dispatch monitors may be provided in one of two forms: Either, this monitoring mechanism would be made a part of agent message whose timing is maintained by the agent generating the message or some independent controllers which may have a check on message invocations and depending on later status of message invocations and agent constraints decides postponement or reordering of the messages in the message queue. Jamali et al. [10] suggest similar approach for multi-agent systems in resource allocation. Although the approach works quite finely, but its performance improvement is limited to resource management, in other words, it covers only one dimension of real-time performance in multi-agent systems.

4.2 Time in agent reasoning

Once a message is passed to the concerned agent, it may take some time to read the message, evaluate the contract and subsequently reply in denial or follow the message contents. If time factor is not involved in such message processing or agent reasoning it would unnecessarily affect the agent performance resulting in delay of overall coordination process. Many works of Julien et al. (like [12, 13, 22]) are addressed on development, design and implementation of real-time agents without considering coordination as the
main subject of studies. Here we need to make such models
enough flexible with other real-time computations like the
above mentioned ones. In absence of timing constraints the
system processes the messages may take too much time and
leading to affect the overall progress of the system.

4.3 Time in resource management

Agents being part of open systems compete for resources
due to sharing of independent computations. Such com-
petition to acquire resources leads to functional and non-
functional dependencies. Functional dependencies are about
whether sufficient resources are available or not, how to ac-
quire and release certain resources and how to deal with
multiple requests of the same resource at the same time.
By non-functional dependencies, we mean that availability
(or at least information of unavailability) of the required
resources in certain time bounds. Such availability or un-
availability information would be seen as an important fac-
tor in overall agent coordination. Here we need to manage
autonomy of agents in a way that agents are not to be let
to accumulate all the resources so here some type of re-
source management behavior is also recommended. Jamali
et al. [10]'s work on real-time resource allocations is seminal
on the subject that it not only ensures real-time in resource
allocations but also handles excessive resource acquisitions
problem common to agents based on actor model.

4.4 Time in negotiations

Although agent coordination doesn't imply cooperation
but many times coordination is seen as a co-operative pro-
cess to maintain heterogeneous body of agents in an envi-
nronment. Agent negotiations are used as a means to reach
an accord through communications. Agent negotiations are
usually seen as a compromising tool to mutual benefits of
efficient resource usage and task distribution. Despite ben-
etits of reaching an agreement, agent negotiations process is
presumed as a costly and time consuming practice. When
agent negotiations are left to work on their momentum it
would not only delay the coordination process but also con-
sume unnecessary resources. A model for real-time agent
negotiations for sensor networks is presented in [20], other
important works on the subject include Kraus et. al [15] and
Fatima et al. [8] but both of these works address negotia-
tions to be constrained by time rather than directly treating
it as real-time issue.

After a brief introduction of different real-time mecha-
nisms in multi-agent systems, we return to our earlier propo-
sition that coordination should be viewed as a meta-collection
of different sub-processes based on universal guidelines cov-
ering all aspects of coordination. Here we present our cri-
teria as a set of basic guidelines that we propose to be part
of any coordination mechanism suggested to fully address
real-time issue at all levels.

5. KEY PRINCIPLES AND GUIDELINES IN
REAL-TIME AGENT COORDINATION

Whether it is designing a coordination mechanism or de-
veloping and implementing a coordination model we obvi-
uously need certain guidelines and principles on which that
particular solution would be based. The idea is to see coor-
dination as a composition of different sub-processes and ap-
ply real-time constraints on all the constituent steps based
on some criteria. Our vision of real-time is partially influ-
on all sub-activities and Soh [21]'s ideas on for setting up a
criterion-based real-time systems. Here we briefly describe
our set of guidelines for real-time coordination:

1. Agent coordination may be seen as a compound pro-
cess involving all four sub-process viz a viz agents, re-
source management, negotiations and message passing.
Here an efficient approach to realise real-time behav-
ior would be the one which involves real-time in all
sub-processes.

2. Agent coordination must be confined by time bounds.
Take warbot (Unmanned systems in military applica-
tions playing a role in determining the success or fail-
ure of combat mission) agents as an example whose
job is to coordinate with other agents before a mis-
sile enters into their covered territory. Even if an effi-
cient coordination is achieved after the target reaches
their covered zone, the coordinate would lose its ap-
pliability as well as appropriateness. As a result, the
system is no longer useful for tracking the target. Sim-
ilar applications widely exist in health systems, sensor
networks and complex simulations where loosing time
constituent profoundly affects the progress and pur-
pose of a system.

3. All sub-processes of coordination should have a notion
of promptness. As coordination activity may involve
certain sub-tasks or steps like: Message passing, re-
sponding to the messages, negotiating, etc. All these
sub-steps must be performed with consideration to the
time issues.

4. Coordination process should have minimum number of
sub-activities. Coordination may involve multiple
activities of negotiating to reach an accord, generat-
ing message, handling individual message s and re-
sponding received messages. Although the number of
sub-processes may vary as per the situation, sensitivity
and nature of coordination but one universal principle
needs to respected is that as increase of the number of
sub-processes is directly proportional to increase in
time. The systems critical to time factors need to min-
imize number of such activities.

5. A coordination related message should have minimized
contents. Message processing would be as time taking
job as the contents of an individual message. Once a
short message is received the agent would facilitate it
in timely processing also if it needs to generate other
messages having shorter contents would not halt its
other processes.

6. Autonomous behavior in time message processing should
be used. Many times dispatching a message takes time
more than even message processing due to network
congestion, bandwidth limitation and other issues. In
such cases the agents should show a rational behavior
by aborting that communication and move on for next
message or agents instead of too much wait.

7. A coordination mechanism should be rapidly devised.
As a scenario shows a need of coordination, the partici-
pating agent(s) should device how much time is allotted
to the coordination, number of messages and other related prerequisites as soon as possible. Once these are decided agents can move for next steps. If chalking out the strategy takes much time, it will of course affect overall coordination process.

8. A coordination model should have suitable mechanism to differentiate between different tasks. Coordination in multi-agent systems have a dynamic nature in a way that the participating agents may be involved in some other activities while busy in coordination for some task. Here there are two issues: If agents choose to notice every tiny activity of course it will delay the coordination process while going for the coordination process by ignoring all the activities may bring finally an irrelevant solution. Here we need to device a mechanism that in which cases agents have to respond to the other actions besides their coordination activities and when they have to ignore tiny activities.

9. Agent negotiations should occur within time constraints. Agent based coordination is often involved by negotiations to reach an accord. There has been various attempts to improve the negotiation languages, terminology, and communication modes to achieve better communication but what we emphasize is to consider notion of “time” in the negotiation model.

10. Resource allocations should have a notion of time. Agent coordination frequently involves generation, consumption and freeing up resources for the computations. We need to emphasize that this important aspect of agent coordination should have some temporal behavior as it would affect overall coordination process.

11. Agent interactions should be time-bounded activity. Agent interactions are the means to interchange information with other agents, roles and environments. Emphasis on time bounds reminds the specification of temporal restrictions in these interactions. Here we need to specify interaction protocols with timing deadlines and expected influence on the quality of overall timing in the system.

6. DISCUSSION

Multi-agent systems have been studied from different aspects of “real-time”, namely reasoning, message passing, resource management and negotiations. All these aspects of “real-time multi-agent systems” can be seen as “sub-processes” in overall agent computation and coordination process. Real-time distributed computing processes can be viewed as a composition of sub-processes (instead of a single coordination process) where each process coordinates with its comprising components along with other sub-systems at its stage while being part of the global coordination process. There are two approaches to see the real-time coordination in multi-agent systems:

1. Incorporating time constraints on individual processes and coordination of any of the individual process would of course bring amelioration in the performance of that system but not at the optimum level.

2. Setting up a meta approach of real-time as well as coordination in a way that the coordination is involved at all sub-processes’ level which improves overall real-time performance of the system at global level.

Our vision to see real-time agent coordination distinct from other coordination mechanisms may prove useful in understanding both coordination as well as real-time performance of multi-agent systems. Due to the differences in the architectures and performance measures of different systems we suggest an integrated treatment of “real-time” problem at the level of each sub-process. Human societies also adopt coordination mechanisms which may involve myriad sub-processes, at some extent seem even irrelevant but after all serve a global purpose. Like an office working procedure may adopt different procedures and sub-processes for their coordination and time constraints but after all it serves timely performance of the main objective. Real-time multi-agent systems have special architectural foundations and design aspects current approaches in real-time agent systems were not set forth with those considerations therefore there is a performance as well as efficiency gap in effective agent coordination. Earlier studies on the subject have either dealt scalable agent coordination [6, 7] or particularizing coordination for different application domains (like [5, 16]). As per our knowledge we haven’t seen studies on real-time agent coordination which involves real-time in all sub-processes, although similar studies are carried out in robotics and communication domain from the perspective of coordination [2] and communication [18].

7. CONCLUSION

Here in this paper, we attempt to understand the peculiarities of real-time coordination for multi-agent systems. We argue that real-time coordination can be viewed as composed process consisting different sub-processes and an efficient model of coordination need to address and apply time constraints on all of them. Our findings of the particular guidelines and constituting sub-processes would have an impact in developing the relevant and potential works in the field. Our position regarding the subject defines the distinguishing guidelines of real-time multi-agent systems which may be considered as starting points for devising efficient coordination schemes in such systems. The approach discussed here would let both multi-agent and real-time communities to see each other’s requirements and prospectus in their domains. More precisely, the agent community to see coordination in multi-agent systems deal differently than it has been and the real-time community to take a more realistic picture about the agents’ functionality and effectiveness in multi-agent systems.

8. REFERENCES


Selection of Rendezvous Points for Multi-Robot Exploration in Dynamic Environments

Julian de Hoog  
Computing Laboratory  
Oxford University  
Wolfson Building, Parks Road  
Oxford OX1 3QD, UK  
 julian.dehoog@comlab.ox.ac.uk

Stephen Cameron  
Computing Laboratory  
Oxford University  
Wolfson Building, Parks Road  
Oxford OX1 3QD, UK  
stephen.cameron@comlab.ox.ac.uk

Arnoud Visser  
Intelligent Systems Laboratory  
Universiteit van Amsterdam  
Science Park 107  
NL 1098 XG Amsterdam, Netherlands  
a.visser@uva.nl

ABSTRACT

For many robotics applications (such as robotic search and rescue), information about the environment must be gathered by a team of robots and returned to a single, specific location. Coordination of robots and sharing of information is vital, and when environments have severe communication limitations, approaches must be robust to communication drop-out and failure. The difficulties are compounded in dynamic environments, where paths previously believed to be free can suddenly become blocked.

In this paper, we introduce a novel way of calculating rendezvous points for robots to meet and share information. Using role-based exploration, some robots continuously explore the environment while others ferry information back and forth to a central command centre. Optimal rendezvous point selection leads to more efficient exploration, and allows robots to replan when one of them has unexpected obstacles in its path.

Categories and Subject Descriptors
I.2.9 [Artificial Intelligence]: Robotics—Autonomous vehicles; I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence—Intelligent agents, Multiagent systems; I.4.10 [Image Processing and Computer Vision]: Image Representation—Morphological

General Terms
Algorithms, Experimentation, Performance

Keywords
Robotics, exploration, multi-robot cooperation, limited communication, search and rescue robots, role-based exploration, rendezvous points, dynamic environments

1. INTRODUCTION

Advances in robotics and multi-agent systems mean that robots will be used for an ever wider range of applications in the near future. Such tasks include reconnaissance, surveillance, exploration of environments inaccessible to humans (e.g. underwater or in space), and missions in potentially dangerous environments (e.g. bomb disposal or search-and-rescue).

In this paper we are particularly interested in the robotic search-and-rescue task, although our results are applicable to various robotic tasks. Search-and-rescue robots are used to explore environments after disaster scenarios (such as earthquakes) that are otherwise not accessible due to risks of secondary disaster, environmental hazards, or a lack of spatial access for humans or dogs. The hope is that robots will be able to efficiently explore and map disaster environments and find locations and statuses of human victims, so that human responders know where to focus their rescue efforts.

Currently, search-and-rescue robots are typically at least 40 - 60cm wide, long, and high, and are usually controlled directly by a human operator. Promising approaches include track-based robots, snake robots, and flying robots. As technologies improve and miniaturise, we believe that future robotic search-and-rescue efforts will involve teams of small rolling, crawling, or flying robots that autonomously explore environments of interest together. Such teams will require robust strategies for typical multi-robot team problems: team coordination, sharing of information, and limited communication.

Figure 1: A partially explored environment: walls are black, unexplored space is blue, explored (free) space is white. Thinning on the free space allows for calculation of possible rendezvous points (green dots). By choosing the best rendezvous points, robots can meet to exchange information more efficiently, and can replan to meet at another rendezvous point if one of them encounters unexpected obstacles.
In search-and-rescue environments, the limited communication problem is particularly relevant as disaster environments are likely to be full of obstacles and interference. Moreover, the additional problem of dynamic environments must be considered: unstable and burning rubble, for example, may well shift or change as the exploration effort unfolds.

In this paper we hope to make first steps in the direction of solving the following problem: how can a team of agents, subject to limited communication, be coordinated to (i) explore an unknown and possibly dynamic environment as quickly as possible while (ii) relaying known information back to a central command centre as quickly as possible?

Central to our approach is the use of role-based exploration, and the determination of optimal rendezvous points. In role-based exploration, team members assume one of two roles: exploring the far reaches of the environment, or relaying known information from explorers back to the command centre. To coordinate efficient meetings between explorers and relays (for information exchange), calculation of optimal rendezvous (i.e., meeting) points is crucial, and can significantly speed up the exploration effort.

While our experiments are an abstraction from the real-world problem of robotic search-and-rescue, we hope that our ideas and conclusions will be applicable to future robot rescue teams, possibly as an extension of existing multi-robot exploration algorithms. The results are also applicable to other problems where information by a team of communicating robots must be consolidated at a single location, such as for example in underwater or planetary exploration.

This paper is structured as follows: In Section 2 we discuss related work. Section 3 describes in detail our approach, including how we determine rendezvous points. Our simulation framework and communication model are described in Section 4, while our experimental results are outlined in Section 5. Finally we discuss the ramifications of our work in Section 6, and conclude in Section 7.

2. RELATED WORK

Multi-robot Exploration

Multi-robot exploration has received considerable attention in recent years but only a small number of approaches have taken limited communication into account.

In early approaches, a line-of-sight constraint was used to keep robots within communication range [3, 12]. This has been extended to robots reactively choosing a direction that will most likely keep them within sight of the rest of the team [17].

Several authors propose multi-robot exploration strategies based on market principles, in which robots place bids on subtasks of the exploration effort [22, 8, 28, 21]. These bids are typically based on values such as expected information gain and travel cost to a particular location in the environment, and may be assigned in a distributed fashion among team members, or by a central agent. When strength of communication is factored into the bids, robots avoid areas outside of communication range.

Another common strategy for robotic exploration is to use frontiers [27], which can easily be extended for use by multiple robots [5, 10, 18, 24]. Similar to bids described above, utilities of individual frontiers may include a factor related to likelihood of communication success, so robots are less likely to explore areas that take them out of the team communication range.

Further approaches include the use of 'energy fundamentals' to maintain network connectivity [20], results from graph theory to keep individual robots in 'comfort zones' [23] and the application of synthetic 'spring forces' to keep robots close to one another [16].

While several of these approaches have proven successful in maintaining team connectivity during the exploration effort, they are usually limited by the constraint of having to keep team members within communication range. Even if members of a team are dispersed to the maximum extent that their communication ranges allow, in large and complex environments unexplored areas will remain.

A solution to this problem is to allow robots to autonomously explore beyond communication range limits. This can be implemented in terms of 'robot pack' or clustering behaviour, in which groups of robots stay close together as they explore the environment [18, 21, 10].

However, little work has been done towards the typical search-and-rescue problem of gathering information in a severely communication-limited environment at a single location as efficiently as possible.

Rendezvous points

In several robotic exploration approaches, shared knowledge communicated at meetings between multiple robots is used for multi-robot localisation, although such meetings are not explicitly planned [9, 13]. The term rendezvous itself was introduced by Roy and Dudek in 2001 [19]. In their approach, robots wander through the environment and choose suitable landmarks for rendezvous, returning to the most suitable at a pre-arranged time.

Several authors have worked on the problem of efficiently gathering multiple agents with limited visibility at a single meeting point [2, 15], but in these approaches, exploration of the environment is not a goal. The rendezvous problem has also been phrased in terms of two agents entering a known environment at separate locations and having to find one another in minimal time [1]. We do not cover this problem here, however, since we are interested in applications where agents know one another's locations at the start of the exploration effort, but the environment is unknown.

Robot rendezvous is most relevant to exploration approaches in which individual robots (or groups of robots) are out of one another's communication range for extended periods of time. Since little work has been done in this direction, rendezvous selection and use remains a young field of study.

3. OUR APPROACH

3.1 Role-based Exploration

While most frontier-based exploration approaches lead to quick and efficient exploration, they do not take into account the need to relay new information back to a central command centre in communication-limited environments. To take advantage of frontier exploration's strengths while still maintaining as well connected a robot team as possible, we propose role-based exploration. We present a brief overview here; interested readers are referred to [7] for a more thorough description.
In role-based exploration each member of the team is assigned one of two roles:

1. **Explorer.** Explorers are meant to explore the farthest reaches of the environment. To communicate their findings, they return periodically to previously agreed rendezvous points where they pass their knowledge to a relay.

2. **Relay.** Relays ferry information back and forth between explorers and the command centre. This is achieved by meeting the explorer periodically at aforementioned rendezvous points, exchanging all relevant knowledge, and then returning to the command centre. If a relay discovers information about the environment while relaying, this is added to the team knowledge, but exploration is only a by-product of the relay’s movement.

### 3.2 Team Hierarchy and State Transitions

The team hierarchy is determined in advance. There may be multiple relays between the command centre and an explorer, and a relay may serve more than one explorer (see Figure 3). We are interested in dynamic team hierarchies as well, but leave this as future work for now.

State transition diagrams for Explorers and Relays are presented in Figure 2. Explorers’ GiveParentInfo state (which coincides with Relays’ GetInfoFromChild state) is particularly relevant to this paper: it is in this state that an Explorer plans next exploration steps, recalculates possible rendezvous points, and tells his parent relay where to rendezvous next.

Note that an Explorer and Relay do not need to reach rendezvous to transition to the next state. If there is a chance meeting between the two earlier than expected, it is advantageous to replan at that moment, rather than wait until both reach rendezvous.

### 3.3 Teammate Modeling

When two teammates, an Explorer and a Relay, meet, they exchange all relevant knowledge of the environment. After exchange, each robot will have the same map, and know exactly what its teammate knows at that point in time. Since relays’ movement is highly predictable and both robots use the same path planner, the Explorer can calculate exactly how long the Relay will need to return to the Command Centre (or its parent relay), turn around, and make its way back to the next jointly agreed rendezvous point. Thus the Explorer knows exactly how much time it has to continue exploring before having to turn around and rendezvous once again, and subsequent meetings can be timed in such a manner that neither Relay nor Explorer waste time waiting for the other to return to the rendezvous point – both should reach the rendezvous point at almost the same time.

Moreover, if the Explorer stores the map exchanged at rendezvous separately from its own evolving map, then it can at any point predict the Relay’s likely position, even when not in communication range (since the Relay’s map is unlikely to change much). Explorer and Relay can also agree on fallback rendezvous points, in case the preferred rendezvous point can unexpectedly not be reached. This has significant implications for rendezvous in dynamic environments, discussed in more detail in section 3.6.

### 3.4 Frontier Assignment

Assuming that the team hierarchy has been determined and each robot assigned a role, how does exploration actually take place? For this, we apply simple frontier exploration [27], which is among the most popular and promising
approaches today. Frontier exploration is heavily influenced by how utilities are calculated for individual frontiers. For every frontier \( f \) we calculate a utility \( U(f) \) as follows:

\[
U(f) = A(f)/C^n(f)
\]

where \( A(f) \) is the area of frontier \( f \), \( C(f) \) is the path cost from the robot to that frontier, and exponent \( n \) determines the exploration behaviour. High values of \( n \) lead to exploration of nearby frontiers (such as rooms) whereas low values mean that robots are more likely to pursue larger frontiers (such as hallways) [25]. For experiments reported later in this paper we use \( n = 2 \).

An additional consideration is that it is undesirable to send two robots into the same frontier. Elsewhere segmentation and the Hungarian method have been proposed [26], but we use a simple agent-frontier assignment algorithm detailed in [25]; in short, every robot determines frontier utilities for itself and its nearby teammates, and iteratively calculates a robot to frontier assignment that maximises joint utility. While this method is not necessarily optimal, it is fast, and in our experience entirely sufficient for distributed exploration.

### 3.5 Selection of Rendezvous Points

It turns out that this rendezvous point selection is an important factor in the exploration effort, and good rendezvous point selection both drives the exploration effort deep into the environment while minimising time required to communicate information up the communication chain. In our previous work, the Explorer stored its own current location at the moment that it turned to meet a Relay for use as the following rendezvous point. This did lead to deeper and deeper exploration of the environment, as with each rendezvous the Relay had to come deeper and deeper into the environment to meet the Explorer. However, in certain circumstances, rendezvous points chosen in this manner were less than favourable and led to inefficiencies (for example, when an Explorer chose a rendezvous point in an already fully explored part of the environment and had to backtrack unnecessarily to meet the Relay).

Here we propose a novel approach: subsequent rendezvous is calculated by the Explorer while it is in communication range of the Relay, and uses thinning on the free space in the map. Thinning is a technique from digital image processing that is meant to reduce a shape to its skeleton by making the shape as thin as possible while keeping it connected and centred. There are many parallels between thinning, skeletonisation, and Voronoi diagrams. A wide range of thinning techniques have been proposed since the 1960’s, having various advantages or disadvantages (for a review, see [14]).

In our approach we use Hilditch’s algorithm [11], since it is fast, returns a connected skeleton, and is easy to implement. A typical skeleton calculated using Hilditch’s algorithm is presented in Figure 1.

Hilditch’s algorithm requires the calculation of a neighbour traversal function \( T(p_i) \), described in Figure 4. This function can also be used to find junction points in the skeleton: any point \( p_i \) that is a junction in the skeleton will have \( T(p_i) \geq 3 \). A skeleton may contain long stretches without

\[\begin{array}{cccc}
p_9 & p_2 & p_3 \\
p_8 & p_1 & p_4 \\
p_7 & p_6 & p_5 \\
\end{array}\]

Figure 4: Traversal function \( T(p_i) \) is the number of 0,1 patterns in the sequence \( p_2, p_3, p_5, p_6, p_7, p_8, p_9, p_1 \)

junction points, for example along a hallway – to fill out the resulting graph, we iterate over all points in the skeleton and add those that are a minimum distance from all existing rendezvous points (filling). On the other hand, complex parts of the environment may contain a large number of junction points in a small area – to simplify calculations we choose only one point per given density (pruning). This gives a nice set of possible rendezvous points, distributed fairly evenly over the known environment and including all junctions. The full algorithm for rendezvous point calculation is presented in Algorithm 1.

List skeletonPoints = hilditchThinning(map);
List rendezvousPoints = new List;
for each \( sp \in \) skeletonPoints do
if \( neighbourTraversal(sp) \geq 3 \) then
    rendezvousPoints.add(sp);
end
end
for each \( sp \in \) skeletonPoints do
    boolean addToList = true;
    for each \( rp \in \) rendezvousPoints do
        if \( sp.distanceTo(rp) < threshold T_j \) then
            addToList = false;
            break;
        end
    end
    if addToList then
        rendezvousPoints.add(sp);
    end
end
for each \( rp1 \in \) rendezvousPoints do
    for each \( rp2 \in \) rendezvousPoints, \( rp2 \neq rp1 \) do
        if \( rp1.distanceTo(rp2) < threshold T_j \) then
            rendezvousPoints.remove(rp1);
        end
    end
end
Return rendezvousPoints;

**Algorithm 1**: Calculation of rendezvous points.

Now that we have a list of potential rendezvous points, which is the best one? We examined a number of different utilities and combinations thereof: estimated communication range at the rendezvous point, proximity to nearest frontiers, and path cost. Since we want the Relay to follow the Explorer, however, it turned out that the most important consideration is the Explorer’s next choice of frontier. In other words, placing the next rendezvous deep into the next frontier that the Explorer plans to enter, while ensuring that the rendezvous point has a strong communication range, gave the best results. (A large communication range is a desirable characteristic for a rendezvous point since as two robots approach it, they will be able to detect and com-
communicate with one another earlier. Communication range at a particular point can be easily estimated using the communication model described in section 4.2.

More specifically, in our implementation we choose a rendezvous point by considering only a small number of points near the Explorer’s next frontier of choice and choosing the one having highest neighbourTraversals value (since this is the most important junction). If multiple points have equal neighbourTraversals values, we choose the one with the best estimated communication range. The full process is outlined in Algorithm 2.

```
List rendezvousPoints = rvCalculation (Algorithm 1);
List chosenPoints = new List;
Point frontierCentre =
agent.chosenFrontier.getCentre();
int highestDegree = 0;
for each rp ∈ rendezvousPoints do
  if rp.pathCost(frontierCentre) < threshold T3 then
    if rp.degree > highestDegree then
      chosenPoints = new List;
      chosenPoints.add(rp);
      highestDegree = rp.degree;
    end
    else if rp.degree = highestDegree then
      chosenPoints.add(rp);
    end
  end
end
double bestRange = 0;
Point bestPoint = new Point;
for each cp ∈ chosenPoints do
  if CommModel.rangeEstimate(agent.occupancyGrid, cp) > bestRange then
    bestRange = CommModel.rangeEstimate(agent.occupancyGrid, cp);
    bestPoint = cp;
  end
end
Return bestPoint;
```

Algorithm 2: Choosing the best rendezvous point.

### 3.6 Replanning in Real-time and Dynamic Environments

#### Real-time

On a modern computer, selection of a rendezvous point typically takes a few hundred milliseconds (see Table 1). Hillicth thinning, generation of a list of rendezvous points, and selection of the final point take longer as the effort progresses, as there is more free space, a larger skeleton, and longer paths to compute. Choosing a frontier takes less time as the effort progresses since the number of open frontiers decreases.

Exact computation times are highly dependent on the specific approach, and numerous optimisations are possible (for example, considering only a subset of all frontiers when choosing a frontier). Nevertheless, while real search-and-rescue scenarios are likely to involve larger occupancy grids and environments extending across multiple levels, we believe that as robots are equipped with better and better processors, this method would scale well and be possible to compute in real-time.

#### Dynamic Environments

At this point we have an efficient method for calculation of rendezvous points, and each robot has a fairly accurate picture of where its parent relay is likely to be. What happens in dynamic environments? Among the possible problems that arise in dynamic environments, we consider two cases:

1. A parent finds that the path to rendezvous with its child becomes blocked
2. A child finds that the path to rendezvous with its parent becomes blocked

We will look at the case where the parent is a Relay and the child is an Explorer (although interaction between two relays connected in the team hierarchy would be the same).

#### Case 1: Relay cannot reach rendezvous

In the first case, the Relay finds that it cannot reach rendezvous. It recomputes to find the next best rendezvous point (within a maximum distance threshold), reaches this point, and waits, hoping that the Explorer will find it.

The Explorer reaches the originally agreed rendezvous point, and waits. If after a specific amount of time the parent Relay has not arrived, the Explorer must assume that it could not reach rendezvous, and must replan. The Explorer will have stored the map known to its parent Relay at the previous rendezvous between the two. Since a Relay ferries back and forth between rendezvous and the command centre, its map will not have changed significantly. Thus the Explorer can predict which new rendezvous point the Relay is likely to choose (or if a fallback rendezvous point has been agreed on, it can choose this as a target).

Now, the Explorer must reach this new rendezvous point. If it is reachable by path planning on the Explorer’s map, the problem is solved and the Explorer goes to the new rendezvous. If, however, there is currently no path to the new rendezvous, the Explorer continues with frontier exploitation, but significantly favouring frontiers that are closer to the new rendezvous point. This can be integrated into the frontier utility equation in terms of a proximity factor.

This does not guarantee that Relay and Explorer will meet

<table>
<thead>
<tr>
<th>Early in the exploration effort (20% of env. explored)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hilditch thinning</td>
</tr>
<tr>
<td>Generation of rendezvousPoints list</td>
</tr>
<tr>
<td>Choosing a frontier</td>
</tr>
<tr>
<td>Deciding on the exact rendezvous point</td>
</tr>
<tr>
<td><strong>Total</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Late in the exploration effort (80% of env. explored)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hilditch thinning</td>
</tr>
<tr>
<td>Generation of rendezvousPoints list</td>
</tr>
<tr>
<td>Choosing a frontier</td>
</tr>
<tr>
<td>Deciding on the exact rendezvous point</td>
</tr>
<tr>
<td><strong>Total</strong></td>
</tr>
</tbody>
</table>

Table 1: Typical computation times for elements of the rendezvous point selection process in an 800 x 600 occupancy grid involving four robots (two explorers, two relays), using a 2.4GHz, 2GB machine
again (indeed, in some cases a robot may become entirely 
trapped), but it does allow them to make a second attempt.

An alternative solution is for both Relay and Explorer to 
find their way to the next highest point in the communica-
tion chain (the Relay’s parent relay) — we hope to examine 
this scenario in future work.

Case 2: Explorer cannot reach rendezvous
Let’s assume that the Explorer has had his return path to 
the rendezvous point blocked. In this case, the Relay can 
still reach the originally agreed rendezvous point, and waits 
there.

The Explorer, on the other hand, does not have a path to 
the rendezvous. Again, it tries to find an alternative route, 
by favouring frontiers that are closer to the originally agreed 
point.

Hopefully the Explorer finds an alternative route. If not, 
and if a significant amount of time has passed without ren-
dezvous, the Relay can either return up the chain of com-
munication (to its parent), or convert to becoming an explorer 
itself.

4. SIMULATION ENVIRONMENT

4.1 Simulator Framework

To implement our multi-robot exploration approach and 
compare it with other existing approaches, we have de-
veloped our own JAVA-based simulation environment, the Multi-
Robot Exploration Simulator (MRESim). MRESim allows 
for full configuration of environments, either manually or by 
import of binary image. The simulation framework handles 
collisions, sensor data and communication as follows: At 
every time step, the simulation framework requests from each 
agent a new desired location. If the location is valid, the 
agent is moved to this location, and new sensor data is sim-
ulated and sent to the agent. Following the movement of 
all agents, the communication model used to determine 
whether any agents are within range of one another, either 
directly or via multi-hop. If yes, all relevant knowledge of 
the environment is shared between all communicating agents.

At any point a simulation may be paused and agents’ in-
dividual knowledge bases may be examined. This includes 
all known free space, safe space, frontiers, calculated paths, 
communication ranges, map skeleton and rendezvous points.

4.2 Communication Model

We have implemented and tested a variety of communi-
cation models in our simulations. For experiments reported 
here we use a standard path loss model with a wall attenua-
tion factor as described in [4]:

\[
S = P_{t_0} - 10 \times N \times \log_{10}\left(\frac{d_{\text{ref}}}{d_{\text{wall}}^2}\right),
\]

\[
\begin{align*}
& \begin{cases} 
  nW \times \text{WAF} & \quad nW < C \\
  C \times \text{WAF} & \quad nW \geq C 
\end{cases} 
\end{align*}
\]

where \(P_{t_0}\) is the reference signal strength, \(N\) is the path 
loss rate, \(d_{\text{ref}}\) is the distance, \(d_{\text{wall}}\) is the reference distance, 
\(nW\) is the number of obstructing walls, \(\text{WAF}\) is the wall 
attenuation factor and \(C\) is the maximum number of walls to 
consider. This model is widely used in simulation, including 
the popular USARSim simulator [6]. A typical communica-
tion range for an agent is displayed in Figure 5.

\[\text{Figure 5: Typical communication range for an agent using} \]
\[\text{the communication model described in section 4.2} \]

4.3 Noise

Currently, we assume perfect sensor data and localisation. 
We are well aware that this is not realistic and real-world 
systems need to cope with sensor noise and inaccurate maps. 
However, we believe that our results are useful nevertheless, 
because:

\[\begin{itemize}
\item \text{steady advances in robotic mapping are leading to ever} 
\item \text{more accurate mapping techniques} \]
\[\begin{itemize}
\item \text{even with imperfect localisation, the approach is likely} 
\item \text{to work. When rendezvous points with large communi-} 
\item \text{cation range are chosen, teammates do not need to ren-} 
\item \text{dezvous at precise locations, they merely need to en-} 
\item \text{ter the communication range of the rendezvous point,} 
\item \text{which leaves room for error. Once they meet, they can} 
\item \text{relocalise based on one another’s maps. The shared} 
\item \text{map itself does not need to be perfect; it’s more im-} 
\item \text{portant is that both robots share the same frame of} 
\item \text{reference.} \]
\[\begin{itemize}
\item \text{this is an early work examining some of the high-level} 
\item \text{aspects of multi-robot exploration under limited com-} 
\item \text{munication. We hope to take the results from these} 
\item \text{simulations and test the successful methods in more} 
\item \text{noisy, realistic simulation environments (such as US-} 
\item \text{ARSim [6]) in the near future.} \]

5. RESULTS

5.1 Static environment

To examine whether the new method for calculating ren-
dezvous points improves exploration efficiency, we ran three 
algorithms and compared results. The three approaches were:

A. Frontier Exploration.
Normal frontier-based exploration without concern for 
communication range limits. Robots choose frontiers 
according to the method in section 3.4, and don’t re-
turn to the ComStation until the whole environment 
has been explored.

B. Role-based Exploration, simple rendezvous point 
calculation.
Role-based exploration, without the novel method for
rendered calculation. Explorers choose their current location as subsequent rendezvous points.

C. Role-based Exploration, advanced rendezvous point calculation.

Role-based exploration, using the method for rendezvous calculation detailed in section 3.5.

We compared these three approaches in office-like, open, and cluttered environments. We use two performance metrics to compare the methods:

1. Total area explored. We use the union of the area explored by each robot.
2. Total knowledge of the environment at the command centre. Since the goal is to return all information to human responders at the point of entry in a search-and-rescue scenario (and since known information that doesn’t reach human responders is useless), this metric is of particular interest.

The full details of these experiments for runs involving a team of four robots (two explorers, two relays) are presented in Figure 6. Frontier exploration leads to faster coverage of the environment in both office-like and cluttered environments. However, this advantage is not noticeable at the command centre – it is clear that both role-based approaches are significantly better at relaying new information back to the command centre in all three types of environments.

The novel rendezvous point calculation method proposed in this paper leads to significantly more efficient exploration than the previous rendezvous point calculation – the percent knowledge gain is outlined in Table 2.

<table>
<thead>
<tr>
<th>Environment type</th>
<th>% gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Office-like</td>
<td>9.43</td>
</tr>
<tr>
<td>Open</td>
<td>9.64</td>
</tr>
<tr>
<td>Cluttered</td>
<td>1.12</td>
</tr>
</tbody>
</table>

Table 2: Overview of improvement of novel rendezvous point calculation over previous method, in terms of percentage of exploration known at the command centre. More extensive experimental results are presented in Figure 6.

5.2 Dynamic environment

To examine our proposed solution to problems of dynamic environments, we spontaneously let obstacles and walls appear in our testing environments. An example of a ‘Case 1’ situation (Relay cannot reach rendezvous) is presented in Figure 7.

In this case the Relay reverted to the junction point as expected from Algorithm 2. Beta could predict Alpha’s choice and pursued frontiers close to the new rendezvous point. In all of our initial experiments, teammates were able to find one another again after being blocked off. We hope to examine these ideas in more detail in future work.

6. DISCUSSION AND FUTURE WORK

While results presented here are preliminary, the role-based approach and rendezvous point selection methods presented here show promise regarding future robotic applications such as robotic search-and-rescue.

The novel rendezvous point selection method means that role-based exploration almost matches frontier-based exploration in terms of speed of exploration, while significantly outperforming it in terms of returning knowledge to a central location. Most of today’s multi-robot exploration approaches keep team members within range of one another, but for severely communication limited environments, other methods such as these will be crucial.

As robots are likely to be used for exploration of larger and larger areas, methods need to scale up in terms of memory requirements and computation time. The rendezvous selection process presented here is easy to implement and fast, and we believe that it could be applied to multi-level maps as well.

However, much work remains to be done: the noise and localisation assumptions must be relaxed, and more realistic simulations conducted. Experiments need to be conducted with larger numbers of robots. There are numerous potential extensions to the role-based approach, such as the development of dynamic hierarchies or the deployment of RFID tags or motes to aid in the exploration process. We are also interested in the fusion of aerial data (e.g. overhead cameras) and ground data (e.g. rangefinders), and the use of aerial robots to create a communication infrastructure for ground robots. We hope to apply some of the ideas presented here to such scenarios in the future.

7. CONCLUSIONS

For many robotic applications, information about an environment must be gathered by a team of robots and returned to a central location. In robotic search-and-rescue, this corresponds to the human responders’ point of entry. Using a team of robots for such a task brings up problems of team coordination, knowledge sharing, and communication. Particularly in search-and-rescue scenarios, communication can be severely limited and robust strategies must be devised that take this into account.

We have proposed Role-based Exploration, in which robots either explore the deep reaches of the environment, or ferry information from explorers up the communication chain to the central command centre. While purely frontier based methods cover an environment faster, role-based approaches allow for information to reach human responders more quickly and more often.

Explorers and relays must rendezvous to exchange information and the selection of rendezvous points turns out to have a significant effect on the exploration effort. We propose a method from digital image processing, thinning, to skeletonize the map. This skeleton can be used to find an even distribution of possible rendezvous points, including those found at junctions. By careful selection of rendezvous points (close to future exploration areas), exploration becomes significantly more efficient. These rendezvous points can be used for replanning when unexpected changes in the environment occur. This could be of great use in dynamic environments such as those encountered in search-and-rescue scenarios.

While role-based exploration is in an early stage, we believe that certain applications will require explicitly planning for autonomous exploration beyond communication range limits. We hope the ideas presented here are an early step in that direction, and may be used on their own or as extensions of existing approaches in the near future.
8. REFERENCES


(a) Three environments used for testing: office-like, open, and cluttered

(b) Results in an office-like environment: % of environment explored (left) and % known at command centre (right)

(c) Results in an open environment: % of environment explored (left) and % known at command centre (right)

(d) Results in a cluttered environment: % of environment explored (left) and % known at command centre (right)

Figure 6: Evaluation of performance metrics after running algorithms A, B and C in 3 different types of environments
(a) A sudden wall (previously not there) blocks Alpha from reaching the rendezvous point (yellow). Beta waits a specific amount of time for Alpha to appear.

(b) Alpha recalculates, and chooses the rendezvous point having highest degree (a junction point). Beta recalculates, assumes Alpha will head for the same junction point, and pursues frontier exploration with a preference for frontiers near the new rendezvous point.

(c) Alpha and Beta meet at the new rendezvous point and exploration can proceed as normal.

Figure 7: Use of rendezvous points to deal with unexpected obstacles in dynamic environments
Decentralised Coordination of Unmanned Aerial Vehicles for Target Search using the Max-Sum Algorithm

F. M. Delle Fave¹, Z. Xu², A. Rogers¹ and N. R. Jennings¹

¹ University of Southampton, Southampton, SO17 1BJ, UK
{fmdb@acm.ac.uk}@ecs.soton.ac.uk
² Australian Centre for Field Robotics, The University of Sydney, Bldg. J04, Sydney, NSW 2006
zhuxu@cs.uni.sydney.edu.au

ABSTRACT

This paper considers the coordination of a team of Unmanned Aerial Vehicles (UAVs) that are deployed to search for a moving target within a continuous space. We present an online and decentralised coordination mechanism, based on the max-sum algorithm, to address this problem. In doing so, we introduce a novel coordination technique to the field of robotic search, and we extend the max-sum algorithm beyond the much simpler coordination problems to which it has been applied to date. Within a simulation environment, we benchmarked our max-sum algorithm against three other existing approaches for coordinating UAVs. The results showed that coordination with the max sum algorithm outperformed a best response algorithm, which represents the state of the art in the coordination of UAVs for search, by up to 26%. The results further showed that the max-sum algorithm outperformed an implicitly coordinated approach, where the coordination arises from the agents making decisions based on a common belief, by up to 34% and finally a non-coordinated approach by up to 68%.

1. INTRODUCTION

In recent years, Multi-Agent Systems (MAS) research has started to focus on coordination mechanisms for teams of agents, in which the agents represent robotic platforms, such as Unmanned Aerial Vehicles (UAVs), that are deployed for information gathering tasks, such as searching for a moving target [12]. Within these domains, a decentralised approach to coordination is often favoured due to its robustness to failures of individual agents [4], the potential presence of complex topologies [5], the absence of a single point of failure [12], scalability and modularity [6]. However, decentralisation introduces two key challenges: (i) how to fuse information from the UAVs in order to maintain a common belief about the state of the target, and (ii) how to coordinate the motion of the UAVs in order to collect the most informative observations and avoid redundant coverage of the environment. The first of these challenges is commonly known as the Decentralised Data Fusion (DDF) problem and has received extensive attention in the data fusion literature [2, 6]. The second of these challenges has received less attention, and is thus the topic of this paper.

Typically, the coordination must also be performed within a highly dynamic environment; the most evident reason for which is the dynamic motion of the target and the UAVs. For this reason, offline approaches, such as pre-computing the paths for the whole team, are not feasible, and the coordination must be performed online in real-time. Now, an online and decentralised coordination mechanism, based on the max-sum algorithm, has recently been demonstrated to be effective on a number of benchmark problems, including graph-colouring [7] and the coordination of mobile sensors that were constrained to move within a graph representing an indoor environment [12]. However, to date, such approaches have not been applied to the more challenging problem of coordinating the paths of UAVs that move in an unconstrained continuous space.

Thus, it is this shortcoming that we address in this paper. More specifically, we present a study of how the max-sum algorithm can be applied to the coordination of UAVs tasked to search for a target in a continuous space. We benchmarked our approach against an adaption of the best response algorithm used in [3] and against two other approaches to coordination found in the literature: a non-coordinated approach and an implicitly coordinated approach where the agents make decisions based on a shared understanding of the environment [10]. Our empirical results showed that the max-sum approach outperformed all the aforementioned approaches.

Thus, against this background, this paper makes the following contributions to the state of the art:

- For the first time, we apply the max-sum algorithm to the challenging task of coordinating a team of UAVs tasked to search for a target in a continuous space. By doing this, we introduce a novel coordination technique to the field of robotic search, and we extend the max-sum algorithm beyond the simpler coordination problems to which it has been applied to date [7, 12].

- We benchmarked the max-sum coordination algorithm against three existing approaches that have been proposed for coordinating UAVs for search and showed that it out-performed the explicitly coordinated approach based on a best response algorithm [3] by up to 26%, the implicitly coordinated approach by up to 34% and finally a non-coordinated approach by up to 68%.

The remainder of this paper is organised as follows. In Section 2 we analyse the relevant literature that motivated our study of this problem. In Section 3 we formulate the search task as a multi-agent coordination problem. In Section 4 we define the coordination framework that we built in order to address the coordination problem. We then empirically compare the coordination approaches, and present
the results in Section 5. Finally, we conclude in Section 6, and mention some ongoing improvements to this work.

2. RELATED WORK

Significant contributions towards solving the decentralised coordination problem have been made in both the MAS and robotics communities. Approaches to coordinating a multi-agent system were classified into three different levels in [10]:

- A non-coordinated approach where the agents do not share messages or otherwise take into account the actions of other agents in the system.

- An implicitly coordinated approach where the agents share their observations, but make individual decisions on what action to take next. This is also known as a coordinated approach [6, 10].

- An explicitly coordinated approach where the agents share both observations and predictions of what they expect to gain in future observations. This is also known as a cooperative approach [6, 10].

This framework has been widely used in the coordination literature in both the MAS [11] and robotics [6] communities, and is also used in this paper.

Previous work from the robotics community has considered a search problem similar to the one studied here and proposed a decentralised Bayesian solution to the DDF problem. While the approach to coordination in the aforementioned work was limited to implicit coordination, the work was extended to consider explicit coordination by formulating the problem as a distributed optimisation problem [3]. This was iteratively solved by using either a Jacobi or Gauss-Seidel type algorithm, where at every iteration, each individual calculates its best response, given the previously communicated best responses of other UAVs and the expected impact on the environment associated with these actions. Once calculated, every individual communicates its new best response, and its expected impact on the environment and the cycle starts again. As acknowledged by the author, this approach is susceptible to converging to a local optimum, rather than the global optimum. For the purposes of this paper, this approach is referred to as the best response algorithm.

A variation of this approach was used in another study on the use of UAVs for search and track [6], where each agent calculates its best response based on all previously communicated best responses of other agents and their expected impacts on the environment, but in this case, the algorithm does not iterate. While it was acknowledged that this approach is sub-optimal, it was argued that this was necessary in a highly dynamic environment, since in the time required for the iteration to take place, the environment would have changed, rendering the iterated solution sub-optimal [6].

On the other hand, work from the MAS community showed that the max-sum algorithm could out-perform best-response algorithms for coordination [7], albeit on less complex problems. The max-sum algorithm is an approximate message passing algorithm, where every agent tries to find the best joint control action by negotiating with its neighbouring agents. The max-sum algorithm is an application of the generalised distributive law, a class of message passing algorithms [1], and has been well used and studied in the field of information theory and for coordinating mobile sensors [11, 12].

The application of the max-sum algorithm to the coordination of mobile sensors to monitor spatial phenomena [11, 12] was particularly interesting, as it demonstrated that the max-sum algorithm is applicable to more complex problems than described in [7]. Additionally, a number of similarities between the search problem and the spatial phenomena monitoring problem were noted, such as the need to cooperatively explore an environment.

These similarities motivate the study of the applicability of the max-sum algorithm to the online, decentralised coordination of UAVs for search. As a matter of fact, the max-sum algorithm presents a set of features that make it attractive for this problem [7, 12]. The algorithm is decentralised and allows multiple agents to negotiate locally over a function to optimise. The topology of the interactions is modelled as a factor graph, a particular type of bipartite graph that will be presented in more detail in Section 4.2.4. The algorithm is able to scale up to a high number of agents because it exploits the neighbouring interactions among these agents. When the factor graph is acyclic, the max-sum algorithm computes the optimal result, and also has the advantage of being any-time. When the factor graph is cyclic, empirical evidence shows that the max-sum algorithm still performs well [7].

3. THE TARGET SEARCH PROBLEM

In this section, we introduce the model of the search task. Fundamentally, the problem is to coordinate the motion, and hence observations, made by a team of UAVs so as to search for a target in a timely manner. We define a continuous search area $A$ where the UAVs operate to search for the target. Each UAV maintains an internal representation of the area $A$ by discretising the area into a grid $G$ with a resolution depending on the setting of the problem. Each cell of the grid $G$ represents a specific rectangular area within the search area. We model the continuous time system using small, discrete time steps, and assume that all the UAVs have time synchronisation (e.g. through GPS time).

We define the search task by modeling the motion of the agents, the target and the UAVs, and the sensor model. First, we outline the motion model of the two classes of agents, the UAVs and the target. Secondly, we introduce the sensor model. In this work, it is assumed that each UAV uses a fixed, downward pointing camera to detect the target.

3.1 Agent Motion Model

The scenario mentioned in the previous section suggests two types of agents, the UAVs and the target. In the following sections, we will first describe the motion model of the target, and then of the UAVs.

3.1.1 Target Motion Model

The target moves following a simple probabilistic Markov motion model. The state of the target at time $k$ is defined as $x_t^k = (i, j)$, where $(i, j)$ are the coordinates of the grid cell that contains the target. The probability of the target transitioning to another cell is modelled as:

$$P \left( x_{k+1}^T | x_k^T \right) = \frac{1}{|A|}$$

(1)
where $A_D(x_k^T)$ is the set of cells adjacent to $x_k^T$ as well as $x_k^T$ itself.

3.1.2 UAV Motion Model

The team of UAVs is formally defined as a set $S$ of agents. Every UAV has the following kinematic motion model:

$$
\begin{align*}
\dot{x} &= V \cos \psi \\
\dot{y} &= V \sin \psi \\
\dot{z} &= 0 \\
\dot{\psi} &= \frac{g \tan \phi}{V}
\end{align*}$$

where $V$ is the UAV velocity, $g$ is the acceleration due to gravity, $\psi$ is the UAV heading and $\phi$ is the UAV bank angle, which is limited to some maximum value $|\phi| \leq \phi_{max}$. We assume that the velocity of the UAV remains constant at the cruise speed, which was taken as 25m/s, and that the maximum bank angle is 25 degrees.

3.2 Sensor Model

Each UAV uses a fixed, downward pointing camera to detect the target. The camera is assumed to capture one frame per second. The primary interest in the sensor model is to characterise the footprint of the camera, as well as the probability of detecting a target within that footprint.

Formally, we denote sensor observations by the $i^{th}$ UAV at the $k^{th}$ time step as $z_k^i$, where $z_k^i$ can take on one of two values, $D_k^i$, representing a target detection event, or $\hat{D}_k^i$, representing a no-detection event. We further define $z_k$ as the net observations by all UAVs.

Moreover, we define a matrix $o_k^i$, where the $(i, j)^{th}$ element, denoted by $o_k^i (i, j)$, represents the probability of the sensor on UAV $i$ not detecting the target, conditional on the target being at the $(i, j)^{th}$ cell:

$$o_k^i (i, j) = P \left( z_k^i = D_k^i | x_k^T \right)$$

Naturally, $P \left( z_k^i = D_k^i | x_k^T \right) = 1 - P \left( z_k^i = \hat{D}_k^i | x_k^T \right)$.

In order to model $P \left( z_k^i = D_k^i | x_k^T \right)$, we first characterised the footprint of the camera, which was modeled as a pin-hole camera [14]. As a consequence of this, the footprint can be easily computed by making a flat Earth assumption and by simple geometric arguments. Figure 1 shows an example of a camera footprint.

When the quadrilateral defined by the points $P_i^k$ for $i \in [1, 4]$ in Figure 1 is overlaid onto the grid $G$, the probability of detecting the target, that is $P \left( z_k^i = D_k^i | x_k^T \right)$, is assumed to be linearly proportional to the ratio of the area of the cell covered by the quadrilateral to the total area of the cell, multiplied by a term $\alpha$ that models the range-dependent characteristics of the sensor. In this case, the range dependent characteristics were modeled as:

$$\alpha = \exp \left( -\frac{R}{R_0} \right)$$

Figure 1: Illustration of camera footprint.\(^1\)

Where $R$ is the range from the sensor to the cell in question and $R_0$ is a constant term that was tuned to model the range-dependency of the sensor. Therefore, the value of $P \left( z_k^i = D_k^i | x_k^T \right)$ for each cell in the grid $G$ has a value varying from 0, when the cell is not within the footprint and $\alpha$ when the cell is completely covered by the footprint. It can be noted that this model accounts only for false negatives, where the sensor fails to detect a target that is present in the sensor field of view, but not for false positives, where the sensor reports a detection when the target is not present in the field of view. This model can be justified if the sensor detection characteristics are tuned conservatively.

The UAVs share these observations with each other to maintain a consistent belief of the distribution over the state of the target across the UAVs. Assuming that the observations by the UAV sensors are conditionally independent, then

$$P \left( z_k = D_k | x_k^T \right) = \prod_{i=1}^{S} P \left( z_k^i = D_k^i | x_k^T \right)$$

where $\tilde{D}_k = D_k^1 \cap \ldots \cap D_k^S$. Defining $o_k (i, j) = P \left( z_k = \tilde{D}_k | x_k^T \right)$, the above equation means that:

$$o_k = \prod_{i=1}^{S} o_k^i$$

4. OUR COORDINATION SOLUTION

In this section, we outline the framework for coordinating the sensor platforms such that they can collectively search for a target. We first introduce the data fusion methodology, based on the Bayesian formulation in [2], and then the coordination approach we used.

4.1 Bayesian Estimation

In this work, the probabilistic belief of the target’s position over the grid $G$ is defined as a matrix $P_k^{G^T}$, with each element of the matrix representing the probability of the target being in the corresponding cell in $G$ at time $k$:

$$P_k^{G^T} (i, j) = P \left( x_k^T = (i, j) \right)$$

\(^1\)The authors would like to acknowledge V Scordamaglia's MATLAB function "Trajectory and Attitude Plot Version T" as the source of the aircraft model used in a number of illustrations in this document. It can be obtained from http://www.mathworks.com/matlabcentral/fileexchange/4572-trajectory-and-attitude-plot-version-2
The estimation process involves two steps: the update step that fuses observations into the belief, and the prediction step that propagates the belief to account for the dynamic nature of the target. These steps are described further in the following sub-sections.

A series of snapshots showing the change in the distribution over the state of the target as the estimation process is carried out is shown in Figure 4.

### 4.1.1 Update

We adopted the same Bayesian update equation as used in previous work [3, 8] to fuse the observations made by the UAVs into their belief of the state of the target. Thus,

$$
P(\hat{z}_{k+1}^T|z_k, \ldots, z_1) = \frac{1}{C_1} P(\hat{z}_{k+1}^T|z_k, \ldots, z_1) P(z_{k+1}^T|z_{k+1}^T)$$

where $C_1$ is a normalising constant to ensure that the probability distribution function (PDF) integrates to unity, and is equal to $P(z_{k+1}^T|z_k, \ldots, z_1)$. Here, $P(z_{k+1}^T|z_k^T)$ is the sensor model, and takes on the value of $P(z_k^T = D_k^T|z_k^T) = 1 - o_k$ when the target is detected, and $P(z_k^T = \hat{D}_k^T|z_k^T) = o_k$ in the case the target is not detected.

Now, assuming conditional independence of the observations by the UAVs, we have:

$$P(\hat{z}_{k+1}^T|z_k, \ldots, z_1) = \frac{1}{C_2} P(\hat{z}_{k+1}^T|z_k, \ldots, z_1) P(z_{k+1}^T|z_{k+1}^T)$$

Again, $C_2$ is a normalising constant. An illustration of the Bayesian update step is shown in Figure 2.

### 4.1.2 Prediction

The Bayesian prediction step is used to estimate the target’s current state considering the target’s motion model and the belief on its previous state, before incorporating the new observations. As with the Bayesian update equation, we adopted the prediction equation used in previous work [3, 8], which is based on the Chapman-Kolmogorov theorem [9]. Adaptation to this problem gives:

$$P(\hat{z}_{k+1}^T|z_k, \ldots, z_1) = \int P(\hat{z}_{k+1}^T|z_k^T) P(z_k|z_k, \ldots, z_1) dz_k^T$$

Here, $P(\hat{z}_{k+1}^T|z_k^T)$ is the target probabilistic motion model described previously.

### 4.2 Coordination

In this section, we define the approach that was applied to the cooperative search problem. We first introduce the control space for the UAVs and the concept of receding horizon control. We then define the utility function and finally present the max-sum approach for explicit coordination.

#### 4.2.1 Control Space

We assume that there exists a discrete set of pre-computed, dynamically feasible trajectories that can be followed by the UAVs, from which the coordination strategies can choose. This approach of discretising the action space of a robotic platform has been widely used in the robotics community, with a famous example being the online path planner on Stanley, the robot which won the DARPA Grand Challenge [13]. In this work, this set of pre-computed trajectories is calculated based on a set of nominal bank angles, in this case $[-25^\circ, -5^\circ, 0^\circ, 5^\circ, 25^\circ]$. These were chosen as they give a good spread in the resulting control actions, which is illustrated in Figure 3.

From this description of the control space, it should be clear that the future observations of a UAV can be predicted for each member of the control space. Since each member is defined by a path and bank angle, the camera footprints, and hence the sensor model, can be predicted before taking the
4.2.2 Receding Horizon Control

A number of factors mean that it is difficult or impossible to determine the control actions that the UAVs should take to find the target in the shortest time to optimality. These include:

- Imperfect models of the sensor characteristics and UAV motion and control
- The dynamic nature of the environment, as manifested in the ever-changing PDF representing the belief of the state of the target. This means that in the time it takes to compute the optimal control action, the state of the world has changed, possibly rendering the computed action sub-optimal.
- The large search space, defined by the combinations of the control spaces of each UAV over the entire mission

To compensate for these factors, receding horizon control is used to approximate the optimal solution. Receding horizon control was selected as it is a common technique used in the literature to give computationally tractable solutions given imperfect models of the world and a dynamic environment.

4.2.3 Utility Function

The global utility function used in this work, which sums up the performance of the UAV team, is a function of the cumulative probability of detecting of the target, that is, the probability of detecting the target given all the observations made, and borrows from previous work on coordinating UAVs for search [3]. Over a prediction horizon of $N$ steps and for a control action $u$, the utility function is defined as:

$$ J(u, N) = P(D_{1:k}) - P(D_{1:k+N}) $$

![Figure 4: A series of snapshots showing the changes in the probability distribution over the state of the target as the UAVs search. The height of the distribution is exaggerated for visualisation purposes.][1]

![Figure 5: Receding horizon control][2]
Here,  
\[ P(D_{1:n}) = \prod_{i=1}^{n} P(z_i = D|z_{i-1}, ..., z_1) \]  
(17)

It can be seen that Equation 17 can be evaluated by calculating the cumulative product of the normalisation constants in the Bayesian update equation. To calculate utilities for each member of the control space, the observations that the UAV was predicted to make were fused into a copy of the PDF of the state of the target maintained by the UAV. As noted in the work which introduced this utility function, it attempts to maximise the increase in the cumulative probability of detection [3]. This utility function was selected as it is already established in the search and track literature [3].

Hence, the goal of the system is to find the joint control that maximises the global utility - for a detailed definition of how such joint paths are computed, refer to Section 4.2.1. However, the computation of the global utility for a joint control is not trivial, as the actions of one UAV may affect the utility of another UAV. This occurs when the sensor footprints of two UAVs overlap, as shown in Figure 2.

In order to apply the max-sum algorithm, the global utility function must be the sum of the individual contributions of each UAV in the team. To address this requirement, we decomposed the global utility function into the utilities of the individual UAVs using the concept of incremental utilities [11]. This approach consists of establishing an ordering of the agents in the team by assigning each agent a unique ID number. In the context of UAVs, this could be the tail number of UAV. The individual utility of UAV i is then defined as the incremental increase in the global utility function due to the predicted action of UAV i, considering the predicted actions of all UAVs j where j < i. It should be noted that the ordering chosen for the UAVs does not impact on the value of the calculated team utility.

As an example, consider the utilities shown in Figure 2. The incremental utility of UAV (b) is calculated by taking into account the value of its own observations and those made by UAV (a). Specifically, the incremental utility of UAV (b) is calculated by subtracting the value in Figure 2 (a) from the value in Figure 2 (c), namely, 0.5694 − 0.5192 = 0.0502.

4.2.4 Explicit Coordination with Max-Sum

The max-sum algorithm is an approximate message passing algorithm, where every agent tries to find the best joint control action by negotiating with its neighbouring agents. For completeness, we present a brief description of the max-sum algorithm here. For a more sound and complete description, refer to [7]. The max-sum algorithm operates over a factor graph, a particular type of bipartite graph, containing two types of nodes, “variables” and “functions”. Each variable node is connected to a subset of the function nodes, while each function node is connected to a subset of the variable nodes.

In order to apply the max-sum algorithm to our multiagent framework, a variable node \( p_n \) and a function node \( U_n \) were defined for each UAV \( n \). Each variable node \( p_n \) represents the possible trajectories that the \( n^{th} \) UAV can take, as defined in Section 4.2.1. Each function node represents the individual utility of the \( n^{th} \) UAV, as defined in Section 4.2.3. The edges of the factor graph were computed dynamically, by building connections between function and variable nodes. Each variable node \( p_n \) was always connected to the corresponding \( U_n \) and vice-versa, as the control action by the \( n^{th} \) UAV always affects its utility. Every time the UAVs negotiate, the function node \( U_n \), owned by UAV \( n \), is connected to a subset of variables \( p_m \), owned by a subset of UAVs \( m \). To belong to \( m \), a UAV must have lower ID than \( n \), due to the way in which the individual utilities of the UAVs were defined in Section 4.2.3. Additionally, the predicted observations made by the UAV must overlap with the predicted observations of UAV \( n \).

Whether two predicted observations overlapped was determined by evaluating the utility of the two individual predicted observations, and the joint utility of both predicted observations together. If the two individual utilities summed to the joint utility, then the two actions were considered additive, and hence independent, and the variable node of the other UAV was not connected to the function node \( U_n \). Otherwise, the utilities were sub-additive, and the variable node was connected to \( U_n \). An example of a factor graph built following this procedure can be found in Figure 6.

We chose to use this utility-based method of determining whether predicted observations overlapped over geometrically-based methods as this makes the source of the predicted observations anonymous. By this, we mean that the UAV receiving the predicted observations does not need to know about the sensing model of the UAV that transmitted it, as the sensing model is already encoded in the predicted observation. On the other hand, if a geometrically based method was used, the receiving UAV would need to have knowledge of the sensor model of the UAV sending the predicted observation, which would restrict its applicability to homogeneous systems. However, it is acknowledged that a limitation of this approach is that it scales linearly with the number of UAVs in the system, as each UAV needs to consider the predicted observations of every other UAV. Having said this, it would be trivial to introduce domain-specific heuristics that allow a UAV to reject predicted observations that clearly do not overlap with its predicted observations, without having to calculate the utilities. This would mean that a UAV would only need to calculate the utilities for a subset of the other UAVs, thereby improving the scalability. One such heuristic that could be applied in this situation is a threshold on the Euclidean distance between the UAVs. If another UAV is sufficiently far away so that its footprint will not overlap, then its predicted observations can be safely ignored.

Once the factor graph was computed, negotiation started. During negotiations, function and variable nodes each sends a different type of message:

- From variable to function:

\[ Q_{p_n \rightarrow U_n}(p_n) = \sum_{U_n \in A(d(p_n)) \setminus U_n \in \cap U_n} R_{p_n \rightarrow p_m}(p_m) \]  
(18)
5. EXPERIMENTS

5.1 Benchmark Algorithms

In other words, had there been a target in the search area, there would have been a 95\% probability of detecting it in this time. At time \( k \), this probability was calculated by \( 1 - P(D_{t+k}) \). As noted previously, \( P(D_{t+k}) \) is the probability of not detecting the target up to time \( k \), based on the observations by the team of UAVs up to time \( k \).

This section first presents a description of other approaches to coordination against which the max-sum algorithm was benchmarked in Section 5.1. Following this, the hypotheses themselves and the experimental methodology used to test these hypotheses are described in Section 5.2. Finally, the results of the experiments are presented and discussed in Sections 5.3 and 5.4.

5.1 Benchmark Algorithms

In this section, we outline the three approaches to coordination that the max-sum algorithm was benchmarked against. These approaches are classified into the three levels of coordination described in [10].

Non-coordination: In the non-coordinated approach, each UAV selects its control to optimise the utility function described in Section 4.2.3 over a given horizon, independently of the other UAVs. This optimisation occurs on the basis of different PDFs on the state of the target, since the UAVs also do not share observations.

Implicit Coordination: In the implicitly coordinated approach, each UAV selects its control to optimise the utility function described in Section 4.2.3 over a given horizon, independently of the other UAVs. In this case, the UAVs communicate observations so that each UAV maintains the same belief of the state of the target, and makes decisions based on this shared belief. In this case, the implicit coordination arises because each UAV is making its decision based on a common prior belief of the state of the target.

Explicit Coordination with Best Response: In the explicitly coordinated approach, the UAVs make a team decision based on both the common prior information, as well as the predicted observations communicated by other UAVs. The best response algorithm is an example of an explicit coordination algorithm that is the state of the art for the coordination of UAVs for search [3, 6]. It is for this reason that this paper benchmarks the performance of the max-sum algorithm against the performance of the best response algorithm. The best response algorithm operates by having every UAV determine the best control action it can choose, given its belief, and given its knowledge about the control actions that the other UAVs of the team are going to take, based on what the other UAVs have previously communicated. Every UAV then broadcasts its new decision, and the cycle is repeated.

In the previous application of the best response algorithm for coordinating UAVs, two termination criteria were defined, a theoretical one, where the procedure was iterated until the best joint control action was found, and a practical one where the procedure iterated until the solution converged to a given threshold [3]. In our case, to be able to compare this approach to the max-sum algorithm, we fixed the number of iterations that both the max-sum algorithm and the best
response algorithms were allowed to go through before termination. For the simulations, this was fixed at six iterations. Additionally, while the previous work optimised over a continuous control domain [3], the best response algorithm was only allowed to optimise over the discrete control domain available to the max-sum algorithm in this work.

In terms of the coordination overhead, the best response algorithm requires slightly less communication than the max-sum algorithm. This is because once the predicted observations are communicated, the best response algorithm only communicates the index representing its best response at each iteration. On the other hand, the max-sum algorithm needs to communicate the variable to function and function to variable messages, which are vectors of length equal to the number of possible values each variable can take on, which is five in these simulations, there being five control actions in the control space.

5.2 Methodology
This section describes the methodology that was used to test the following hypotheses in simulation:

- Teams of UAVs that were explicitly coordinating using the max-sum algorithm would out-perform teams of the same size that were explicitly coordinating by using the best response algorithm, which would in turn out-perform teams that were implicitly coordinating, which would out-perform teams that were not coordinated. Naturally, a shorter time taken to obtain a 95% cumulative probability of detecting the target meant that a team had performed better.
- A single UAV would perform equally well for all levels of coordination.

First, the methodology is outlined. This is followed by a listing of the controlled variables in the experimental design. Lastly, the experimental variables being tested are presented.

In recognition of the possibility that the initial positions of the UAVs would influence the result, ten sets of random initial positions were generated for teams of one, two and five UAVs. Each team then flew a simulated search mission using each of the four types of coordination, starting from each of the ten initial positions. For each simulated mission, the time to achieve a 95% confidence in detecting the target was recorded. After all 120 simulated missions were complete, the mission times for each team size/level of coordination combination was averaged across the ten initial positions. This gave average mission times for each team size/level of coordination combination. Finally, for each team size, a Student's t-test with a 95% confidence interval was applied to determine if the differences between the average times for each type of coordination were statistically significant.

5.2.1 Controlled Variables
The controlled variables in this experiment were:

- The characteristics of the UAV (e.g. cruise speed, control space, initial position)
- The characteristics of the sensors used to detect the target (e.g. field of view, the $R_0$ value used to model the range-dependent error characteristics).
- The data fusion method and implementation used to fuse observations from the sensors on-board UAVs, observations received from other UAVs, as well as the prior belief on the state of the target.
- The prediction horizon over which the utility function was evaluated

5.2.2 Experimental Variables
The experimental variables used in this work were:

- The three team sizes
- The four levels of coordination (non coordinated, implicitly coordinated, explicitly coordinated using the best response algorithm and using the max-sum algorithm)

5.3 Results
The results obtained from the simulated missions are illustrated in Figure 7. The differences between the times for the different types of coordination in the two and five UAV teams are evident. A Student's t test showed that the differences in the performance of the max-sum algorithm compared to the best response algorithm, implicit coordination and no coordination were statistically significant. Specifically, these results illustrated that for a team of two UAVs, explicit coordination with the Max-Sum algorithm reduced mission times by 7% compared with explicit coordination with the best response algorithm, by 17% compared with implicit coordination, and by 48% compared with no coordination. In a team of five UAVs, explicit coordination with the max-sum algorithm reduced mission times by 26% compared against explicit coordination with the best response algorithm, by 34% compared against implicit coordination, and 68% when compared against no coordination. On the other hand, the differences that can be observed in the one UAV team are much smaller, and a Student's t test showed that these differences were statistically insignificant. This self-evident result verified that no level of coordination was unfairly advantaged or disadvantaged.

5.4 Discussion
The simulation results showed that explicit coordination using the max-sum algorithm out-performed the best response algorithm, implicit coordination and no coordination. The simulation results further showed that explicit coordination using the best response algorithm outperformed implicit and no coordination. The simulation results finally showed that implicit coordination out-performed no coordination for both teams of 2 and 5 UAVs. From these observations, we can see that all of the hypotheses outlined previously were verified.

The results also showed that by explicitly coordinating with the max-sum algorithm, the performance relative to other methods of coordination was not affected by the size of the team, except for the degenerate case of a single UAV team. Hence, it can be asserted that the max-sum algorithm is a valid approach to inducing explicit coordination in a team of robotic platforms.

6. CONCLUSIONS
In this paper, we applied the max-sum algorithm, developed in [7], to coordinate a team of UAVs to search for a dynamic
target. We benchmarked the max-sum algorithm against three other types of coordination approaches, namely the best response algorithm [3], representing the state of the art in the coordination of UAVs for search, the implicitly coordinated approach where agents make decisions individually, based on common information, and the non-coordinated approach, where the UAVs do not share any information at all. To compare the performance of the different approaches, we measured the average time taken for a team of UAVs to obtain a given confidence of detecting the target. By doing this, we showed that coordination with the max-sum algorithm out-performed the best response algorithm by 26%, implicitly coordinated approach by 33% and the non-coordinated approach by 68% in the case of five UAVs, and by 7%, 17% and 48% respectively for teams of two UAVs.

These results indicate that the max-sum algorithm has potential to be applied in complex systems operating in dynamic environments. Hence, the future of this work is to test the application of max-sum algorithm for coordination in high-fidelity simulations, such as Software-in-the-Loop and Hardware-in-the-Loop simulations and finally, through a flight demonstration.

7. ACKNOWLEDGMENTS

The work reported in this paper was funded as part of the ALADDIN (Autonomous Learning Agents for Decentralised Data and Information Systems) Project and is jointly funded by a BAE Systems and EPSRC (Engineering and Physical Research Council) strategic partnership [EP/C548051/1], and also by the Systems Engineering for Autonomous Systems (SEAS) Defence Technology Centre established by the UK Ministry of Defence. The Xu is supported by the ARC Centre of Excellence programme, funded by the Australian Research Council (ARC) and the New South Wales State Government.

8. REFERENCES


Learning Opponent’s Strategies in the RoboCup Small Size League

Felipe W. Trevizan
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA, USA
fwt@cs.cmu.edu

Manuela M. Veloso
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA, USA
veloso@cs.cmu.edu

ABSTRACT
One of the machine learning challenges posed by the robot soccer domain is to learn the opponents strategies. A team that may be able to do it efficiently may have the advantage to adapt its own strategy as a response to the opponent’s strategy. In this work, we propose a similarity function to compare two teams, and consequently their strategies, by the ability of one team to mimic the behavior of the other. The proposed function can be used to classify opponents as well as to decompose an unknown opponent as a combination of known opponents. We apply the proposed function to classify opponent’s defense strategies in real world data from the RoboCup Small Size League collected during the RoboCup 2007, RoboCup 2008 and USOpen 2009. We also use this similarity function to discover patterns in the logs of these championships, such as, similar teams and the number of major defense strategies.

Categories and Subject Descriptors
I.2.6 [Computing Methodologies]: Artificial Intelligence—Learning

General Terms
Experimentation

Keywords
Machine Learning, RoboCup, Small Size League

1. INTRODUCTION
A multi-agent, dynamic and adversarial domain offers several challenges for machine learning, for instance, learning how the environment evolves and how the adversary behaviors. One example of such domain is robot soccer, in special the RoboCup Small Size League (SSL) [10, 11].

The SSL consists of two teams, each one with five robots, that play robot soccer on a field of 6 by 4 meters with global overhead perception and control (Figure 1). Also, the robots must conform to the specifications about their size and shape and they are equipped with kicking devices. The main difference between SSL and the other RoboCup robot

Cite as: Learning Opponent’s Strategies in the RoboCup Small Size League, Felipe Trevizan and Manuela Veloso. Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010), van der Hoek, Kaminka, Lopesarc, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. XXX-XXX.
Copyright © 2010, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

Figure 1: The CMDragons robots (built by Michael Licitra) posed and in a game.

soccer leagues are: (i) the allowed use of cameras placed over the field, for shared global perception; and (ii) the allowance of a centralized computer to coordinate the robots, therefore the overall team is autonomous. For this work, we use the logs captured by the CMDragons team [4], the SSL team of Carnegie Mellon University, during 3 championships: RoboCup 2007, RoboCup 2008 and USOpen 2009.

Our approach to classify opponent’s strategies uses features extracted from the game logs, such as distance from the ball to the robots and from the CMDragons goal to the robots, and is composed by two steps: (i) segmentation of games into episodes; and (ii) the comparison of episodes.1

In the first step, we segment games in a set of small time series called episodes. Each episode encompasses a defense attempted by the opponents and it is obtained by selecting the time intervals in which the game logs registered the employment of an attack strategy by CMDragons. This procedure assumes that when a team is attacking, the opponent’s response is to employ a defense strategy.

In the second step of our approach, we compare two sets of episodes A and B by representing them as matrices $E^A$ and $E^B$ and computing the error of expressing $E^A$ using only a conical combination of the column of $E^B$. Formally, we compute $d = \min_W \| E^A - E^B W \|_F^2$ such that $w_{ij} \geq 0$, where $\| \cdot \|_F$ is the Frobenius norm, and define $d$ as a (non-symmetric) similarity measure from A to B. This measurement can be seen as the ability of team B to mimic the

1In this work we focus in the defense strategies, however all the techniques developed can be directly applied to analyze attack strategies.
behavior of team A. We can also use the obtained matrix $W$ in the computation of $d$ to explain the behavior of $A$ as a function of $B$. This is specially interesting when $F^A$ is the set of episodes generated by a new and unknown opponent and $F^B$ is the set of all the episodes seen so far. Then $W$ represents a decomposition of the unknown opponent's strategy as a function of the strategies already known.

The remainder of this paper is organized as follows. In Section 2 we discuss the previous approaches for learning opponent's strategies. In Section 3 we present the data set used in this work. Our approach to learn defense strategies based in the previous games is developed in Sections 4 and 5. In Section 6 we test our proposed approach by presenting a set of experiments, involving classification and pattern discovery. Section 7 brings a few conclusion remarks and future research directions.

2. RELATED WORK

Using logs of the RoboCup Simulation Soccer League, Visser and Weland [15] tackle a similar problem: classify the behavior of the goalkeeper and the pass behavior of the opponent players. Their approach uses decision trees to label non-overlapping intervals of a given time series. For instance, in the goalkeeper experiments the labels used are: the goalkeeper backs up, the goalkeeper stays in the goal and the goalkeeper leaves the goal. Using the same technique and a different set of labels, they also analyze the pass behavior of the opponent player.

Another work in the simulation soccer is given by Fard et al. [6]. They proposed an approach to learn opponent's strategy that relies on modeling the opponent as an automaton. This automaton is learned for each previously played opponent by using a predefined payoff matrix, that is designed by an expert, and solving a Prisoner's Dilemma game instance. This payoff matrix is defined through high-level features, such as intercept, pass, shoot and dribble and relate the payoff of playing one of this simple strategies with the opponent response (also represented using the same simple strategies). The main limitation of this approach is the assumption that the opponent plays a fixed deterministic strategy.

For SSL, an alternative approach is given by Bowling et al. [2] which does not model the opponent's behaviour. Instead, their approach to adapt to the opponent is based on the outcome of the attack strategies employed so far. Although this technique has had success when employed in the real games of the SSL, it does not consider previous games against different teams. That is, the authors do not provide a method to relate two teams that play similarly in order to reuse the learned responses.

A more similar approach to the one proposed in this paper is given by Riley and Veloso [12]. This approach, which is applied to simulation soccer, uses a discretization of the observed features, for instance the position of the robots and the ball, and decision trees to classify opponents. The limitation of this method relies on the usage of decision trees since decision trees only offer hard classification, that is, each sample $s$ is classified as belonging to exactly one class as opposed to returning a confidence bound of $s$ being from class $c$, for every class $c$. This implies that an unknown opponent is classified as exactly one of the previously seen opponents instead of describing this opponent as a function of the previous opponents.

Another work related to SSL and pattern recognition is given by Vail and Veloso [14]. Instead of classifying opponent's strategies, they focus on the problem of activity recognition. More specifically, a framework using conditional random fields, a temporal probabilistic graphical model, is developed to classify robots by a set of predefined roles, including attacker, marker and defender.

Similar to [14], Ball and Wyeth [1] classify the roles of each opponent and instead of using conditional random fields, a naive bayes classifier is applied. Their experiments consist in classifying the roles of the robots of RoboRoos, a SSL team that the authors had access to the ground truth roles. The authors also suggest a method to classify opponent teams by adding a layer to their system that builds a model of the opponent team based on the empirical probability distribution of the roles of each opponent robot. No experiment is provided for team classification.

3. THE DATA SET

The data set used in this work is the collection of 13 games played by the CMDragons team during the RoboCup 2007, RoboCup 2008 and US Open 2009. Each logged game is a multivariate time series in which a new data point encompasses an interval of 1/60 seconds. These time series represent CMDragons perspective of the game, for instance, the position of the robots in the data set was obtained by the vision system of CMDragons and is noisy [3]. The number of variables in the time series is 198 and, in this work, only the following variables are considered during the classification task:

- distance from each robot to the ball (10);
- distance from each robot to their defense goal (10);
- distance from the ball to each goal (2); and
- current CMDragons strategy (1),

leading to a total of 23 variables.

All variables, except from the current CMDragons strategy, are computed variables, that is, they are obtained by applying a function to one or more variables in the original game logs. For instance, the distance from a robot to the ball is computed by using the Euclidean distance between the robot's position and the ball position. The motivation to use computed variables instead the original variables is to build a new set of variables that is: (i) invariant to flipping the image obtained by the vision system vertically and/or horizontally; and (ii) invariant to the robots id [13]. Property (i) guarantees that the learned patterns are independent of the side in which CMDragons started the game as well as the left and right orientation in the field. The identity tracking problem, i.e., the problem of labeling consistently each robot in different games, is avoided by property (ii) that anonymizes the robots. In order to respect property (ii), the distances between robots (CMDragons and opponents) and landmarks are sorted.

Using this set of variables, we want to learn the defense strategy of the opponent. One possible approach is to analyze the time series defined by the games as a whole. However, these time series contain several realizations of the opponent's defense strategy since we assume that every time
Table 1: Statistics about the episodes for each team CMDragons played against in the RoboCup 2007, RoboCup 2008 and US Open 2009. The table is ordered by ascending number of episodes.

<table>
<thead>
<tr>
<th>Goals scored by CMDragons</th>
<th>Number of episodes</th>
<th>Length of the episodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSmar0-07</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>Skuba-08</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>Botnia-07</td>
<td>10</td>
<td>17</td>
</tr>
<tr>
<td>Kiks-08</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>WrightEagle-07</td>
<td>10</td>
<td>30</td>
</tr>
<tr>
<td>EagleKnight-07</td>
<td>9</td>
<td>46</td>
</tr>
<tr>
<td>Plasma-08</td>
<td>2</td>
<td>79</td>
</tr>
<tr>
<td>Plasma-07</td>
<td>5</td>
<td>84</td>
</tr>
<tr>
<td>Fantasia-08</td>
<td>9</td>
<td>96</td>
</tr>
<tr>
<td>Zjuniort-08</td>
<td>5</td>
<td>97</td>
</tr>
<tr>
<td>GaTech-09</td>
<td>10</td>
<td>112</td>
</tr>
<tr>
<td>Zjuniort-07</td>
<td>7</td>
<td>120</td>
</tr>
<tr>
<td>Robodragons-07</td>
<td>8</td>
<td>138</td>
</tr>
<tr>
<td>CMDragons-09</td>
<td>-</td>
<td>-3</td>
</tr>
<tr>
<td>CMDragons-07</td>
<td>-</td>
<td>-196</td>
</tr>
<tr>
<td>CMDragons-08</td>
<td>-</td>
<td>275</td>
</tr>
</tbody>
</table>

CMDragons attacks, the opponent employs its defense strategy. Therefore, we segment each time series in non-overlapping episodes.

Definition 1 (Episode). Given a time series $T$ representing a game, an episode is a maximal segment $S$ of $T$ such that on each frame of $S$: (i) the game is on; (ii) the ball is in the defensive field of the opponent; and (iii) the current strategy being employed by CMDragons is an attack strategy. Also, if the size of $S$ is smaller than $s_{min}$ or size greater than $s_{max}$, then $S$ is discarded.

The game is on from the moment the game restarts until either a goal is scored, the ball leaves the field or a fault is made by a robot and it is a variable in the game logs. Since we are interested in the opponent’s defense strategy, we added in the definition of episode the requirements that characterizes an attack from CMDragons. This because we do not have access to the internal state variables of the opponents to select episodes when they are defending. Therefore, we infer that the opponent is employing a defense strategy by assuming that a defense strategy is the opponent response to an attack.

For all experiments in this paper, we use $s_{min} = 100$ and $s_{max} = 3000$. Also, we represent each episode by the mean and standard deviation of each one of its variables. Therefore if an episode has $t$ timestamps and $f$ variables, it will be represented as a point in $\mathbb{R}^{t}$ instead of a point in $\mathbb{R}^{f \times t}$. This representation simplifies the processes of comparing two episodes since the episodes can have different length (time duration).

Table 1 presents the number of episodes, the average and standard deviation of the episodes length. CMDragons are also included in the table by using the same definition of episodes and considering CMDragons as the opponent team. Instead of inferring when CMDragons are employing a defense strategy, we use the ground truth that is contained in the game logs. As one may notice, the team BSmar0-07 has less episodes than the amount of goals scored by CMDragons, 5 episodes against 10 goals. This is possible because the definition of episodes does not encompass directed kicks from CMDragons' defense field to the opponent's goal and short segments (less than 100 timestamps).

For the remainder of this paper, we denote by $f$ the number of variables considered in the classification task and $E^A$ the matrix with the episodes of team $A$. Therefore if there are $m$ episodes of team $A$ in the game logs, then $A \in \mathbb{R}^{f \times m}$. We also denote by $n$ the total number of episodes in the game logs and $E \in \mathbb{R}^{f \times n}$ the matrix with all episodes. In the next two sections we explore the definition of episodes to develop a measurement to compare episodes and to find the most relevant episodes in the CMDragons game log.

4. Comparing Defense Strategies

In this section, we develop a measurement to compare two episode matrices. The first question worth notice is if this measure should be symmetric. We illustrate this problem with the following example: consider three teams $(A, B$ and $C)$ and three defense strategies $(s_1, s_2$ and $s_3$); the probability $P r(c u r r e n t \ s t r a t e g y = Y | t e a m = X)$ is:

<table>
<thead>
<tr>
<th></th>
<th>$s_1$</th>
<th>$s_2$</th>
<th>$s_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$B$</td>
<td>1/2</td>
<td>1/2</td>
<td>0</td>
</tr>
<tr>
<td>$C$</td>
<td>1/3</td>
<td>1/3</td>
<td>1/3</td>
</tr>
</tbody>
</table>

The team that most resembles the behavior of $A$ is $B$ since in expectation it has more episodes of type $s_1$ than $C$ and both $A$ and $B$ do not play strategy $s_3$. On the other hand, $C$ is the best team to mimic $B$ since it plays strategies $s_1$ and $s_2$. Thus, the measurement to compare two teams does not necessarily need to be symmetric.

As one may notice, the previous example can be solved by using the KL-divergence, that is, given two probability distributions $P$ and $Q$, compute $D[P||Q] = \sum_x P(x) \log \frac{P(x)}{Q(x)}$, where $X$ is the sample space of $P$ and $Q$. However, in the problem of learning the opponent's strategies, the set $X$ is unknown since we do not know all possible strategies.

To overcome this problem, we propose a measurement that compares how well one team can simulate/mimic another one.

Definition 2 (Function $s(., .)$). Let $E^A$ and $E^B$ be the episode matrices for team $A$ and team $B$ respectively, then $s(A, B) = \min_w ||E^A - E^B W||_F$ such that $w_{ij} \geq 0$, where $|| . \||_F$ is the Frobenius norm ($||E||_F = \sqrt{\sum_{i,j} c_{ij}^2}$).

The intuition behind the function $s(., .)$ is that, the smaller $s(A, B)$ is, the better team $B$ can simulate team $A$. This because, if $s(A, B)$ is small, then the norm of $E^A - E^B W$ is small, therefore it is possible to reconstruct the episode matrix $E^A$ with low error by using conical combinations of the episodes of $B$ (columns of $E^B$). Also, each column $w_i$ of the matrix $W$ can be seen as an unnormalized probability distribution over the episodes of $B$. If $B$ plays according to $w$, then the Euclidean norm between the episode $i$ of $A$ and $E$ for the KL-divergence, we consider $0 \log \frac{0}{c} = 0$, for $c \geq 0$ and $c \log \frac{c}{c} = \infty$, for $c > 0$. 47
the expected episode of $B$ is minimized. The function $s(\cdot, \cdot)$ is non-symmetric and in the previous example it gives the same result: $B$ minimizes $s(A, \cdot)$ and $C$ minimizes $s(B, \cdot)$.

In order to compute the function $s(\cdot, \cdot)$, we cast the proposed minimization problem as a quadratic programming problem. First, notice that $\|E\|_F = \sqrt{\text{tr}(EE^T)}$ since $E$ is a matrix defined over the reals. Moreover, for our purposes we can use $s(A, B)^2$ instead of $s(A, B)$ since $s(A, B) \geq 0$ for all $A$ and $B$. Thus we solve the following program

$$\min_W \|E^A - E^B W\|_F^2$$

$$= \min_W \text{tr}((E^A - E^B W)(E^A - E^B W)^T)$$

$$= \min_W \text{tr}(A^T A) - 2\text{tr}(A^T B W) + \text{tr}(W^T B^T B W)$$

which shows that the problem is a quadratic program in $W$. In all these forms, the only constraints of the programs are $w_{ij} \geq 0$.

Using the function $s(\cdot, \cdot)$ to compute the difference between two episode matrices we can classify teams according to the distance to the teams that we already know. Also, we can use this measurement to find patterns in the data set and relations between the teams that we have played before. In section 6 we explore these ideas through a series of experiments and in the next section we extend the usage of $s(\cdot, \cdot)$ to find the main defense strategies in our data set.

5. MAIN DEFENSE STRATEGIES

In this section we look at all the episode at once, in order to find a small set of the most relevant episodes. In other words, we want to find a matrix $D \in \mathbb{R}^{k \times k}$ where $k \ll n$ such that it is possible to reconstruct $E$, with low error, by using linear combinations of the columns of $D$. Clearly, there is a trade off between $k$ and the reconstruction error and in this section such trade off is explored.

Using $D$, it is possible to decompose each episode as a function of the columns $D$. The advantage of this new representation is mainly computational: we can compute an approximation of $s(A, B)$ by using the new representation of $E^A$ and $E^B$ and since this new representation is smaller than the original, a speed up can be obtained.

The problem of finding $D$ such that $D \in \mathbb{R}^{k \times k}$, $k > 0$, is equivalent to find a rank $k$ approximation of the matrix $E$. This can be found by solving a similar optimization problem as the one presented in Section 4: $\min_W \|E - DW\|_F$ such that $\text{rank}(D) = k$.

This problem can be solved optimally through the singular value decomposition of $E$ (SVD decomposition). The SVD decomposition of a matrix $M \in \mathbb{R}^{m \times n}$ is the product $USV^T$, where $U \in \mathbb{R}^{m \times m}$ is unitary, $\Sigma \in \mathbb{R}^{m \times n}$ has nonnegative real numbers on the diagonal and zeros otherwise, and $V \in \mathbb{R}^{n \times n}$ is unitary. This factorization always exists for matrices defined over the reals and $Y = US\Sigma V^T$, where $\Sigma$ has only the first $k$ largest values. $\Sigma$ is the rank $k$ matrix that minimizes $\|M - Y\|_F$ [8]. In our case, $D$ equals the first $k$ columns of $U$ and $W$ is the first $k$ rows of $\Sigma V^T$.

Besides the mathematical interpretation of $D$ obtained using the SVD decomposition, this approach does not offer an interpretation in the robot soccer domain since each $b_{ij}$ can be negative. This implies that features whose meaning requires a positive value, such as the standard deviation the closest robot to the ball, can be represented by a negative value.

In order to get a direct interpretation in our domain, we can enforce that $D$ is composed by subset of the columns of $E$, i.e., $D$ is a submatrix of $E$. The hardness of this new problem, referred in the literature as column-based low-rank matrix approximation [5] and CX-decomposition [9], is unknown [5]. The best approximation algorithm for this problem is proposed by [5]: given $k$, $\epsilon$ and $\delta$, it finds $D$ and $W$ such that:

$$\|E - DW\|_F \leq (1 + \epsilon)\|E - \hat{E}_k\|_F$$

with probability at $1 - \delta$ where $D$ has $O(k^2 \log(1/\delta)/\epsilon^2)$ columns of $E$ and $\hat{E}_k$ is the best rank-$k$ approximation of $E$ (with no constraint).

---

3We assume that the values $\Sigma_{ii}, i \in \{1, \ldots, \min\{m, n\}\}$, are in descending order.
6. EXPERIMENTS

We perform 5 experiments to evaluate our proposed similarity function \( s(\cdot, \cdot) \). In the first experiment we use the \( s(\cdot, \cdot) \) to find similarities between teams and in the second experiment we estimate the number of strategies contained in our game logs. Experiments 3 and 4 evaluate the accuracy of classifying teams according to the proposed similarity function. The last experiment explores the approximated computation of \( s(\cdot, \cdot) \) proposed in Section 5.

### 6.1 Experiment 1: Relation between teams

In the first experiment, we relate the teams that played against CMDragons as well as CMDragons by finding, for each team \( A \), \( \arg\min_{B} s(A, B) \). The result of this experiment is depicted in Figure 2, where each solid arrow \( A \to B \) represents that \( B = \arg\min_{A} s(A, A) \). As shown in Figure 2, we obtain a disconnected graph composed by two directed graphs, \( G_1 \) and \( G_2 \). By looking at the score of each game, one may hypothesize that the teams are separated by their defense strength. That is, teams in \( G_1 \) have a better defense than teams in \( G_2 \), since CMDragons scored less goals on the teams in \( G_1 \) than in the teams in \( G_2 \). In order to verify if this hypothesis is true, in Table 2 we present statistics about the games played by CMDragons against each of the teams.

### 6.2 Experiment 2: Estimating the number of defense strategies

In the second experiment, we estimate how many defense strategies are in our data set. To perform this, we consider the Schwarz regularization criterion (SIC), a criterion used by the machine learning community to evaluate clustering models \( [7] \). The SIC criterion is defined as: given a penalty \( \lambda > 0 \) and the matrices \( D, W \in \mathbb{R}^{n \times n} \), \( SIC(\lambda, D, W) = \|E - DW\|_F^2 + \lambda \log n \). This value is the score of the model, and we want to find a model with small score, since it is composed by the error of approximating \( E \) by \( D \) and \( W \) plus a regularization term that penalizes large models, i.e., large \( \lambda \). Therefore, for a fixed \( \lambda \) we can vary the value of \( k \) and find the one that minimizes \( SIC(\lambda, D, W) \).

Figure 3 presents, for different values of \( \lambda \), the value \( k \) that minimizes \( SIC(\lambda, D, W) \) for the two approaches presented in Section 5, namely, SVD-decomposition and CR-decomposition, to find the matrix \( D \) and \( W \) given \( E \) and \( k \). This plot gives evidence that there are between 7 to 17 different defense strategies in our data set since this range of values lies in the transition phase between the under-constraint (small \( \lambda \)) and over-constraint (large \( \lambda \)) optimal value of SIC.

### 6.3 Experiment 3: Classification according to the defense strategy

The third experiment consists of classifying the teams by their defense strategy. For a given percentage \( p \), we randomly select \( p \) episodes of each team \( i \), denoting these

<table>
<thead>
<tr>
<th>Goals Scored by CMDragons</th>
<th>Ratio of the episodes outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Direct Free Kick</td>
</tr>
<tr>
<td></td>
<td>Indirect Free Kick</td>
</tr>
<tr>
<td></td>
<td>ours</td>
</tr>
<tr>
<td></td>
<td>theirs</td>
</tr>
</tbody>
</table>

| Botnia-07                 | 0.9000                      | 0.3333 | 0.0000 | 0.6667 | 0.0000 |
| BSmart-07                 | 1.0000                      | 0.3000 | 0.3000 | 0.3000 | 0.0000 |
| EagleKnight-07            | 0.9333                      | 0.6667 | 1.3244 | 0.1143 | 0.2000 | 0.3714 |
| Kike-08                   | 1.0000                      | 0.7693 | 1.3237 | 0.1429 | 0.2479 | 0.2307 |
| WrightEagle-07            | 0.9615                      | 0.3035 | 1.9996 | 0.3500 | 0.3000 | 0.0000 |
| GaTech-09                 | 0.9906                      | 0.0094 | 2.3644 | 0.2927 | 0.2073 | 0.1463 | 0.2195 |
| RoboDragons-07            | 0.8168                      | 0.1832 | 3.0577 | 0.2368 | 0.3031 | 0.1754 | 0.4737 |
| Fantasia-08               | 0.9888                      | 0.0125 | 3.9168 | 0.5238 | 0.3357 | 0.0595 | 0.1439 | 0.2381 |
| Zjunclict-07              | 0.6552                      | 0.3446 | 5.1857 | 0.5893 | 0.0982 | 0.0804 | 0.0804 | 0.1518 |
| PlasmaZ-07                | 0.7266                      | 0.2714 | 6.6640 | 0.5038 | 0.0625 | 0.1719 | 0.0781 | 0.0938 |
| Zjunclict-08              | 1.0000                      | 0.0000 | 7.1343 | 0.6586 | 0.1098 | 0.1585 | 0.0244 | 0.0488 |
| Skuba-08                  | 1.0000                      | 0.1402 | 2.0114 | 0.8750 | 0.0120 | 0.0000 | 0.0000 | 0.0000 |
| PlasmaZ-08                | 2.0000                      | 1.4784 | 8.6104 | 0.6119 | 0.0299 | 0.1045 | 0.0896 | 0.1642 |

Table 2: Statistics about the games played between CMDragons and opponent teams. Columns 3 and 4 represent ratio between the two different attack strategies employed by CMDragons; columns 5 and 6 presents statistics about the time to CMDragons score a goal; columns 7 to 11 represent ratio between the possible outcome of an episode. This outcome can be: (7) defense, i.e., the opponent successfully neutralized the attack and started a counter-attack; (8,9) a directed free kick for CMDragons and the opponent; and (10,11) a indirect free kick for CMDragons and the opponent. This table is sorted by ascending average time to score a goal.
episodes as $T^i$, and use them to classify the remainder episodes. The remainder episodes $R$ are grouped by team, such that each set of episodes $R^i$ has only samples of the team $i$. We classify $R^i$ as team $j$ if $T^j = \arg \min_x s(R^i, x)$.

Besides the classification accuracy, we also present a second measurement called rank. The rank of a team $i$ is the position of $T^i$ when all matrices $T^j$ are sorted, in ascending order, by the value of $s(R^i, T^j)$. Therefore if the rank of $i$ is 1, then $R^i$ is correctly classified, since $T^i$ has the minimum value of $s(R^i, T^j)$ for all $T^j$. Table 3 presents the result of 25 executions of this experiment for $p$ equals 30%, 40% and 50%.

This experiment shows that we can perfectly classify 7 out of 16 of the teams, namely Fantasia-08, Zjuniect-08, GaTech-09, Zjuniect-07, RoboDragons-07, CMDragons-07 and CMDragons-08, using 40% and 50% of the data as training. Also, the average rank for EagleKnight-07, CMDragons-09, PlasmaZ-08 and PlasmaZ-07 is at most 2, i.e., the proposed measurement $s(\cdot, \cdot)$ ranked the correct answer as the second most similar team. For the remaining 5 teams, namely Skuba-08, Kiks-08, Botnia-07, WrightEagle-07 and BSmart-07, the classification accuracy is not satisfactory since the average rank for these teams is at least 4. One explanation to this poor performance for these teams is that they are the 5 teams with the least amount of episodes in our data set (Table 1).

### 6.4 Experiment 4: Classification of random mixture of teams

The fourth experiment is an extension of the previous one. Instead of using teams $R^i$ with only samples of team $i$ for testing, a random mixture of the teams $\tilde{R}^i$ is used. The testing set $R^i$ is built by selecting 15 random episodes of team $i$ and one episode of each team $j \neq i$. Therefore, the probability of an episode in $\tilde{R}^i$ is from team $j$ is $1/2$ if $i = j$ and $1/30$ otherwise. Given the values of $s(\tilde{R}^i, \cdot)$, we define the induced probability distribution $\tilde{P}(\tilde{R}^i)$ as the team $j$ as proportional to $1/s(\tilde{R}^i, T^j)$, where $T^j$ is the matrix with the training episode for team $j$. Table 4 contains the results of this experiment. The KL-divergence between the original distribution and the obtained $\tilde{P}$ is smaller, i.e. differ less, than the uniform distribution (random guess) in half of the cases. By looking at the mode of $\tilde{P}$, i.e. $\arg \max_x \tilde{P}(\tilde{R}^i)$ the team $x$, we observed that for all teams, except CMDragons-08, the mode of $\tilde{P}$ is either GaTech-09 or Fantasia-08. This is interesting since these two teams are the centers of each graph in Figure 2.

### 6.5 Experiment 5: Approximating the value of $s(\cdot, \cdot)$

In the last experiment we compare the approach presented so far, i.e., the exact computation of $s(\cdot, \cdot)$, with the approximated approach suggested in Section 5: to use a rank $k$ approximation of the episodes matrix to decrease the number of variables describing each episode. For this experiment, 50% of the data set was used as training set and remaining 50% as testing set. The chosen values of $k$, namely 17, 12 and 7, are based in the second experiment (Figure 3). The results are presented in Table 5 and as expected, the classification accuracy, the rank and the running time decrease as $k$ decreases. One may also notice that the exact computation and the SVD decomposition for $k = 17$ approaches achieve almost the same classification accuracy and rank, however the SVD approach is about 0.2 seconds faster than the exact computation approach.

### 7. CONCLUSIONS AND FUTURE WORK

In this paper we have introduced a novel approach to compare team strategies. This approach relies on the best approximation, according to the Frobenius norm, of the matrices representing the episodes in our data set of each team. Therefore, we consider that a team $A$ is similar to a team $B$,
<table>
<thead>
<tr>
<th></th>
<th>30% for training</th>
<th>40% for training</th>
<th>50% for training</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy</td>
<td>Rank</td>
<td>Accuracy</td>
<td>Rank</td>
</tr>
<tr>
<td>BSma Starts-07</td>
<td>0.04</td>
<td>0.20</td>
<td>7.60</td>
<td>4.40</td>
</tr>
<tr>
<td>Skuba-08</td>
<td>0</td>
<td>0</td>
<td>7.28</td>
<td>1.92</td>
</tr>
<tr>
<td>Botnia-07</td>
<td>0.20</td>
<td>0.40</td>
<td>6.92</td>
<td>5.13</td>
</tr>
<tr>
<td>Kit-08</td>
<td>0</td>
<td>0</td>
<td>6.12</td>
<td>2.58</td>
</tr>
<tr>
<td>WrightEagle-07</td>
<td>0</td>
<td>0</td>
<td>5.40</td>
<td>2.74</td>
</tr>
<tr>
<td>EagleKnight-07</td>
<td>0.04</td>
<td>0.48</td>
<td>2.08</td>
<td>1.66</td>
</tr>
<tr>
<td>PlasmaZ-08</td>
<td>0.00</td>
<td>0.50</td>
<td>1.56</td>
<td>0.76</td>
</tr>
<tr>
<td>PlasmaZ-07</td>
<td>0.08</td>
<td>0.47</td>
<td>1.32</td>
<td>0.47</td>
</tr>
<tr>
<td>Fantasia-08</td>
<td>0.06</td>
<td>0.20</td>
<td>1.04</td>
<td>0.20</td>
</tr>
<tr>
<td>Zunlucht-08</td>
<td>1.00</td>
<td>0</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>GaTech-09</td>
<td>1.00</td>
<td>0</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Zunlucht-07</td>
<td>1.00</td>
<td>0</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>RoboDragons-07</td>
<td>1.00</td>
<td>0</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>CMDragons-09</td>
<td>0.04</td>
<td>0.50</td>
<td>2.00</td>
<td>1.11</td>
</tr>
<tr>
<td>CMDragons-07</td>
<td>1.00</td>
<td>0</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>CMDragons-08</td>
<td>1.00</td>
<td>0</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Statistics about the classification experiment. For each percentage of the data used for training, we classified each team in the remainder of the data (testing data). This table presents the average and standard deviation of the classification accuracy and the average and standard deviation of the rank for 25 executions of this experiment. The rank of a team A is defined as the position of s(A', A) in the sorted vector of s(A', i), where A' are the instances of A in the test data. This table is sorted by the number of episodes of each team (see Table 1).

<table>
<thead>
<tr>
<th></th>
<th>KL-divergence</th>
<th>Mode</th>
<th>Pr(_Mode)</th>
<th>Pr(real Mode)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GaTech-09</td>
<td>0.3753</td>
<td>GaTech-09</td>
<td>0.1625</td>
<td>0.1625</td>
</tr>
<tr>
<td>Fantasia-08</td>
<td>0.4174</td>
<td>Fantasia-08</td>
<td>0.1457</td>
<td>0.1457</td>
</tr>
<tr>
<td>RoboDragons-07</td>
<td>0.4745</td>
<td>GaTech-09</td>
<td>0.1369</td>
<td>0.1313</td>
</tr>
<tr>
<td>CMDragons-08</td>
<td>0.4791</td>
<td>CMDragons-08</td>
<td>0.1382</td>
<td>0.1382</td>
</tr>
<tr>
<td>CMDragons-07</td>
<td>0.5342</td>
<td>GaTech-09</td>
<td>0.1341</td>
<td>0.1208</td>
</tr>
<tr>
<td>Zunlucht-08</td>
<td>0.6400</td>
<td>Fantasia-08</td>
<td>0.1370</td>
<td>0.0921</td>
</tr>
<tr>
<td>Zunlucht-07</td>
<td>0.6711</td>
<td>Fantasia-08</td>
<td>0.1257</td>
<td>0.0887</td>
</tr>
<tr>
<td>CMDragons-09</td>
<td>0.7012</td>
<td>GaTech-09</td>
<td>0.1468</td>
<td>0.0822</td>
</tr>
<tr>
<td>Kit-08</td>
<td>0.7502</td>
<td>GaTech-09</td>
<td>0.1636</td>
<td>0.0722</td>
</tr>
<tr>
<td>PlasmaZ-08</td>
<td>0.8111</td>
<td>Fantasia-08</td>
<td>0.1387</td>
<td>0.0675</td>
</tr>
<tr>
<td>WrightEagle-07</td>
<td>0.8418</td>
<td>Fantasia-08</td>
<td>0.1316</td>
<td>0.0583</td>
</tr>
<tr>
<td>PlasmaZ-07</td>
<td>0.8550</td>
<td>Fantasia-08</td>
<td>0.1379</td>
<td>0.0587</td>
</tr>
<tr>
<td>Skuba-08</td>
<td>0.8900</td>
<td>Fantasia-08</td>
<td>0.1505</td>
<td>0.0591</td>
</tr>
<tr>
<td>EagleKnight-07</td>
<td>1.0590</td>
<td>GaTech-09</td>
<td>0.1155</td>
<td>0.0367</td>
</tr>
<tr>
<td>BSma Start-07</td>
<td>1.1018</td>
<td>GaTech-09</td>
<td>0.1215</td>
<td>0.0315</td>
</tr>
<tr>
<td>Botnia-07</td>
<td>1.1302</td>
<td>Fantasia-08</td>
<td>0.1410</td>
<td>0.0306</td>
</tr>
</tbody>
</table>

Table 4: Results of the classification experiment using random mixture of the teams. For each team A, it was select 15 random episodes of A and one episode of each other team. The remaining of the data set is used for training. The first column contains the KL-divergence between the original distribution and the one obtained by our proposed method. The KL-divergence between the original distribution and the uniform distribution is 0.7254, therefore, our method performs better than the random guess for the 8 teams (top 8 lines). The second column contains the mode of the induced probability distribution, i.e., the team that has maximum probability and the third column its probability; and the fourth column presents the induced probability of the mode of the original distribution (the team on each line). The induced probability distribution was obtained by averaging 25 runs of the experiment. This table is sorted by ascending KL-divergence.
if the episode matrix of A is best approximated by a conical combination of the episodes in the episode matrix of B.

We presented experiments, using real data from the RoboCup 2007, RoboCup 2008 and USOpen 2009, showing how classification can be performed using the proposed measurement. We also applied this measurement to find similarities in the defense strategies of the teams in our data set. The obtained patterns are corroborated by the presented statistics of the games played by CMDragons against these teams.

Possible future research directions include extending the proposed approach to handle episodes represented as time series instead of the representation by mean and standard deviation used in this work. This extension is non-trivial since each episode has different lengths (time duration). The trivial extension of applying the same definition of $s(\cdot, \cdot)$ in episode tensors, i.e., matrices in $\mathbb{R}^{T \times T \times n}$ where $t$ the length of the episode, does not work, thus additional research is needed to find a suitable approach.

A second general direction for further investigation is to explore adaption according to the opponent. That is, to use of the knowledge from the previous opponents when playing against an unknown opponent through the proposed decomposition of the unknown strategies.

Acknowledgments

The authors thank the past and current CMDragons team members, in particular Stefan Zickler, Joydeep Biswas and James Bruce, for developing and sharing the CMDragons log data used in this work.

8. REFERENCES


Hypotheses based multi-object tracking in the RoboCup MidSize league

(Extended Abstract)

Rob Janssen
Eindhoven University of Technology
Department of Mechanical Engineering
Control Systems Technology group
P.O. Box 513 Eindhoven, the Netherlands
r.j.m.janssen@tue.nl

René van de Molengraft
Eindhoven University of Technology
Department of Mechanical Engineering
Control Systems Technology group
P.O. Box 513 Eindhoven, the Netherlands
m.j.g.v.d.molengraft@tue.nl

ABSTRACT
One of the main challenges in the RoboCup MidSize league is to create a global view, or world model, of peer and opponent players in the game environment. This view is essential for strategic gameplay and global path planning. In this paper the team of TechUnited Eindhoven describes their solution to this issue. Ego and omnivision object measurements are shared amongst peer players. Each peer individually processes the measurements according to an hypotheses based sequential clustering algorithm. For each cluster Kalman observers are initiated from which an estimated position and velocity can be derived. This paper describes the first world model design known in the MidSize league that includes dynamics and labeling of peer and opponent players.

Categories and Subject Descriptors
H.4 [Information Systems Applications]: Miscellaneous

General Terms
Algorithms

Keywords
Collective intelligence, Mobile agents

1. INTRODUCTION
In the RoboCup MidSize league two teams of autonomous robots compete with each other in a game of soccer. By using an omnivision camera a robot can obtain the position of other robots in its nearby environment, but due to resolution deterioration this becomes difficult for objects positioned farther away (>5m). Another problem when using only omnivision information is that a robot can not visually distinguish between peer and opponent players. To overcome these issues a global view must be created, in which all peer players share their available information of the objects in the field. In the RoboCup MidSize league this global view is referred to as a world model. With a world model it is for instance possible to plan a path from a defensive position straight to the opponent goal, or to pass a ball over large distances to a locally unobservable peer. In this paper the team of TechUnited Eindhoven explains the design and implementation of their world model. In this approach the ego and object measurements of all peer players are shared and efficiently clustered such that for each object in the field, peer or opponent, a unique position and velocity is estimated. Other teams participating in the MidSize league have implemented a basic world model, from which only a rough estimate for the position can be obtained. The reigning world champion RFC Stuttgart uses a shared database for all agents [1]. In this implementation no data association between measurements is performed. An agent can simply access the information another agent has available. The Brainstormer tribots use a grid-based approach to determine overall occupancy on the field [2]. This approach lacks any form of tracking, and therefore no velocity estimates for opponent players are determined. The team of Cambada has implemented a more advanced method [3]. In this method a shared database can be accessed, in which a basic form of outlier-based clustering is performed.

The world model of TechUnited is an hypotheses based sequential clustering algorithm that clusters ego and omnivision object measurements into an a priori unknown number of dynamic objects. Basically, it is a Bayesian filter in which the state space grows with each new measurement. Each state describes a hypothesis of possible associations between measurements and clustered objects. Updating of the hypotheses probabilities is done by evaluating the metric distances between measurement and associating objects. To cope with dynamics, Kalman observers are initiated for each new observed cluster. Clustering data usually requires to know the actual number of clusters up-front [4], or to perform several trials while evaluating the rate of change using for instance the L method [5]. The described method in this paper is based on the work of Schubert and Sidenbladh [6], where the actual number of originating objects is supposed to be unknown. In their approach only static objects were considered. The approach described in this paper also includes the dynamics involved in the RoboCup environment, together with an addition that labels an observed object as peer or opponent.

2. METHOD DESCRIPTION
The measurements of all peer players are shared through WiFi. The measurements that each peer shares include a global ego-position and the global position estimates of objects that the peer observes through its omnivision camera. Each shared measurement holds a timestamp, by which the
measurements are sorted in time before they are sequentially processed. A label is added to the measurement, to classify a clustered object as either peer or opponent. For each measurement the following steps are performed:

1) Expansion of the hypotheses tree
In Fig 1 the hypotheses tree is visualized, where each new level describes the growing state space. For each measurement a new level in the hypotheses tree is appended. When processing the first measurement two new hypotheses are created. Either the first measurement can be classified as clutter ([0]), or it can associate with a new observed object ([1]).

When the second measurement is processed the discrete state-space grows to five possible hypotheses. The circle with ([0,1]) describes the hypothesis that the first measurement was clutter and the second measurement associates to a new object. The circle with ([1,2]) describes the hypothesis that the first measurement associates to a first object and the second measurement associates to a second object, etc...

2) Inheritance of attached label
The object measurements obtained through the omnivision camera are labeled zero and the ego-position measurements are labeled X, the peer’s ID number (1,2,6). If an observed object associated with a measurement labeled X, the observed object is labeled as peer X. If all associating measurements of a clustered object contained a zero-label, the object is labeled as opponent.

3) Object propagation
To cope with the dynamics of the moving objects, the objects are propagated during processing of a measurement. This propagation is done by initiating a Kalman filter for each object in the hypotheses tree. Each observed object is propagated according to the time interval between consecutive measurements. To minimize processing time a constant velocity model for the objects is assumed, and the Kalman gains are static and determined empirically. If the processed measurement associates with an observed object in the hypotheses tree, also a measurement update is performed.

4) Updating of hypotheses probabilities
The update for the hypothesis probability depends on the Gaussian distance from a measurement to an associating object. The closer a measurement is to an associating object, the larger the increase in hypothesis probability.

5) Pruning and normalization
The hypotheses tree is pruned and normalized to keep it maintainable. Pruning is performed by keeping only a fixed number of hypotheses that have the highest probabilities.

6) Selecting the best hypothesis
Selecting the best hypothesis is done according to the Maximum A Posteriori Estimate. The selected hypothesis contains the total number of peer and opponent players. From the associating Kalman filter the respective position and velocity of a player can be derived.

3. RESULTS
A static representation of the outcome of the algorithm is depicted in Fig. 2.

![Figure 2: World model output semifinal 2010.](http://www.youtube.com/watch?v=7C8cilgU66Q)

Measurements are indicated by crosses, peer players by blue dots and opponent players by red dots. An animated video that visualizes tracking performance can be found on http://www.youtube.com/watch?v=7C8cilgU66Q.

4. REFERENCES
Human-Guided Real-Time Multi-Agent Coordination in Dynamic Uncertain Domains

Rajiv T. Maheswaran  
Univ. of Southern California  
Information Sciences Institute  
4676 Admiralty Way #1001  
Marina Del Rey, CA, USA  
maheswar@isi.edu

Craig M. Rogers  
Univ. of Southern California  
Information Sciences Institute  
4676 Admiralty Way #1001  
Marina Del Rey, CA, USA  
rogers@isi.edu

Romeo Sanchez  
Univ. of Southern California  
Information Sciences Institute  
4676 Admiralty Way #1001  
Marina Del Rey, CA, USA  /rsanchez@isi.edu

Pedro Szekely  
Univ. of Southern California  
Information Sciences Institute  
Marina Del Rey, CA, USA  
psezekely@isi.edu

ABSTRACT
Creating decision support systems to help people coordinate in the real world is difficult because it requires simultaneously addressing planning, scheduling, uncertainty and distribution. Generic AI approaches produce inadequate solutions because they cannot leverage the structure of domains and the intuition that end-users have for solving particular problem instances. We present a general approach where end-users can encode their intuition as guidance enabling the system to decompose large distributed problems into simpler problems that can be solved by traditional centralized AI techniques. Evaluations in field exercises with real users show that teams assisted by our multi-agent decision-support system outperform teams coordinating using radios.

Categories and Subject Descriptors
I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence—Multiagent systems

General Terms
Algorithms, Performance, Design, Experimentation

Keywords
Real-Time Dynamic Planning and Scheduling, Human-Agent Interaction, Human Guidance, Decision Support

1. INTRODUCTION
Teams of people need to coordinate in real-time in many dynamic and uncertain domains. Examples include disaster rescue, hospital triage, and military operations. It is possible to develop plan a priori, but many parts of these plans must be left unspecified because people won’t know exactly what needs to be done until

The work presented here is funded by the DARPA COORDINATORS Program under contract FAK500-05-C-0032. The U.S. Government is authorized to reproduce and distribute reports for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of any of the above organizations or any person connected with them. Approved for Public Release, Distribution Unlimited.

they are executing the plan in the field. Additionally, requirements and tasks can evolve during execution.

Our work addresses a fundamental multi-agent systems endeavor of creating decision support systems that help humans perform better in these domains. The technical challenges to compute good solutions for these problems have been well documented [10, 7, 3].

Established approaches address subsets of the problem, but none have adequately addressed the full problem. Classical planning techniques can barely compute the sets of actions that each person should perform for large problems involving metric resources and cannot cope at all with uncertainty and distribution. Decision-theoretic planning addresses uncertainty, but performance degrades with increased distribution and scale. Distributed constraint optimization techniques address distribution, but do not address temporal reasoning, uncertainty or scale.

In practice, it is possible to address specific domains with custom algorithms that use powerful heuristics to leverage the structures unique to that domain. These solutions are expensive to create as even these domains involve planning, uncertainty and distribution. The goal remains to develop generic approaches that produce good solutions that help human teams in many domains.

We introduce a new approach, STaC, based on the premise that people have good intuitions about how to solve problems in each domain. The idea is to enable users to encode their intuition as guidance for the system and to use this guidance to vastly simplify the problems that the system needs to address. The approach is related to heuristic planning, but differs in two important aspects. First, the goal is to capture intuition about solving specific instances of the problem rather than providing heuristics that apply to many instances in the domain. End-users rather than domain experts or developers encode heuristics for the system. Second, in STaC, the intuition is not captured by rules of what actions to take in specific situations, but rather as a decomposition of the problem into simpler problems that can be solved independently.

Figure 1 illustrates the approach. The large box P denotes all problems from multiple domains that can be encoded using a generic modeling formalism. We used a formalism based on TAEMS [9], but the approach can be used with any hierarchical task network formalism. The oval labeled AF represents generic artificial intelligence approaches designed to solve all the problems that can be encoded in P. The solution quality of these generic solvers is questionable because, as detailed above, the problems present chal-
Figure 1: The STaC Approach

Figure 2: Field Exercise Images from Rome, NY

Park are shown in Figure 3. They were organized and evaluated by independent parties contracted by the DARPA Coordinators program. The rules of the field exercise were created collaboratively by the teams building coordinator agents, the independent evaluation team, and subject matter experts. The specific instances or scenarios that comprised the test were chosen by the independent evaluation team.

2.1 Sites

Various locations were selected as sites and a feasible road network was constructed. If the site was populated, it could have injured people in either critical and serious condition. Populated sites would also have gas, power and water substations which may have been damaged. In addition, any site could have facilities such as a hospital, clinic, warehouse, gas main station, power main station and water main station. A team would obtain points by rescuing injured to hospitals or operational clinics (before a deadline associated with each injured person) and by repairing main stations and substations. The goal of a scenario was to accumulate as many points as possible before the scenario deadline.

2.2 Teams

The teams were composed of 8 field agents and 2 command agents. Each agent had a different set of skills. Three specialists in gas, power and water could perform major and minor repairs in their respective skill area. The medical specialist could load any type of injured person by themselves. The remaining four survey specialists could have any collection of skills involving minor repairs. The field agents could move throughout the field exercise area and perform actions. The command agents were located at a base where they helped to coordinate the activities of the team. The Radio Team communicated only with radios. Our CSC Team had ruggedized tablet computers on which our agents were loaded, in addition to radios. The tablets had cell modems and GPS.

2.3 Dynamism and Uncertainty

Many outcomes were revealed during the game for which little or no likelihood information was given a priori, i.e., no probability distribution functions over outcomes. Teams did know the space of possible outcomes beforehand. A survey for damage at a main station or substation revealed the number and type of problems chosen from a set of known possible problems. A survey for injured at a populated site revealed the number, types and deadlines for the injured at that site. As the result of a survey, any team member might be injured, forcing them to go to an operational medical facility to recover before proceeding with any other action. A survey could
also reveal that the vehicle of the agent doing the survey had failed and would require a vehicle repair before the agent could travel to any other site. While traveling, agents could encounter road blocks which could not be passed until fixed. Travel and repair times could vary and repairs could fail. These dynamic and uncertain events were planned parts of the exercise. In addition, the teams had to address uncertainties inherent in the environment, such as noisy radios, weather, and other activities in the public settings. Furthermore, most of these outcomes were only observable by the agent encountering the outcome.

2.4 Scenario

The independent evaluation team chose the scenario from the space of possible exercises and informed the teams of the details below one day prior to the test: (1) the locations of populated sites and facilities, (2) the road network and ranges on travel times between sites, (3) a range for the total number of injured at each site, (4) the points for rescuing each type of injured, which could vary by type and site, (5) the points for repairing each substation or main station, which could vary by type and site, (6) potential problems after surveys for damage and corresponding repair options, (7) ranges on repair times, (8) likelihoods of failure for every repair activity, and (9) the skills of the survey specialist agents. The deadlines (for the scenario and injured) did not allow teams to do all possible repairs and rescues. The teams had one day to form a high-level strategy. The only element of uncertainty which could be modeled accurately with a probability density function was (8). When a team member completed a repair activity, they would call the evaluation team, which would report whether the repair was successful or a failure. The range in (3) was respected by the scenario designers, i.e., the number of injured did not fall outside the given range.

2.5 Coordination

There were many rules and couplings that forced agents to coordinate. To do surveys, gas and power substations at the site had to be off, which required agents with those skills. Two agents had to be at the same location simultaneously to load a critically injured person or repair a road block. Repair options could involve multiple tasks and require two agents with certain skills to act in synchrony or in a particular sequence. Some repair options required kits which guaranteed their success, but kits were available only at warehouses. Agents could transport at most one entity, i.e., either a repair kit or a single casualty. A substation was considered repaired only if the corresponding main station was also repaired. A clinic was not operational until all substations at the site and all corresponding main stations were repaired. These are examples of rules that, along with the dynamism and uncertainty in outcomes mentioned earlier, created challenging real-time real-world distributed coordination problems.

The goal was to see if humans operating with radios and a multi-agent decision-support system could outperform humans operating with only radios. Although the field exercises still abstracted some aspects of a real-world disaster scenario, we believe they closely approximated the challenges of helping a human team solve difficult real-world problems.

3. RELATED WORK

The STaC framework was developed during the DARPA Coordinators program. In the first two years, DARPA ran competitive evaluations on simulated scenarios, and CSC, the underlying system behind the STaC framework, won such evaluations by considerable margins against two competing approaches: an MDP-based approach [11] and an STN framework [14].

Figure 3: Stanton Woods Park, Herndon, VA

The MDP-based [11] approach addressed the infeasibility of reasoning over the joint state space by setting the circumstance set to a subset of local state space that is reachable from the current local state, unrolling the state space by doing a greedy estimation of boundary values. It biased its local reward function on the commitments made by the agents during execution. However, such approximations lose critical information, exploring state spaces that are far from good distributed solutions.

The STN framework [14] addressed temporal uncertainty by using a time interval (instead of a point) as the circumstance that denotes feasible start times for a method to be executed. The system used constraint propagation to update the start intervals of the agents' activities during execution. A policy modification phase was triggered if execution was forced outside the given set of intervals. One of the problems of this approach is that agents tried to maintain consistency and optimize their local schedules, losing information that was needed to timely trigger policy modifications for their schedules.

We encoded scenarios of the field exercise as planning problems using PDDL [5]. The motivation was to identify to the extent to which current automated planning technology can address complex distributed, resource-driven, and uncertain domains. Unfortunately, this proved to be extremely difficult for state-of-the-art planning systems. From the set of planning systems tried, only LPG-TD [6], and SGPAN [3] solved a simplified problems, after uncertainty, dynamism, non-determinism, resource-metrics, partial observability and deadlines were removed. Planners were unable to scale to more than 5 sites. LPG-TD produced solutions more efficiently but less optimally.

In general, mixed-initiative approaches where humans and software collaborate can often produce better solutions for complex problems. Mixed-initiative planning systems have been developed where users and software interact to construct plans. Users manipulate plan activities by removing or adding them during execution while minimizing the changes from a reference schedule [1, 8, 12]. However, most of these systems are centralized, so humans and systems are fully aware of the entire plan, and of the consequences of updating it. In our scenario, agents (including humans) have subjective views of the world, and any decision may trigger many unknown global effects.

Multi-agent systems for disaster domains have been studied in
the context of adjustable autonomy. The idea is to improve limited human situational awareness that reduces human effectiveness in directing agent teams by providing the flexibility to allow for multiple strategies to be applied. A software prototype, DEFACTO, was presented and tested on a simulated environment under some simplifications (e.g., no bandwidth limitations, reliable communications, omnipresence) [13]. Our work also recognizes the importance of flexible frameworks to allow better human-agent interactions. The test-bed presented in this paper does not make any major simplifications, being a first step toward creating multi-agent systems for real-world problems.

4. THE STaC APPROACH

Our goal is to create a general framework for incorporating human strategic guidance. We introduce the formalism for STaC guidance and give an example from our domain. We then describe how this guidance is executed with the use of Total Capability Requirement (TCR) sets. We provide an example of a TCR set and discuss how dynamic updates enable execution of the guidance.

4.1 STaC Guidance

We make the following assumptions about a general multi-agent coordination problem. There are a set of agents $N$ and a set of actions $A$. Agents have capabilities from a set of capabilities: $\Theta_n \in \Theta$. Each action is mapped to a capability, i.e., $\gamma: A \to \Theta$. An agent can perform any action for which it has the capability.

The problem is composed of a collection of tasks $T$. Each task $t \in T$ is associated with a set of potential actions involved in completing it: $A^t \subset A$. It is not necessary that $\{A^t\}$ be disjoint. Furthermore, for the purposes of guidance, it is not relevant how these tasks relate to the actual reward function. It is only important that the notion of tasks exists.

We can define a generic representation for human strategic guidance as follows. Guidance is an ordered set of guidance groups: $G = \{G_i\}$. Each guidance group $G_i$ is associated with a subteam of agents $S_i \subset N$ and an ordered set of guidance elements $E_i$. Each guidance element $e_i \in E_i$ is composed of a task $t_i \in T$, a set of constraints $C_i$, and a temporal bound $b_i$. The constraints $C_i$ are a collection of capability-number pairs $(\theta, n)$ where $\theta \in \Theta$ and $n \in \mathbb{Z}^+$ is a non-negative integer. The pair $(\theta, n)$ indicates that each agent in the subteam can use the capability $\theta$ at most $n$ times for the task in the guidance element. The temporal bound $b_i \in \{0\} \cup \mathbb{R}^+$ is another constraint that can indicate that the guidance element is only valid if the time remaining is greater or less than some number ($b_i = 0$ indicates no temporal constraint). Thus:

$G = \{G_i\} = \{(S_i, (t_i, C_i, b_i))\} = \{((S_i, \{t_i, C_i, b_i\}))\}$

We refer to this as the STaC (Subteam-Task-Constraints) formalism for strategic guidance. One can now define a strategy composed of a sequence of subteams, each responsible for a collection of tasks, each of which are to be performed under some constraints. We note that since agents will traverse the elements of this guidance in order, STaCs are always queues.

4.1.1 Field Exercise Example

We first defined a set of capabilities that were relevant to the field exercise. We also associated each capability with several capability classes for more compact expression of constraints. Below are the set of capabilities and associated classes for actions involving gas and injured, respectively. Capabilities and classes at power and water are analogous to those for gas.

<table>
<thead>
<tr>
<th>gas_major: gas, gas_main</th>
<th>gas_minor: gas, gas_main</th>
</tr>
</thead>
<tbody>
<tr>
<td>survey_gas_main: gas, gas_main, survey</td>
<td>survey_gas_sub: gas, survey</td>
</tr>
<tr>
<td>turn_off_gas_main: gas, gas_main, turns off</td>
<td>turn_off_gas_sub: gas, turns off</td>
</tr>
<tr>
<td>pickup_gas_kit: gas, pickup</td>
<td>dropout_gas_kit: gas, dropout</td>
</tr>
</tbody>
</table>

Consider the STaC guidance fragment below. We see an ordered set of guidance groups, each with a subteam of agents and an ordered set of guidance elements. The only() operator sets the capability-number pairs for all capabilities not in the argument to zero. The no() operator sets the capability-number pairs for all capabilities in the argument to zero. The intent of this plan fragment is for the survey specialist to turn off services at the substations at Site 4 and Site 3, enabling other agents to work there. The gas and survey specialists go to the warehouse at Site 6, pick up gas kits, then restore the gas main station and gas substation at Site 1. The medical specialist and survey specialist are responsible for making sure they each rescue two critically injured people before rescuing all others. The gas and power specialist are then responsible for doing everything except water-related actions at Site 4, but if less than 10 minutes are left in the scenario, they switch to rescuing injured.

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Here, the tasks chosen for each guidance element are all those associated with a particular site. This is not a requirement in the guidance formalism. For example, the second guidance group could have also been:

| [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] |

This would have specified a fixed ordering between repairing the main station and the substation which did not exist in the original. The expression of guidance is not necessarily unique and can be tailored to the intuition and structure that the designer finds most appropriate.

4.2 STaC Execution

While STaC guidance is compact and has intuitive meaning for a human, the agents have no semantic awareness of what it signifies beyond identifying tasks and limiting actions. This is due to the generality of the formalism. Furthermore, the guidance does not specify which actions to perform, which agents should perform them, the timing of these actions or how to react to any dynamism and uncertainty during execution. We address those challenges here.
4.2.1 Total Capability Requirement (TCR) Sets

Given the S Tac formalism, one of the key decisions that every agent must make is when to transition from one task to another. A simple solution is to wait until the current guidance element is completed and then move to the next guidance element (which may involve going to the next relevant guidance group). This approach would lead to significant waste if the agent were unable to contribute to the current guidance element.

Consider the example shown in Section 4.1.1. If the gas specialist arrives at Site 1 first and discovers that all the repair options for the gas main station and gas substation can be completed by the gas specialist alone, or that there exists repair options for both the main station and the substation that can be performed by the gas specialist alone and are guaranteed to succeed, the gas capabilities of the survey specialist are not needed. It may make sense for the survey specialist to skip Site 1 and head to Site 4 to help the medical specialist rescue injured, even though the repairs at Site 1 have not been completed. It is important to determine dynamically whether the capabilities of each agent in the subteam are needed for the task being executed in the guidance element.

Total Capability Requirement (TCR) sets are a mechanism to achieve this. For each task $t \in T$, there is an associated TCR set $R^t = \{R^t_1\}$, which is a set of requirements. Each requirement $R^t_1 = (n^t_1, Q^t_1)$ is a tuple of a requirement number $n^t_1$ and requirement type $Q^t_1$. A requirement type $Q^t_1 = (q^t_{i,j})$ is a collection of requirement elements, where each requirement element is a tuple $q^t_{i,j} = (e^t_{i,j}, n^t_{i,j}, l^t_{i,j})$ where $e^t_{i,j}$ is a capability, $l^t_{i,j}$ is a location, and $n^t_{i,j}$ is an element number. Thus, $R^t_1 = \{R^t_1\} = \{(n^t_1, Q^t_1)\} = \{\{(n^t_1, q^t_{i,j})\}\} = \{(n^t_1, q^t_{i,j})\} = \{(n^t_1, (e^t_{i,j}, l^t_{i,j}, n^t_{i,j}))\})$.

Consider the example above which is a possible TCR set for task_site_01. This indicates that there are two instances of the need for a single agent with gas minor capability, one instance of a need for two agents with gas minor capability, four instances of a need for two agents capable of loading a critically injured person and one instance of a need for having an agent with power minor capability at Site 1 at the same time that there is an agent with power minor capability at Site 3. The first requirement could occur because the gas main station has two problems, each of which could be solved with a gas minor repair. The second requirement could occur because the gas substation has one problem that requires two agents with gas minor skills to perform a synchronized repair. The third requirement could be due to the discovery of four critically injured people. The fourth requirement represents the need for remote synchronization: the need for two agents at two different locations at the same time. In the field exercise, some power substations required an agent at the substation and another at the main station simultaneously to turn the power substation on.

If the guidance element was: \( \{\text{task_site_01}, \{\text{gas}\}, 0\} \)
then only the first two requirements involving the gas minor capability would be considered when deciding whether an agent should remain committed or released from the task. The TCR sets are dynamically updated such that once a skill is no longer needed, as repairs are completed or injured are loaded, the appropriate requirements are decremented or deleted.

4.2.2 Calculating TCR Sets

Our calculation of TCR sets can best be described in the context of our modeling specification for the field exercise scenarios. We used a hierarchical task network structure that was an extension of CTAEMS [2], which is itself a variant of TAEMS [9] developed for the DARPA Coordinators Phase 2 evaluation. The essential property was that tasks (including the root task which represented the overall reward function) were composed of subtasks iteratively until reaching a primitive task which was composed of actions. Tasks could also have non-hereditary relationships such as enabling and disabling. Every task was also associated with state aggregation functions that determined how the state of its subtasks (or child actions) affected the state of the task. An example of a template used to model power substations is shown in Figure 4. This also illustrates the issue of dynamism as the task node for \( \text{Problems} \) must remain without children until the power substation is surveyed for damage. Then, the appropriate problems and repair options are added dynamically to the model. It would be cumbersome and practically infeasible to express every possible combination of problems and repair options that could occur. The issues are similar when it comes to modeling the discovery of injured people.

The TCR set for a given task is calculated by applying a TCR aggregation function, chosen based on the state aggregation function associated with the task, to the TCR sets of its subtasks and enabling tasks. For example, a \( \text{sum} \) state aggregation function would correspond to a \( \text{union} \) TCR aggregation function, and a \( \text{sync} \) state aggregation function would correspond to a \( \text{cross-product} \) TCR aggregation function. Thus, TCR sets would start from actions, which are each associated capability and flow forward and up through enablement and ancestral links to form TCR sets for every task. These sets can be dynamically updated as tasks change states.

For example, once a task is completed, the TCR set can be set to null indicating that it does not require any more capabilities. This makes the TCR sets vanish as their tasks are completed, allowing agents to be released as soon as possible. In order to address the dynamic nature of the model, tasks that might be expanded during execution must be marked with TCR sets that indicate reasonable upper bounds on needed capabilities. These sets are then changed to the actual TCR sets once outcomes have been observed in the environment. Having an ITN-based model helps to construct and manage TCR sets, but is not necessary. As long as there exists a non-cyclical mapping that describes the relationships of tasks to other tasks and actions, a dynamic methodology to assign TCR sets to tasks can be constructed.

4.3 Partial Centralization

S Tac execution can be implemented such that a single agent is responsible for choosing all actions involved with a single task, constraint tuple of a guidance element. We create a mapping, \( \omega : T \rightarrow N \), where every task has an owner. The task owner contacts agents who are responsible for related tasks and actions to
subscribe to relevant state updates. When an agent reaches a particular task-constraint tuple, itcedes autonomy to the owner of that task until the task owner releases the agent from that commitment. The owner agent keeps track of the set of capabilities of all agents bound to that task as well as the TCR set of that tasks and repeatedly solves an optimization problem to find the best set of agents to keep for the current TCR set. If the solution is a set that is a strict subset of the bound agents, it can release the other agents. Our optimization problem minimized a weighted combination of the number of agents kept and their capabilities. The key insight here is that partial centralization of autonomy always occurs implicitly and thus, it is beneficial to align the metric for partial centralization with the properties of the domain where it matters.

5. SANDBOX REASONING

Once the task owner has chosen which set of agents to keep, it must then also decide, subject to the constraints in the guidance, which actions to perform and which agents should perform to accomplish the task. We call this process sandbox reasoning because the task owner’s deliberation over what to do for a single task-constraint tuple is isolated from all actions and tasks that are not related to the task at hand. The task owner does not need to consider impact on the future or on concurrently executing tasks. It is given a collection of agents and autonomy to use them however it sees fit to accomplish the task as well as possible. The consequences of the interactions have, in principle, been considered and addressed by human strategic guidance.

In creating the agent model for a field exercise scenario, we instantiated a structure (sometimes, dynamically during execution) from a small set of templates. Examples include power substation restoration (as shown in Figure 4) or critically injured rescue. Similarly, the tasks in the guidance were also a subset of tasks that make intuitive sense to the strategy designer. In our case, the task types involved in guidance were far fewer than the templates involved in model generation. We believe that the notion of using templates for generation and more significantly for guidance is a general principle that is applicable to many domains. This belief was also reflected in the DARPA Coordinators program and its Phase 2 evaluation. The main templates needed for guidance were a repair and rescue manager. We discuss the automated reasoners in these managers below.

5.1 Repair Manager

The repair manager would take as input (1) a collection of facilities that had been damaged, (2) a set of problems for each facility, (3) a set of repair options for each problem, (4) set of agents with (5) associated capabilities and (6) times that they would be available to be scheduled for activities. The output would be a policy that yielded a collection of agent-action tuples given a simplified version of the state. While this may seem field-exercise specific, this reasoner had no semantic knowledge of the field exercise.

The problem was generalized as follows: Given a set of tasks \( \{T_i\} \), where each task is a conjunction of a set of problems, \( T_i = \min \{\{P_j\}\} \), each problem is a disjunction of repairs, \( P_j = \max \{\{R_k\}\} \), and each repair is a function of actions, \( R_k = \odot (\{a_l\}) \) where \( \odot \in \{\text{sync}, \text{sequence}, \min\} \) is a collection of operators that require the elements to have synchronized start times, sequential execution, or conjunctive success in order for the repair to succeed, respectively. Each action is associated with a capability, expected duration, and probability of failure. We also have a set of agents where each have an associated set of capabilities and an availability time. This is a straightforward optimization given an appropriate objective function.

We needed a fast solution (less than five seconds) because users needed guidance from the solver after performing a survey. Our solution to this was to build a policy based on a simplified representation of state. The state vector is indexed by all possible actions and takes values from the set (NotStarted, Succeeded, Failed). The policy output given for a state is a set of agent-action pairs.

The policy is constructed by running simulation traces of the optimization problem. At every time step in the simulation, if there are idle agents and no action-agent tuples for the current state in the policy, the agents are randomly assigned to an action they can perform and marked busy for the expected duration. These assignments are then associated with the state of the simulation where executing actions are interpreted to have succeeded. The outcomes of the actions in the simulation are determined using the given probability of failure. Multiple simulation runs are used to build a single policy which receives a score based on the average idle time of all agents. Multiple policies are generated using the same mechanism and the policy with the best score is stored. To execute the policy, the task owner maps the current state of the actions to the policy state by mapping executing actions to Succeeded and schedule the associated action-agent tuples as the next action for the agent. If there is no agent-action tuple for the translated state or any of the input parameters (e.g. availability times, new tasks) change, policy generation restarts from scratch. While this is a simple stochastic sampling approach, it produces reasonable policies with very limited computation requirements. Policy generation was typically bounded to five seconds. Also, while the potential state space is exponential in the number of actions, typical policies had at most about 200 states.

The key idea is that we could create a sandbox reasoner that can solve a generic context-independent problem which could be applied to many tasks in the guidance. One could, in theory, use an MDP or STN-based approach if it yielded a solution within the limits of bounded rationality in the domain at hand.

5.2 Rescue Manager

The rescue manager would similarly take as input a list of agents and a set of injured with associated types and deadlines, and output a set of agent-action pairs when agents became idle. We used a simple reactive planner with a handful of simple heuristics to determine when to wait for an agent to help load a critically injured (which requires two agents to be present simultaneously) and when to take a serious (which could be done by one agent). This also can be formulated as a generic problem consisting of a set of tasks with associated deadlines and durations and the tasks can be either a singleton action or require synchronized actions by two different agents. The rules were variations of: “If an agent is on the way and it will arrive later than the duration of the singleton action, perform the singleton action and return to the site, otherwise, wait for the agent.” The variations were due to the constraints placed on the rescue task and the number of agents available to do the rescues.

The general philosophy of the STaC approach to guidance, its execution and sandbox reasoning is to create a generic framework for human strategic input to decompose a very difficult problem into smaller problems that can be solved in isolation with automated tools created to solve large classes of task structures that appear in the guidance. Our system was completely unaware of any semantics of the field exercise, and a similar approach could be used in a completely different domain.

6. EVALUATION

Figure 5 shows the scores for the three scenarios run in Herndon. For each scenario, the top, lighter bar shows the radio-team score,
and the bottom, darker bar shows the score of the team using the CSC system described in this paper. The middle bar shows the results of a simulation of the scenario using a baseline version of the system with simple sandbox reasoners. In order to calibrate the simulation scores, we also ran a simulation of the complete version of the system using the same dice rolls used in the physical scenario. The simulation results for the full system were within 200 points of the results obtained in the field, which suggests that if the baseline system had been used in the field, the results would also be close to those shown in the figure.

In the baseline version, the repair manager uses a random strategy where agents randomly select a repair task to perform from the set of tasks that the agent is eligible to perform according to its capabilities. The baseline rescue manager uses a greedy strategy where agents select the repaired with the earliest deadline that lives long enough to arrive to the medical facility before the deadline. In the baseline rescue manager, agents don’t wait for a partner that enables loading a critically injured that they would not be eligible to load otherwise.

The results show that more sophisticated sandbox reasoners always resulted in better scores: 5.8%, 8.4% and 24.8% improvements. The differences in scenarios 1 and 2 were small, and in those scenarios the baseline system also outperformed the radio team. In scenario 3, the difference is more significant. This scenario emphasized injured, and the greedy strategy used in the simple version of the system delayed rescuing the critically injured. Agents rescue seriously injured with later deadlines instead of waiting for a partner to rescue a more valuable critically injured with an earlier deadline.

Figure 6 shows simulation results that compare the effects of alternative strategies. We organized these strategies along two dimensions: the number of clinics that would be made operational (0, 1 and 2), and the number of independent subteams (1, 2 and 4). In the strategies with 1 clinic, the team repaired the clinic that was considered most useful (closest to the most valuable injured). In the scenarios with 2 and 4 teams, we specified the teams so that they could perform repairs independently. In the strategies with 0 clinics, the teams performed no repairs and rescued injured to a medical facility that was always operational and required no repairs. In the strategies with 1 and 2 clinics, the agents first repair the main stations, then the clinics and then visit the remaining sites to rescue all injured and perform all repairs according to the following algorithm. First, the sites are ordered according to the total expected number of points achievable at the site. The teams take turns picking the next most valuable site from the ordered list until the list is exhausted. The idea is to complete the most valuable sites first so that when time runs out the most valuable sites have been completed.

Figure 6 shows that in the Herndon 1 and 2 scenarios, the strategies that repair 1 or 2 clinics are competitive with the radio team, outscoring them in 11 out of the 12 strategies involved. However, in all three scenarios, the CSC strategy used in the field was significantly better than all the alternative strategies. The difference is due mainly to the use of constraints. In the alternative strategies, the agents performed all tasks at a site, whereas the strategies used in the field used constraints to prevent agents from performing tasks that we deemed not worthwhile. In addition, we used constraints on the number of injured rescued to prevent agents from rescuing all injured at a site before moving to the next site. Instead, we used longer itineraries that visited sites multiple times in a round robin, so agents would rescue the most urgent injured first.

7. CONCLUSIONS AND FUTURE WORK

Our 18-month experience working on a system to compete against radio teams in the field exercises provided significant evidence for the benefits of our approach. Our starting point was our generic CSC system developed during the previous two years to solve generic, synthetically generated problem instances specified in CTARMS. Even though the synthetically generated problem instances were generated according to templates that combined "typical" coordination situations, the resulting problems were not understandable by humans. In contrast, the field exercise problems are natural, and appeal to our lifetime of experience coordinating every day activities. Intuitions about space, distance, time, importance and risk all came into play, enabling teams of humans to devise a sophisticated strategy within one hour of brainstorming. It became obvious early on that the generic CSC system would not be able to produce solutions comparable to the desired sophisticated, coordinated behavior of human-produced strategies.

Our existing system had performed extremely well in Phase 2 by using our Predictability and Criticality Metrics (PCM) approach. In the PCM approach, the policy modifications that agents consider are limited to those that can be evaluated accurately through criticality metrics that capture global information. These policy modifications were simple and thus the reasoners that implemented them were simple too.

For the field exercises, we extended our approach so that policy modifications would be constrained using the guidance provided by the users. This guidance was in the form of a sequence of sites to visit. The system was left to make decisions that we believed it could evaluate accurately (e.g., how to perform repairs or rescue injured at a single site). The system relied on the TCR set criticality metric to determine how to move agents along the list of guidance elements. The approach worked well. Our users outperformed the radio team because they were able to communicate their strategy to their agents, and the system optimized the execution of the strategy, adapting it to the dynamics of the environment.

The field exercises in Rome, NY used a simpler language for specifying guidance. It had a single guidance group consisting of the entire set of agents. Also, it did not support constraints to control the capabilities within a guidance element. In that evaluation, our system remained competitive with the radio team, but lost in two out of the three scenarios. The final language for guidance was inspired by our observations of the radio-team strategies, extensive discussions with subject matter experts and extensive numbers of simulations. We noted that while the human team could not execute a strategy as well as we could, the space of strategies that they were able to engage were far more sophisticated than ours. This led to the creation of a more sophisticated formalism for capturing human strategic guidance.
8. REFERENCES


