KTH Computer Science
and Communication

RoboCup 2D Soccer Simulation League

An analysis of the team WrightEagle and the creation of a basic team

PETER NYCANDER
PETERNYC@KTH.SE
+46 70 377 04 29
FORSKARBACKEN 11
114 15 STOCKHOLM,

TOBIAS ANDERSSON
TOBIAS2@KTH.SE
+46 70 630 96 25
UPPLANDSGATAN 74
113 44 STOCKHOLM

KTH School of Computer Science and Communication
DD134X - Degree Project in Computer Science, First Level
Supervisor: Pawel Herman
Examiner: Mårten Björkman

29. maj 2013

Abstract

RoboCup 2D Soccer Simulation League is an international artificial intelligence (AI) competition in which computer programs compete in soccer.

This report will present the work of analyzing the successful team WrightEagle, and the isolation of the key strategic and behavioral aspects that make them successful. It will also present the work of creating a new team in which the aspects exctracted from WrightEagle has been implemented, but in a much simpler way.

It was found that good passing play, good stamina preservation and the ability to stay spread out were WrightEagle's key strategic and behavioral aspects. While these aspects has been implemented in the created team, the lack of good core functionality proved to be a more important factor and the resulting team performs poorly.

Referat

RoboCup 2D Soccer Simulation League är en internationell tävling i artificiell intelligens (AI), där datorprogram tävlar i fotboll.

Den här rapporten kommer presentera arbetet i att analysera det framgångsrika laget WrightEagle, och att hitta de aspekter vad gäller strategi och betéende som gör dem så framgångsrika. Den kommer också att presentera skapandet av ett nytt lag som implementerar de funna aspekterna från WrightEagle, men på ett mycket enklare sätt.

Bra passningsspel, bra uthållighetsbevarande, och deras förmåga att hålla sig utspridda var de funna aspekterna från WrightEagle vad gäller strategi och betéende. De här aspekterna har blivit implementerade i det skapade laget, men bristen av bra basfunktionalitet visade sig väga mycket tyngre och det resulterande laget preseterar dåligt.

Statement of collaboration

The majority of the work was done collaboratively by both authors. It was almost always done by working together in the same room, whether or not the authors worked on different tasks or on the same task together. There was therefore no clear division of work. In order to simplify working on the code together, git was used.

The report was written collaboratively in the same manner using Google Drive.

Two meetings with the supervisor, Pawel Herman, were arranged. The rest of the assistance and communication was done via e-mail. The assistance mainly concerned the report and its structure.

Contents

Chapter 1

Introduction

RoboCup soccer is an international robotics and artificial intelligence (AI) competition in which robots or computer programs compete in soccer. The official objective of RoboCup is to promote AI and robotics research. The creators' long term vision is that by the middle of the 21st century, a team of autonomous humanoid robots shall win against the human World Cup champions in a game of soccer.[1]

While getting robots to play soccer does not in itself yield any significant impact on society, it would be a great achievement in the field of AI and robotics. Furthermore, the knowledge gained might become useful in a more productive way in the future.

RoboCup also creates a very measurable way of improving AI algorithms by using competition as a tool. It is considered that competition creates a very innovative environment, and innovation is the key to development.

This type of project, which has a very attractive and broadly appealing goal, but does not have any significant gains, is called a landmark project[10]. The Apollo space program which had a goal of "landing a man on the moon and returning him safely to earth" is a successful example of such a project.[1] Another example, more closely related to RoboCup, is IBM's chess-playing computer, "Deep Blue". Deep Blue defeated the human world champion in chess in 1997, the same year as the first official RoboCup tournament [7]. RoboCup can be seen as the successor to computer chess, but with some fundamental differences. For example computer chess is based on centralized decision making, while RoboCup uses distributed decision making, which increases the difficulty of coordination.[6, p. 1]

There are five different leagues in RoboCup soccer: humanoid, middle size, small size, standard platform, and simulation, each with a number of subleagues. This project concerns the the 2-Dimensional (2D) subleague of the simulation league, and the term RoboCup will therefore from now on refer to this subleague. In the simulation league there are no physical robots, instead a team consists of 11 autonomous computer programs (from now on referred to as agents or clients). The agents connect to a server which simulates the game. An independent monitor application must also be used to display the game. A screenshot of a typical match

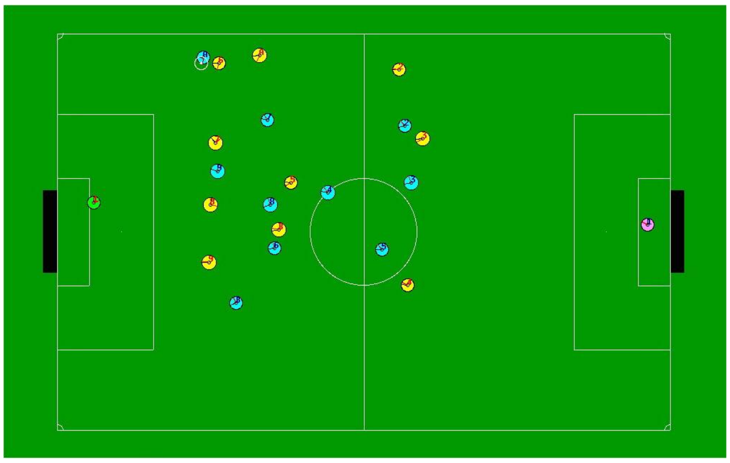

Figure 1.1. Screenshot of a RoboCup 2D simulation league match displayed in the standard monitor.

can be seen in figure 1.1.

In the 2D subleague there are two dimensions, one following the X-axis and one following the Y-axis, which greatly simplifies the environment compared to the 3D-league. Because of the simple environment, the focus in this league is team strategy and advanced AI algorithms, rather than dealing with hardware issues such as making robots move and kick the ball properly while maintaining balance.

As previously mentioned, each player in the RoboCup is controlled by its own independent program with its own control mechanisms and input parameters. A consequence of this is that each player only has access to the information in its own field of vision. This makes coordination of the players and team strategy very complex.

1.1 Problem statement

The main aim of this project is to find important strategic and behavioral aspects of one good RoboCup team, and then to find simpler ways to implement these aspects into a RoboCup team.

The first part of the project is to study a successful RoboCup team and to analyze the following:

- What are the key strategic and behavioral aspects that make them successful?
- How do they implement these aspects?

The problem investigated in the second part of the project is: is it possible to create a team which implements key strategic and behavioral aspects of an advanced, successful team, in a much simpler way, without machine learning or complicated algorithms?

A team will therefore be created based on the results of the first part of the project. The goal is that this team will feature the key aspects found in the analysis (to some degree). The resulting team will be matched against other teams in the simulator and an analysis of the its strenghts and weaknesses will be made.

1.2 Approach and Scope

Three different methods will be used in order to provide a basis for the analysis of design, behavior, and strategy of an exisisting RoboCup team: watching videos of recorded matches, studying code, and reading other online material, which will to a majority be the team's own material. The simulations will be run in the official RoboCup simulator which is downloadable from RoboCup's official sourceforge site. [8]

Because many teams have lacking documentation, the report will focus on the Chinese team WrightEagle which is very successful and has relatively good documentation. [2]

A team will then be created, based on the results from the analysis mentioned above. Since communication between the client controlling an agent and the simulator is implemented via sockets, the client can be written in many different languages[9]. In this project it will be written in java.

1.3 Report outline

The report outline is as follows:

- Background presents the details of how RoboCup simulation works, in order to facilitate understanding of the team creation process for the reader.
- Methods presents the methods used while studying and analyzing the chosen team in order to give understanding of their implementation. That understanding will be required in order to find the key behavioral and strategic aspects and how they are implemented. Methods also present the team creation process.
- Results presents the results from the video analysis and analysis of the other material, the implementation of the above mentioned aspects and then the performance of the created team.
- Discussions and conclusions presents discussions regarding the performance and implementation of the studied team and the newly created team. It also presents the conclusions drawn from the results and analyses.

Chapter 2

Background

This chapter presents the details of how RoboCup works, in order to facilitate understanding of the team creation process for the reader.

In order to simulate a game of RoboCup soccer, a server and two teams consisting of independent agents are required. As previously mentioned an independent monitor application is also needed in order to watch the game, but the technical details regarding the monitor is out of the scope of this project.

2.1 The server

The purpose of the server is to provide a virtual soccer field on which the game takes place. To fulfill this purpose the server:

- holds all information about the game state, for example the score and the position of the ball and players.
- sends sensory information to the agents (explained in section 2.2 below).
- handles and responds to requests sent by the agents.
- runs the simulation by continuosly updating the game state.
- provides a virtual referee which enforces the rules.

2.2 The Agents

Each RoboCup team consists of 11 player agents. Each agent controls the action of a single player and communicates with the server by sending requests about what it wants to do, for example turn, kick or run. The server then handles the request and updates the state accordingly. The server also sends information to the agents' three sensors: the body sensor, the aural sensor and the visual sensor.

The body sensor provides information about the agent like its stamina, current speed and in which direction it is looking. The aural sensor detects messages sent by the server (the referee), the coach and other players. The visual sensor provides information about the game state, such as the position of the ball and other players. Agents only receive visual data about what is in their own scope of vision. Additionally, visual noise is applied, which means that the distance to objects received from the server is only an estimate of the real distance. The actual distance can deviate about 10% from the received distance.[6, p. 11, 32]

As previously mentioned, the server holds information about all of the agents. Except for the obvious information such as position, direction, and speed, it contains stamina which is of importance when it comes to strategy. Stamina is an integer value which decreases when an agent dashes, and increases slightly each cycle. If an agent does not have enough stamina for a given dash command, the dash command will have its power reduced, resulting in reduced speed. [6, p. 36-37].

Apart from the 11 player agents each team can also connect a coach agent to the server. The coach can communicate with the players and receives noise-free information about everything on the field. This enables the coach to analyze the game and give commands to the players. The communication is restricted, in order to prevent coaches to control their teams in a centralized manner.

There is also a different kind of coach which cannot be used in official games, called the “off-line coach” or the “trainer”. As the name suggests it is used during development and is much more powerful than the regular coach. For example it can move objects on the field and provides addidional tools such as automated training sessions for machine learning. [6, p. 85]

2.3 Communication

The communication between the server and the clients is implemented via an UDP/IP socket. As a consequence the clients can be written in any programming language which supports such communication. [6, p. 3]

The actual information sent between the server and clients consists of strings, which form requests to the server or sensory information to the client. The requests follow the format: (command parameter1 parameter2 ...). Sensory information follows the same format: (message-type value1 value2 ...). Values can either be literals (which is typically a number or a keyword) or sub-messages (which has the same format as sensory information). For example a typical “see”-message from the server describing what the agent currently sees could look like:

```
(see 240 ((flag r b) 33.1 -23 -0 0) ((flag p r b) 17.8 13 -0 0)
((ball) 0.5 -25 0 -0.2) ((Player) 1 122) ((Player) 2.7 101)
((Player) 2.2 69) ((Player) 1.1 90) ((Player) 0.6 71)
((player TeamOne 3) 1.3 14 -0 0) ((player TeamOne 5)
1 42 -0 0) ((line b) 30.9 87))
```

In the example above, the agent sees two “flags”, the ball, seven players and one line. Flags are placed on certain points around the field and help the agents to orientate themselves. Lines represent the outer lines around the field and serve the same purpose as flags. Note that the first five player entries contain less information than the last two. That is because the agent is unable to identify which player it sees. The number values in each entry can for example represent distance and direction to the object.[6, p. 27]

2.4 Rules

Each match is divided into 6000 cycles. Each cycle lasts 100 milliseconds making a match 10 minutes long. During every cycle each agent is allowed to perform one action (with a few exceptions; an example is turning the agent's head, which can be performed several times during a cycle in order to provide a tool for gathering information while the agent is performing other actions).

The virtual referee automatically controls the game-mode and enforces most rules, but a human referee is needed to detect violations against "fair play"-rules. Violations against "Fair play" are for example surrounding the ball, blocking the goal with too many players, intentionally blocking other players etc.[6, p. 12-13]

Chapter 3

Methods

This section presents the methods used while studying and analyzing the chosen team in order to give understanding of their implementation. That understanding will be required in order to find the key behavioral and strategic aspects and to understand how they are implemented. This section also presents the team creation process.

3.1 Choice of team

The team that was chosen for this project is a Chinese team called WrightEagle. The reasons for this choice are many. First of all they are very successful; they have been among the top eight teams in every official RoboCup tournament since year 2000 (except for 2005 when they did not participate). Even more impressive is their performance since 2005; they won the tournament three times, in 2006, 2009 and 2011 and finished second the other five years. [4]

Apart from their great performance, WrightEagle also have good documentation, they have released a publication called "team description paper" every year on their website describing their improvements since last year. Their source code is also downloadable from their official website. [2]

3.2 Video analysis

The main method used in order to extract the behavior of WrightEagle has been analyzing recorded videos of their tournament matches. The video analysis is done by observation and discussion amongst the authors of this report. The resulting analysis is presented in section 4.1.

3.3 Source code outline

In order to give understanding of the implementation of WrightEagle, the source code has been studied. This section will present a summary of knowledge gained by studying of WrightEagle's source code. The version of the source code used is the WrightEagleBASE-4.0.0 package which is available from WrightEagle's official website[2].

The code is written in C++, and is divided into over 100 files. There are more than ten files with over 1000 lines of code, but there are a few classes that are considered more important than the others and will therefore be the main focus of the study. These are: WorldState, Parser, Agent and ActionEffector.

WorldState, as the name suggests, contains all the information about the playing field that an agent has available. This includes the position of the ball, all players that can be seen, and possible locations for players that are out of line of sight. The information stored in WorldState is constantly updated, in order for the agents to have an accurate picture of the playing field.

As described in section 2.3, there is a lot of string communication between the server and the agents. Parser is in charge of parsing all information sent from the server and to make it accessible for the other classes. Parser is running in its own thread in order to be able to constantly read the information sent by the server.

Agent is a class which represents an agent on the field. It wraps around a lot of code in ActionEffector into more logically named, more readable functions. One such example is the Agent.Dash(), which wraps ActionEffector.SetDashAction(). An instance of Agent is used as a parameter of several functions in the code, making it a key class.

ActionEffector is the key class for an agent's behavior. As mentioned above, most of the functions of the Agent class are simply wrappers for the functions in ActionEffector. Therefore, it affects almost all of the agent's behavior. ActionEffector then uses even more technical and advanced functions in the respective classes for each action such as the Tackler class, the Dasher class, the Kicker class, etc. ActionEffector also contains a queue of commands that will be sent to the server.

3.4 Study of team description papers

In order to give insight into the strategic reasoning and implementation by the authors of WrightEagle, a study of their team description papers has been done. Team description papers are released by the team authors every year, and describes the research done during the past year. This section presents a summary of the knowledge gained by studying these papers.

3.4.1 Decision making

The decision making has at least since 2007, been modeled after a Markov decision process (MDP). An MDP is an extension of a Markov process in which states, actions and rewards has been implemented. In the scope of this report, it can be thought of as the agents having their own Markov decision process. In the agents' MDP, each state would be a moment in time, the set of actions would be the agents' possible actions like kick, dash or to wait. The rewards for each action is the programmers way to reinforce behavior that is generally best to take, like waiting to preserve stamina. The reward then increases or decreases the probability of the agent preforming the same action the next time the agent is in a similar situation. [30] [28]

Between the years 2007 and 2010, the authors have experimented with several different modifications to the regular MDP model in order to better adapt it to model different actions performed in RoboCup. One modification was to make the agents handle the uncertainty of observation. [29] [27] Another modification was to introduce absorbing states (which could be an agent losing the ball) in order to better give negative rewards. [26] The details of these modifications are out of scope of this report.

3.4.2 Stamina preserving

In order to preserve stamina, the authors of WrightEagle have introduced a couple of techniques over the past few years.

In 2009 they started using the coach agent to monitor the agents' stamina since it receives information without noise, as stated in section 2.2. The coach can then help the team preserve stamina in different ways, for example to tell agents how long to wait before kicking the ball during a kick in, in order to let the agents with the least stamina recover it, but not longer. [27]

In 2011 they implemented dynamic formations, which means that an agent can switch role at any time. This can be used to preserve stamina by, for example, letting the coach order a forward which has low stamina to switch positions with a defender which has high stamina.

3.4.3 Dribbling

In 2007 the authors of WrightEagle introduced a new method of searching for a target destination while dribbling. Their old algorithm used exhaustive search on a few different angles. This was inefficient, and they replaced it with a greedy algorithm that evaluated the best area out of five predetermined ones. The algorithm then does the same procedure recursivly for the selected area until it has selected a point. [29]

In 2010 they improved their dribbling model with a modified MDP which has absorbing states in order to increase the dribbling speed, and changed the algorithm from their previous greedy implementation, to the known A*-algorithm. [26]

3.4.4 Other important steps

In 2008 the creators of WrightEagle decided to increase the number of running threads from two (a parsing thread and a decision thread) to three (a parsing thread, a decision thread, and a transmission thread). The new thread was implemented in order to send commands at an optimal pace. The change allowed the agents to use more time for calculation than before. [28]

3.5 Team design and implementation

The basic layout of the source code was taken from the study of the code in section 3.3. A three thread system was chosen, one parsing thread, one transmission thread, and one decision thread. The parsing thread is responsible for updating the world state, which has information available for the decision thread, which in turn runs the agents logic. The transmission thread is, as the name suggests, responsible for sending requests to the server at an optimal pace.

3.5.1 Source code language

The implementation was written in Java. Java has great classes to use when it comes to network communication, which is important in this kind of project. It is also platform independent, which is desirable if the code needs to be run in a new environment, like when competing in a tournament.

3.5.2 Network communication

In order to establish communication between the agents and the server, the class java.net.DatagramSocket was used, as shown in code 3.1 and code 3.2. DatagramSocket communicates over the UDP protocol.

Code 3.1. The code the agents use in order to send a message to the server

```
sendMessage(String message) {
  message += "\u0000"; //Messages are null terminated
  byte[] buf = message.getBytes();
  DatagramPacket msg = new DatagramPacket(buf,
   buf.length, Constants.Server.IP, Constants.Server.PORT);
  socket.send(msg);
}
```

Code 3.2. The code the agents use in order to receive a message from the server

```
byte[] receiveData = new byte[4096];
DatagramPacket p =
new DatagramPacket(receiveData, receiveData.length);
socket.receive(p); //This is the same DatagramSocket used for sending
if (!hasBeenInitialized) { //After initialization,
              //the server may send a different port and
              //address to communicate on
  Constants.Server.PORT = p.getPort();
  Constants.Server.IP = p.getAddress();
  hasBeenInitialized = true;
}
String message = new String(p.getData());
parse(message); // This will update the world state
```

3.5.3 Parsing

The incoming messages are being parsed by using regular expressions and substrings. Since each cycle is 100 ms long, it was considered acceptable from a performance point of view to simply put all incoming information in to maps for later usage. For example there are maps for: distance to objects, angles to objects, and time since objects were last observed.

3.5.4 Implementation of key strategic and behavioral aspects

In order to implement the key strategic and behavioral aspect into the created team, different approaches were used for different aspects. The core idea in each approach is simplicity, each aspect has been implemented in a way that the authors of this project has deemed to be most simple. The resulting implementation is presented in section 4.3.1.

Chapter 4

Results

This chapter presents the results from the video analysis and analysis of the other material, and then the performance of the created team.

4.1 Video analysis

This section presents analyses of three of WrightEagle's matches in RoboCup 2011.

4.1.1 WrightEagle vs HELIOS2011, match 1

The analysis presented here is of the grand final of RoboCup 2011 and is the first game of two between WrightEagle and HELIOS2011 that was analyzed.

It was clear that WrightEagle had superior passing play, they could make accurate, fast, passes even in very crowded areas. HELIOS2011 were more reserved regarding when they tried to pass and seemed to prefer not to pass unless they had a safer opportunity. The HELIOS2011 player often ended up losing ball possession because of the hesitance to pass.

As a result of this WrightEagle had more ball possession and created more goal opportunities, but they could have scored more goals if they capitalized more on the opportunities that they created. The final score was 3 - 2 in favor of WrightEagle.[16]

4.1.2 WrightEagle vs HELIOS2011, match 2

The analysis presented here is of the winners bracket final of RoboCup 2011.

WrightEagle were not as dominant as in the previously analyzed match. Most of the time they could not get the same flow in their passing play, and as a consequence of that, the ball possession dropped.

WrightEagle showed the same hesitance to shoot on goal as in the first game. They still managed to score two goals, both were made in the same way: by using their superior passing play in front of the opponent's goal until they found a gap in the defense of HELIOS2011 which they could capitalize on. The final score was 2 - 1 in favor of WrightEagle.

One reason to what caused the change from the game analyzed above is that random elements might not have favored WrightEagle's aggressive passing play as much this match. Another is that WrightEagle could have improved their team between the two matches (this game occured earlier in the tournament than the game analyzed above). [17]

4.1.3 WrightEagle vs AbouAliSina

This section presents an analysis of a match that was played during group play of RoboCup 2011.

There was a clear difference in amount of work put into the two teams which could be seen right from the start. WrightEagle completely dominated the game, which is obvious from merely seeing the final score, which was 17 - 0 in favor of WrightEagle.

AbouAliSina had no chance to keep up with WrightEagle's fast paced passing play. That combined with that WrightEagle stayed spread out while AbouAliSina often clumped up, let WrightEagle take complete control of the game.[18]

4.2 Strategy analysis

In this section some final analytical remarks regarding team strategy will be presented. These results are based upon studies of WrightEagle's code and team description papers.

The study of the code did not reveal any strategic choices. The “Strategy.cpp” file, which one would assume contains strategy, is perceived to only contain functions that are designed to be tools in order to create algorithms, rather than actually containing a clear team strategy. These functions could however contain strategic choices such as where the agents should position themselves in regards to the rest of the team. In that case the strategy is well hidden within the code and is thereby difficult to find by simple code study.

The coach is however considered to play a key strategic role, since it will give instructions to the player agents. From the team description papers it is known that the coach can change the roles of the player agents, and similar actions. This is considered to be team strategy and the coach is therefore found to be an important part of the strategy implementation. It is noteworthy that the code study did not reveal how the coach works, or even if it works.

4.3 Strategic and behavioral aspects of WrightEagle

This section presents the three strategic and behavioral aspects of WrightEagle. The way they are impemented in the created team is presented in section 4.3.1 below.

The video analysis showed that WrightEagle had more ball possession than the opposing team in every match. It was concluded from the study of team description papers that part of the reason is due to their dribbling algorithms and the from the video analysis that part of the reason is due to their accurate passing. Therefore it was found that ball possession due to solid dribbling and high accurcy passing is the first aspect.

The video analysis also showed that the WrightEagle agents cooperate well, the agents never clump up like the agents of the others teams did. Instead the agents positioned themselves to interfere if the opposing team took the ball. It proved to be a successful strategy. Therefore it was found that the fact that the agents to not clump up is the second aspect.

When studying the team description papers, it is evident that a lot of effort has been put into preserving stamina. With this in mind during the video analysis, it was noticable that WrightEagle agents rarely runs out of stamina. The fact that they outperform their opponents in stamina preservation seemed to be an important factor to win the games. Therefore it was found that good preservation of stamina is the third aspect.

4.3.1 Implementation of key strategic and behavioural aspects

In order to try to keep high ball possession extra effort was put into passing, as dribbling was found to be more difficult. To make an accurate pass the agents need to kick it with the proper angle and power. The angle used is the same as the parsed angle to the agent, unless the agent is moving. In the case of moving agents the angle is calculated using the distance to the target and the direction it is facing.

The way proper passing power was achieved was by trial and error during practice matches. The agents were observed as they passed each other and the formula was changed in accorance with the error in passing power. It was found that the needed power (power ranges from 0 to 100) is almost linear to the distance (in meters), but not linear enough to use a single simple formula. Instead of using interpolation from the successful passing powers and deriving a formula from the results, a simple if-else-clause was used with different modifying values to change the power for different distance intervals.

To make sure agents don't clump up, a very simple approach was used. There are a lot of flags placed on the field, and every agent has their own flag designated to them, which it will run to and wait for the ball to get within a specified range. This way they are not clumped up when they are not around the ball. When an agent determines whether it should run to ball or not, it will first check if there is already a friendly player with a close angle and distance to the ball; in that case, the agent will stand a few meters away in order to take up the action if the ball-owning agent loses the ball.

In order to preserve stamina and to help the ball possession of the team, the decision making process has a very high priority to pass other players instead of running with the ball. This, combined with agents standing at the most forward flags, which are available to score goals, will make sure stamina is preserved.

4.4 Performance of the created team

This section presents the results and observations from matches played with the created team versus other teams and versus itself in the simulator. A summary of the performance then follows.

4.4.1 Simulation vs WrightEagle

The simulation was run between the created team and WrightEagle. WrightEagle won with the score of 34-0. The only time the created team touches the ball is during kick-offs and kick-ins. The WrightEagle agents seemed to not only play better, but move much faster and have a lot more information about the game state.

4.4.2 Simulation vs AbouAliSina

The simulation was run between the created team and the 2011 version of AbouAliSina. AbouAliSina won with 16-0. The ball was on AbouAliSinas side four times during the entire game, and the ball possession was mostly AbouAliSina's. AbouAliSina seemed to move faster and execute commands at a more optimal pace. However, the agents acting defenders in AbouAliSina were taken by surprise the few times the created teams was on the offensive, and with luck the created team could score.

4.4.3 Simulation vs self

The simulation was run between two identical versions of the created team. Obviously, the game was very even. After watching other teams play, it is clear that the agents of this team move and act slower than agents of other teams. The final score was 4-5.

4.4.4 Summary of performance

The performance of the created team is bad compared to any standards. When playing against a real team, the created team barely touches the ball at all. The overall reason is the low speed of the created team. The passing is rather good, but not the receiving of passes. The passes often get interrupted.

Chapter 5

Discussion and Conclusions

This chapter presents discussions regarding the performance and implementation of the studied team and the newly created team. It also presents the conclusions drawn from the results and analyses.

5.1 Work summary

In this project, the team WrightEagle has been studied and analyzed. WrightEagle are very successful, and the from the analysis it was considered that their success are based on three strategic and behavioral aspects: High ball possession due to solid dribbling and high accuracy passing, spread out agents, i.e. they do not clump up, and good preservation of stamina. A team was then created, with the aim to implement the results from the analysis into the team. The team partially implemented these aspects, but had poor performance overall.

5.2 Project limits

Because of the inherited limits of this project, a lot of the time dedicated to the project was spent on creating the basic outline of functionality, most importantly communication between agents and the server. It was time consuming to make the agents understand where they were and to make sure agents were using updated information for their decisions. This led to a very limited amount of time to actually implement the strategical and behavioral aspects. The fact that programming basic functionality was time consuming paired with the limited time dedicated to the course led to a very hasty coding process. The hasty process led to the error described in section 5.4, and reduced performance overall.

It is important to note that the analyses made were based on the authors subjective perception of the studied content, which means that someone else could draw different conclusions from the same content and would thereby extract different strategic and behavioral aspects.

The aspects found in the analysis might be perceived as unique to WrightEagle simply because the other teams are worse, and the found aspects are simply indications of a team that is winning. However, worse teams did not all seem to implement every found aspect. Therefore the extracted strategic and behavioral aspects are considered to be valid.

5.3 WrightEagle performance and implementation

WrightEagle is an old and complex team. With a developing process like the one WrightEagle uses which has lasted several years, the code becomes more refined and contains less bugs every year. The team is being developed by a university, with new students improving it every year [3]. If one assumes that each new student has a responsibility to improve the team in some way, it implies that improvement is being forced in a way that is hard to achieve if the same people is working on it every year.

During the study of the code in section 3.3 and study of the team description papers in section 3.4 it was partially learned how WrightEagle implement the strategic and behavioral aspect this project considers key.

The solid dribbling found in the first aspect was implemented by WrightEagle by using the A*-algorithm, combined with modelling with Markov decision processes (MDPs).

The reason to their high accuracy passing was not found. It is assumed that it is being modelled by a MDP, which provides good results. The reasoning behind the assumption is that since the team description papers describe work in modelling a lot of different actions with MDPs (but not explicitly passing), they probably use it to model passing as well.

The techniques used to preserve stamina seem unclear. The team description papers describe work in letting the coach help the players conserve stamina by, among other things, letting them switch positions. This was however not observed during the video analysis, which leads to confusion. Since they seemed to have less problems with stamina than comparable teams, it leads to the conclusion that stamina preservation must permeate the entirety of WrightEagle's code, more so than the comparable teams.

The way WrightEagle agents stay spread out has not been found. It is assumed with the same reasoning as above that it is a result of MDP modelling.

5.4 Performance of the created team

The created team has low performance, as stated earlier. An important reason of the low performance is, as previously stated, the low speed. After reviewing the results it has been concluded that an error in the code architecture has been made. The agents are only allowed to run their logic if they have received a fresh 'see'-

message from the server. The ‘see’-message is sent every 150 ms, while each game cycle lasts 100 ms [6, p. 48].

In order to fix the error mentioned above, the world state needs to be updated with estimations of new positions of all objects after each cycle. This would allow agents to make informed decisions even without a new ‘see’-message. The fix is by no means trivial, as it introduces a number of problems, one example would be the concurrency issues when an agent is parsing angles from a ‘see’-message while it is updating the angles from this turning action. Another example of a problem that needs to be handled while fixing the error is updating angles in accorance with the distance travelled.

The speed issue is far from the only problem, other issues like the lack a system good way to determine the agents’ positions, unoptimized path finding and general limitations of our simplified model also reduces performance greatly.

However, it is important to remember that the goal was never to create a perfect team and due to the limitations of this project which are discussed in section 5.2, the project is by no means considered a failiure. The question in the problem statement which was attempted to be answered by the creation of a team: “Is it possible to create a team which implements key strategic and behavioral aspects of an advanced, successful team, in a much simpler way, without machine learning or complicated algorithms?”, was successfully answered. The answer drawn from the results is that while it is possible, it does not guarantee any performance similarities between the successful team and the created team.

5.5 Conclusions summary

The key strategic and behavioural aspects found in WrightEagle were: high ball possession due to solid dribbling and high accuracy passing, spread out agents and good preservation of stamina. In WrightEagle these aspects are implemented by using the A*-algorithm, MDPs, and other techniques.

The three key strategic and behavioural aspects found in WrightEagle were partially implemented into the created team. Focus was put on passing, spreading out agents and conserving stamina. However, the team did not have good core functionality, and as a result the created team was unable to compete with teams that do have good core functionality, and could therefore not achieve high ball possession.

It is concluded that it is possible to create a team which implemented WrightEagle’s key strategic and behavioural aspects using a simple approach, but it remains unsesolved whether it is actually a good idea, and to what degree it limits the performance of a team with good core functionality.

5.6 Future work

To continue the project, the first task would be to fix the error described in section 5.4. This will also, if done correctly, improve the overall available information for the agents greatly. The next step would be to improve the world state to include calculated coordinates, in part to order to create better slot positions. Another problem that needs to be solved in current version is to avoid off-side. To implement off-side avoidance one would need to improve the world state with an estimation of all enemy players position. These changes would lead to a lot more tools when designing algorithms for actions such as dribbling, and would lead to an improvement in the performance of the team.

Bibliography

[1] Objective [Internet]. [cited 2013 Mar 7]. Available from: `http://www.robocup.org/about-robocup/objective/`

[2] University of Science and Technology of China. WrightEagle 2D Soccer Simulation Team [Internet]. [updated 2013 Jan 1; cited 2013 Mar 18] Available from: `http://wrighteagle.org/2D/`

[3] University of Science and Technology of China [Internet]. 2009 [cited 2013 Apr 11] Available from: `http://wrighteagle.org/en/people/index.php`

[4] Soccer Simulation League [Internet]. [updated 2013 Mar 23; cited 2013 Apr 4]. Available from: `http://wiki.robocup.org/wiki/Soccer_Simulation_League`

[5] RoboCupSoccer: Simulation League: Teams [Internet]. [cited 2013 Mar 18]. Available from: `http://www.robocup2012.org/comp_RCSoccer-simLe-Teams.php`

[6] Chen M, Foroughi E, Heintz F et al. RoboCup Soccer Server [Internet]. 2002 Aug 2 [cited 2013 Mar 15]. Available from: `http://wwfc.cs.virginia.edu/documentation/manual.pdf`

[7] RoboCup: The Robot World Cup Initiative [Internet]. [cited 2013 Mar 14]. Available from: `http://www.sonycsl.co.jp/person/kitano/RoboCup/RoboCup-old.html#ROBOCUP97`

[8] Hidehisa Akiyama, Oliver Obst, Hedayat Vatankhah et al. The RoboCup Soccer Simulator [Internet]. [updated 2013 Feb 21; cited 2013 Apr 8]. Available from: `http://sourceforge.net/projects/sserver/`

[9] Users manual [Internet]. [updated 2009 Jul 31; cited 2013 Feb 1]. Available from: `http://sourceforge.net/apps/mediawiki/sserver/index.php?title=Users_Manual/Overview#The_Server`

[10] A Brief History of RoboCup [Internet]. [cited 2013 Mar 18]. Available from: `http://www.robocup.org/about-robocup/a-brief-history-of-robocup/`

[11] Tavafi A, Nozari N, Vatani R, Yousefi MR, Rahmatinia S, Pirdir P. MarliK 2012 Soccer 2D Simulation Team Description Paper [Internet]. 2012 Nov 3 [cited 2013 Mar 9]. Available from: `http://ebookbrowse.com/tdp-marlik-pdf-d416493819`

[12] Partially observable Markov decision process [Internet]. [updated 2013 Mar 14; cited 2013 Mar 22]. Available from: `http://en.wikipedia.org/wiki/Partially_observable_Markov_decision_process`

[13] Losers' Final : RoboCup2011 Soccer Simulation 2D League [Internet]. 2011 Jul 10 [cited 2013 Feb 20]. Available from: `http://www.ustream.tv/recorded/15906843`, 2013-02-20

[14] Losers' Pre Final : RoboCup2011 Soccer Simulation 2D League [Internet]. 2011 Jul 10 [cited 2013 Feb 20]. Available from: `http://www.ustream.tv/recorded/15906462`, 2013-02-20

[15] Second Round Group G : RoboCup2011 Soccer Simulation 2D League [Internet]. 2011 Jul 8 [cited 2013 Feb 20]. Available from: `http://www.ustream.tv/recorded/15865123`, 2013-02-20

[16] Final Match : RoboCup2011 Soccer Simulation 2D League [Internet]. 2011 Jul 10 [cited 2013 Feb 16]. Available from: `http://www.ustream.tv/recorded/15907824`

[17] Winners' Final : RoboCup2011 Soccer Simulation 2D League [Internet]. 2011 Jul 10 [cited 2013 Feb 16]. Available from: `http://www.ustream.tv/recorded/15906095`, 2013-02-16

[18] First Round Group A,B : RoboCup2011 Soccer Simulation 2D League [Internet]. 2011 Jul 7 [cited 2013 Feb 16]. Available from: `http://www.ustream.tv/recorded/15844783`, 2013-02-16

[19] Eighths Finals : RoboCup2011 Soccer Simulation 2D League [Internet]. 2011 Jul 9 [cited 2013 Feb 23]. Available from: `http://www.ustream.tv/recorded/15883860`

[20] Second Round Group E : RoboCup2011 Soccer Simulation 2D League [Internet]. 2011 Jul 8 [cited 2013 Feb 23]. Available from: `http://www.ustream.tv/recorded/15863719`

[21] Pagello E, D'Angelo A, Montesello F, Garelli F, Ferrari C. Cooperative behaviors in multi-robot systems through implicit communication. ScienceDirect - 1999 Oct 31; 29(1):65-77.

[22] Hochstein L, Lerner S, Clark JJ, Cooperstock J. Soccer-Swarm: A Visualization Framework for the Development of Robot Soccer Players [Internet].

[cited 2013 Mar 14]. Available from: http://cim.mcgill.ca/~clark/vmrl/web-content/papers/jjclark_isr_2000.pdf

[23] Faria BM, Reis LP, Lau N, Castillo G. Machine Learning Algorithms applied to the Classification of Robotic Soccer Formations and Opponent Teams [Internet]. 2010 [cited 2013 Mar 14]. Available from: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5518540

[24] Bai A, Zhang H, Lu G, Jiang M, Chen X. WrightEagle 2D Soccer Simulation Team Description 2012 [Internet]. 2012 [cited 2013 Mar 14]. Available from: http://ai.ustc.edu.cn/en/robocup/2D/tdps/WrightEagle2012_2D_Soccer_Simulation_Team_Description_Paper.pdf

[25] Bai A, Lu G, Zhang H, Chen X. WrightEagle 2D Soccer Simulation Team Description 2011 [Internet]. 2011 [cited 2013 Mar 7]. Available from: http://ai.ustc.edu.cn/en/robocup/2D/tdps/WrightEagle2011_2D_Soccer_Simulation_Team_Description_Paper.pdf

[26] Bai A, Wang J, Lu G, et al. WrightEagle 2D Soccer Simulation Team Description 2010 [Internet]. 2010 [cited: 2013 Mar 9]. Available from: http://ai.ustc.edu.cn/en/robocup/2D/tdps/WrightEagle2010_2D_Soccer_Simulation_Team_Description_Paper.pdf

[27] Shi K, Bai A, Tai Y, Chen X. WrightEagle2009 2D Soccer Simulation Team Description Paper [Internet]. 2009 [cited 2013 Mar 14]. Available from: http://ai.ustc.edu.cn/en/robocup/2D/tdps/WrightEagle2009_2D_Soccer_Simulation_Team_Description_Paper.pdf

[28] Shi K, Liu T, Bai A, Wang W, Fan C, Chen X. WrightEagle2008 2D Soccer Simulation Team Description Paper [Internet]. 2008 [cited 2013 Mar 19]. Available from: http://ai.ustc.edu.cn/en/robocup/2D/tdps/WrightEagle2008_2D_Soccer_Simulation_Team_Description_Paper.pdf

[29] Chen X, Changjie F, Feng W, Jiliang W, Jingman C. Wright Eagle 2D Simulation Team [Internet]. 2007 [cited 2013 Mar 19]. Available from: http://ai.ustc.edu.cn/en/robocup/2D/tdps/WrightEagle2007_2D_Soccer_Simulation_Team_Description_Paper.pdf

[30] Markov decision process [Internet]. [updated 2013 Mar 15; cited 2013 Mar 23]. Available from: http://en.wikipedia.org/wiki/Markov_decision_process, 2013-03-09