RoboCup Soccer Simulation
A hand-coded approach to creating autonomous agents

Ludwig Pethrus Engström
ludwigpe@kth.se

Erik Haqvinsson
erikhaq@kth.se

12 April 2013

Abstract

This thesis aims to implement and analyze the success of a number of autonomous soccer teams implemented for RoboCup Soccer Simulation. RoboCup is an initiative with the goal of advancing research in the field of artificial intelligence and robotics, hosting international tournaments each year in both computer simulated and robot soccer.

While many of the competitive teams of RoboCup use advanced machine learning algorithms in their design, the teams presented here exclusively use hand-coded methods, and this thesis aims to explore the success of such teams.

The different strategies used in the teams are presented and compared to each other along with a more precise analysis of the different player skills and their usefulness. Additionally, possible further improvements are recognized, as well as their plausibilities. Finally, the thesis touches on the goal of the RoboCup organization and its role as a gateway for developers entering the world of artificial intelligence.

Sammanfattning

Målet med denna rapport är att implementera och analysera framgången hos ett antal autonoma fotbollslag implementerade för RoboCup Soccer Simulation. RoboCup är ett initiativ vars mål är att driva utvecklingen av artificiell intelligens och läran om robotik framåt. Varje år genomför de internationella turneringar inom datorsimulerad och robotstyrd fotboll.

Även om de flesta tävlingsinriktade lag inom RoboCup idag implementerar avancerade maskininlärningsalgoritmer, använder sig de lag som presenteras här endast av handkodade metoder, och rapporten syftar till att utforska framgången hos sådana lag.

De olika lagens strategier presenteras och jämförs mot varandra tillsammans med en noggrannare analys av de olika spelarnas färdigheter och funktionsduglighet. Dessutom identifieras möjliga framtida förbättringar och deras rimlighet. Slutligen så belyses RoboCup-organisationens mål och dess roll som en inkörsport för utvecklare som strävar efter att komma igång med artificiell intelligens.

Preface

The work in this document presents the thesis part of the Bachelor's Degree Project in Computer Science, First Level, at the School of Computer Science and Communication at KTH Royal Institute of Technology.

The authors are Ludwig Pethrus Engström and Erik Haqvinsson, and the supervisor of the thesis was Petter Ögren, Associate Professor at the Computer Vision and Active Perception lab. Examiner was Mårten Björkman, Associate Professor and Director of Studies in Computer Science at CSC.

Thanks is directed to Sébastien Lentz, for the work in his Master's thesis *Design of intelligent agents for the soccer game*, which has been of much help during the work of this project.

Contents

1 Introduction

1.1 Background

This section will explain what RoboCup is, and briefly describe the Soccer simulator. It also touches on the artificial intelligence aspects of the project.

1.1.1 RoboCup

The RoboCup organization is a scientific initiative aiming to advance research in artificial intelligence and robotics [4]. It first started in 1997 with a soccer simulation league, where soccer teams consisting of players controlled by autonomous computer programs competed each other in a game of simulated soccer. Since then, every year RoboCup has hosted an international tournament, which has evolved to include other leagues beyond simulation, such as physical robot leagues, and has also added alternative domains besides soccer. As such, todays tournaments does not only host competitions with simulated soccer players but also with robots of different kinds, sizes and functions. This was a natural step in the evolution of RoboCup, due to its original mission:

> "By mid-21st century, a team of fully autonomous humanoid robot soccer players shall win the soccer game, comply with the official rule of the FIFA, against the winner of the most recent World Cup." [5]

The RoboCup initiative has provided much research in the field of artificial intelligence and team strategy, and also has led to many surprising technological advances. One such technology is the development of the domain RoboCup Rescue, which promotes research in the field of search and rescue robots.

1.1.2 RoboCup Soccer Simulation

This essay will focus on building a team for the RoboCup 2D Soccer Simulation league, the original soccer simulation league. A game in RoboCup simulation is based on a client/server architecture, where the server provides a virtual soccer field, and the clients control each player's movement independently.

1.1.2.1 Soccer Simulator Server

The server that the clients interact with is called the Soccer Simulator Server. It is the main part of the whole simulator package, as it runs the actual simulation. Clients and server communicate via strings sent through UDP-packets. The server runs in discrete time, with each *time step* being 100 milliseconds. Every time step the server provides each individual client with sensory information about the state of the game, such as position and movement of players and ball. In the original configuration, this information is only partial and will be different for each client, as the server tries to simulate a real-world soccer game, where the players have a limited perceptual information [10] [2]. The clients in turn send commands to the server, such as dash, kick or turn, and the server then updates the environment accordingly.

1.1.2.2 Clients

The client programs that control the players in Soccer Simulation are so called intelligent agents, that is, they are autonomous entities which perceives an environment through sensory input and acts upon this to achieve a goal [13]. Simply put, each client receives information about the current state of the game from the server, and then must form a decision of what command it wants to send based on this input. It must do so within the time limit of the current time step. The commands allowed are in the form of *dash, turn*, and *kick* and only one command is allowed to be sent during each time step. There are also commands which does not directly affect the environment for other players, and consequently can be sent many times during a time step, such as *turn neck, say* and *hear*. Each agent can only control one player and any communication between agents must also go through the server through the *say* and *hear* commands.

1.1.2.3 Monitor

Another part of the simulator package is the monitor. It connects to the server and provides visual view of the soccer field and the players. It is also used to start the actual game and can be used to manually referee the game.

1.1.3 Artificial Intelligence in RoboCup Simulation

Since there is no central client that controls all the agents in the simulation, all agents must individually evaluate the environment and make their decisions. At the same time, the agents must work together within a multi-agent system to achieve a common goal.

The simplest form of an intelligent agent is a so called *simple reflex agent* [13]. It decides what actions to take based on simple, *hand-coded*, if-then conditioning. As there is a very large amount of possible states in the simulation, it can become very tedious trying to manually define enough rules to make the agents perform something complex with a simple reflex agent.

As such, many competitive teams today make use of one or more machine learning algorithms, such as reinforcement learning. Reinforcement learning allows agents to maximize their long term reward by training in the environment, and will theoretically under the right conditions converge with an optimal set of rules. This not only gives the agents a more precise set of rules, but it also allows them to take actions for unknown states, and then learn from the outcome.

Even though using some form of machine learning techniques is often the preferred method when creating high performing AI-systems [13], it has been proven to be a difficult challenge when it comes to RoboCup soccer [11] due to the vast number of states. Most teams therefore use hand-coded algorithms for low- to mid-level tasks, such as passing, getting in position etc. and then apply machine learning methods for subtasks in the simulation, like keeping possession of the ball.

1.2 Problem statement

The main goal of this essay is to implement a number of different simple team strategies for RoboCup 2D Soccer Simulation, focusing on team formation and passing techniques, and compare them to each other. It will explore the problems and qualities with these strategies and aims to give the reader an introduction to the design process of autonomous agents.

Due to the limited time available for this project, no machine learning will be used and the strategies implemented here will only use hand-coded methods. This will be further discussed in the next section.

Many problems arise when trying to create an optimal soccer team. As mentioned earlier trying to make rules account for all possible situations in the game is very time consuming if not impossible. Still, a moderately good working hand-coded team is possible to achieve, and even if it is far from optimal, it displays artificially intelligent agents, working together autonomously. One of the drawbacks with completely hand-coded strategies though, is that it can be fairly easy to analyze it, and make use of its weaknesses. The success of a specific strategy implementation may therefore vary greatly on the opposing team's choice of game strategy, making it difficult

to rate the viability of a strategy.

The work in the essay will analyze some existing strategies and try to improve upon them, as well as defining new ones. The necessities for developing successful strategies will not only rely on designing efficient implementations that allows for a complete execution within a time step, but also taking relevant actions as far as possible.

1.3 Scope/Limitations

As mentioned earlier, the limited amount of time available puts a certain number of constraints upon the scope of this project.

The strategies created here will only make use of hand-coded methods. Using any form of machine learning methods in a way that would produce strategies superior to the hand coded ones presented is simply not plausible as it requires an extended amount of knowledge and time for implementing and training.

Also, to be able to concentrate the efforts towards building strategies, an old Master's thesis [8] project on the subject will be used as a framework for the clients [9]. It provides simplified higher-level methods for the client commands, handling the communication with the server, as well as some mathematics. More about this is explained in the following section.

Moreover, teams presented here will only make use of 5 players instead of a full 11 player team. This choice was made since the extra work it would require to implement a full team was deemed not to give a satisfactory level of rewards. Because of this it was possible to create several different strategies instead of making a full team. Still, this does not mean that the strategies and skills implemented here are not applicable to a larger team, it would simply require an extended amount of work.

Finally, the configuration of the server has been set to deliver *full* state information to the clients, instead of individual *partial* state information which is common practice in the official tournaments. Although this simplifies the work needed to implement a strategy, it does not allow the creation of strategies otherwise impossible should the partial state messages have been used. It simply shortens the amount of time needed for development.

2 Research

2.1 Client

To keep the server compatible with any programming language, The RoboCup Soccer Simulator provides no libraries for creating clients. There are a couple of sample clients implemented in C++ provided in the manual [2], but these are very simple and a bit outdated. Therefore, when creating a client, one has to either do it from scratch or build upon an existing client or external interface. Creating one from scratch is quite a time consuming job, and as such, the clients introduced in this essay are based on a program called Sebbot [9].

Sebbot is written in Java and implements good basic functionality needed to create new RoboCup clients by providing simple methods for starting players and connecting to the server. Moreover, it includes a parser to parse all messages from the server about the current full state of the game and makes this information easily available for the client. The full state information includes information about the positions, speed and direction of all players and the ball. As mentioned before, in standard configuration the server does not send a full state information message to the clients in order to simulate a realistic game where players actually have a limited perception of their environment, but to speed up development, this full state configuration was chosen. Sebbot also includes some mathematics-tools for easy calculations of distances, angles and vectors between players and other mobile objects on the field. Using Sebbot allows for an easy and quick setup and is great for getting started with creating team formation strategies and new passing techniques for a client.

2.2 Team strategies

Finding teams with only hand-coded strategies proved to be a difficult task since nearly all teams in the RoboCup Soccer Simulation League implement some sort of machine learning algorithms. Also, many of the competitive teams do not release their source code.

However, inspiration for strategies worth exploring can be found from various resources. One of these is an open source project called SoccerBots [3]. It is a part of a larger distribution targeted on robotics research, and simulates the dynamics and dimensions of RoboCup small size robot league. Included with it are several hand-coded team strategies. A report [12] evaluating these teams against each other proved to be valuable for this project,

as these strategies face similar challenges as the ones presented in this essay. Additionally, Sebbot also provided some assistance when designing team formations.

Research of existing hand-coded teams created a foundation from which the teams described in this essay could be built on. The most basic form of hand-coded strategy to implement in a RoboCup soccer team is the "go to ball and shoot" strategy [8] [9]. In this strategy the player turns towards the ball and dashes towards it. When it is in the kickable margin, the player tries to kick the ball in the direction of the goal. This is a very primitive strategy [7], and while a team based only on this certainly would not be competitive, studying it gives a start for the essential skill of successfully advancing towards the ball. Other low-level algorithms worth implementing such as checking for opponents closing in on player were gained from this research.

In the more advanced strategies teams implement some sort of team formation. The strategies studied implemented this in a number of different ways. Some used type based positioning, where players are given a type such as forward, midfielder, defender or wing, and calculate their positions with respect to this. This sort of implementation tries to mimic the real world soccer formations for a more realistic decision making depending on which type a specific player is. Other teams calculated where to move the players in respect to where it was currently positioned, without any static types. These approaches showed both strengths and weaknesses, as shown in the next chapter where they will be compared.

Researching passing strategies was considerably harder though. None of the hand-coded strategies analyzed for this project implemented any form of passing, only some expressed a will to implement it in future releases. A number of passing techniques based on machine learning techniques could be observed but unfortunately fell outside the scope of this project. As such, the passing presented later in the essay are built from scratch.

3 Approach

This essay will introduce three different strategies in three different teams. The teams are Team Gorillas, Team Falcons and Team Ants. The first team, Gorillas, was the first one to be developed, and was a joint effort experimenting with the platform. The subsequent teams were made in parallel to prevent the strategies from exploiting known weaknesses of the opponent and to promote individuality.

A team formation is essential for creating good passing techniques since it replicates a more real world like approach to the decision making of a soccer player. A player should make passing decisions based on its position on the field and also the possibility of the ball being intercepted by the other team if passing. The player should also consider if it is safer to drive the ball forward instead of passing to ultimately score.

The strategies implemented extend on basic functionality from *Sebbot* [8] [9] such as ball interception. Sebbot implements both hand-coded and reinforcment learning policies for intercepting the ball. Since this essay aims to create a RoboCup soccer team which is entirely hand-coded, the team strategies use the hand-coded ball interception policy when trying to retrieve the ball.

As this project does not focus on effective goal scoring strategies, the strategy for trying to score goals is a very simple one. It does not always yield good results and is not in anyway optimal. All players uses the same decision making algorithm to choose when to shoot the ball towards the goal. It only waits until it is within a specified distance of the goal and tries to kick the ball with maximum power towards the center of the goal.

For a better understanding of the formations and positioning along the field, the reader should know that the field is represented by a coordinate system with its origin in the center of the field and the x- and y-axes pointing to the right and down respectively.

3.1 Team Gorillas - The offensive team strategy

The first strategy is based on a very aggressive type of gameplay. There are three players that will work together when retrieving the ball in order to quickly gain control of the ball and not leave open paths for the opponents to attack. The other two players will stay close to the goal in order to fend off incoming attacks.

Player types:

- Forward
- Right wing
- Left wing
- Right defender
- Left defender

Specific parameters for this strategy that can be altered:

safeRadius Detemines how the player reacts to opposing players getting close.

safeFieldOfVision The angle that need to be breached if the player should pass.

defensiveThreshold How far away should the ball be until the defenders move towards the center of the goal.

scoreDistance The distance from which the player should attempt to shoot the ball and score.

3.1.1 Formation

The players are initially placed either in the midfield or in the back, close to the goal depending on their player type. All the players have their own optimal position based on which type they are set to. There is no confirmation that the chosen positions are the optimal in any way, but it is an attempt to spread the players out on the field to cover a wider area. The strategy is not based on the chosen optimal positions and could therefore be changed accordingly to suitable values.

The formation is visualized below in figure[1]. During the game the defenders will mainly stay within their assigned positions and only engage with the ball if they are the closest to the ball or if they have stopped a score attempt from the opposing team. The other players will try to aggressively gain control of the ball and drive it towards the opponents goal.

Figure 1: Initial formation for team Gorillas

3.1.2 Offensive players

3.1.2.1 Positioning

The offensive players (Forward, right wing, left wing) make most of their decisions based on if the team is in possession of the ball or not. If not, the offensive players will start pursuing the ball and try to intercept it. When any of the forward, right wing or left wing players is in possession of the ball, the other two will move towards their specified formation positions and stay along the ball's x-coordinate. All the players will drive the ball towards the center of the goal once in possession of the ball.

3.1.2.2 Passing Technique

The offensive players use a passing technique that only allows them to pass the ball to each other. The right and left wing will only be able to bass to the forward player. The forward player can pass to either one of the right or left wing players. First of all the player needs to decide if it should pass. To know if the player should pass the ball it checks first of all if any of the players on the other team is within the safe radius[fig:2]. If a player is within the safe radius the player checks if that player is within the safe field of vision specified[fig:2]. The player does this check so that it does not pass if any opposing player is behind him when he tries to drive the ball towards the other teams goal. When the decision is made that he needs to make a pass to someone else, he calculates who is the "safest" to pass to. Both of these algorithms are shown below.

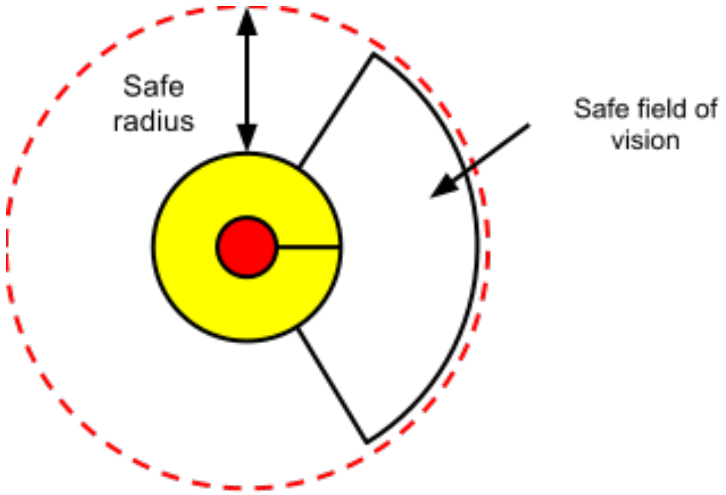

Figure 2: The player model

Algorithm 1 Algorithm for deciding if the player should pass

```
1: List ot ← getOtherTeam()
2: p ← this.Player
3: for all Player op in ot do
4:     if p.distanceTo(op) ≤ safeRadius then
5:         if p.angleFromBody(op) ≤ safeViewDistance then
6:             return true
7:         end if
8:     end if
9: end for
10: return false
```

Algorithm 2 Algorithm for deciding which player to pass

```
1: List ip ← getInterveningPlayers()
2: p ← this.Player
3: rw ← rightWingPlayer
4: lw ← leftWingPlayer
5: passTo ← null
6: float maxTurnDiff ← 0
7: for all Players op in ip do
8:     i ← 2
9:     angRw ← p.angleFromBody(rw)
10:    angLw ← p.angleFromBody(lw)
11:    angOp ← p.angleFromBody(op)
12:    float turndiffRw ← Math.abs(angRw-angOp)
13:    float turndiffLw ← Math.abs(angLw-angOp)
14:    float diff ← Math.max(turnDiffRw, turnDiffLw
15:    if diff ≥ maxTurnDiff then
16:        maxTurnDiff ← diff
17:        if diff == turndiffRw then
18:            passTo ← rw
19:        else
20:            passTo ← lw
21:        end if
22:    end if
23: end for
```

3.1.3 Defensive Players

3.1.3.1 Positioning

The defending players will stay within a specified range of the field. The defenders are allowed to move within their range and will move toward the middle of the goal when the ball has passed the **defensiveThreshold**. They will not leave their positions unless they have the ball and they are driving the ball towards the opposing team's goal. Once another player within the team is closer to the ball, the defenders will return to their assigned positions and resume defending the goal. These two players will be in a very defensive position to try and defend the goal as best they can since there is no goalie.

To determine where on the x- and y-axis the player should position itself they look at the position of the ball. They will always try to stay 2/3 of the distance between the ball and the goal on the x-axis. To determine their position along the y-axis, they check if the ball has passed the **defensiveThreshold** point. If not they go to their specified y-position. If the ball has passed the **defensiveThreshold** point the player linearly interpolates their y-position between its optimal and the center of the goal and moves closer to the center of the goal as the ball gets closer to the goal. The interpolation equation looks like this:

$$y = optimalY \times (1 - (\frac{dtg}{defensiveThreshold}))$$

Where **optimalY** is the set coordinate for the player and **dtg** is the distance between the ball and the goal on the x-axis.

3.1.3.2 Passing Techniques

The defenders can only pass to two players within the team. The right defender will either pass to the forward or right wing player and the left defender to either the forward or left wing player. They both make their decision based on the same algorithm as the forward player does, explained previously.

3.2 Team Falcons - The defensive team strategy

In this strategy all players are assigned a rectangle for where in the field they can move and where to cover. The idea of having rectangles as a bounding box for each player's positioning area makes it easy to see in which rectangle the ball is in and make decisions based on that information. It also presents a somewhat easier way to handle positioning for the players and is a lot more scalable since the algorithm for setting up the players team formation and individual positions along the field is based on each player's given rectangle. The formation of the team could easily be altered by just changing the different rectangles for each player type as desired and setting an optimal position for the player in each rectangle.

Player types:

- Forward
- Right wing
- Left wing
- Right defender
- Left defender

Specific parameters for this strategy that can be altered:

safeRadius Detemines how the player reacts to opposing players getting close.

safeFieldOfVision The angle that need to be breached if the player should pass.

defensiveThreshold How far away should the ball be until the defenders move towards the center of the goal.

scoreDistance The distance from which the player should attempt to shoot the ball and score.

3.2.0.3 Common decision algorithms for all players In this strategy all players use the same algorithms for making decisions on when to pass and to whom. When deciding if the player should pass, this team uses the same algorithm[alg:1] used in team Gorillas.

The passing technique used in this team allows all players to pass to each other. The algorithm to choose which player to pass to first sorts all players in a list from closest to furthest away from the player with the ball. Then it marks each player in the list with the number of players on the other team that is within that player's safe radius. Finally the player passes the ball to the teammate who is closest with the least number of opposing players within its safe radius. The number of opposing players in the safe radius of a teammate takes precedence over distance. The algorithm can be further looked at below.

Algorithm 3 Algorithm for deciding which player to pass

```
1: myTeam ← getMyTeam()
2: otherTeam ← getOtherTeam()
3: sort myTeam from closest to furthest away
4: for all Players tp in myTeam do
5:     Look how many players in otherTeam that is in tp safeRadius
6:     if No one is in the safeRadius of tp then
7:         return tp
8:     end if
9: end for
10: return player in myTeam with the least number of players from
    otherTeam within its safeRadius
```

3.2.1 Formation

This formation is based on the real world soccer formation 4-4-2 [1] but since it only uses five players it looks a little bit different. The defenders are positioned 1/3 of the length from the middle line to the sideline. The right and left wings are positioned 2/3 from the middle line to the sideline optimally. The forward player stays in the center of the field and staying behind the offside line.

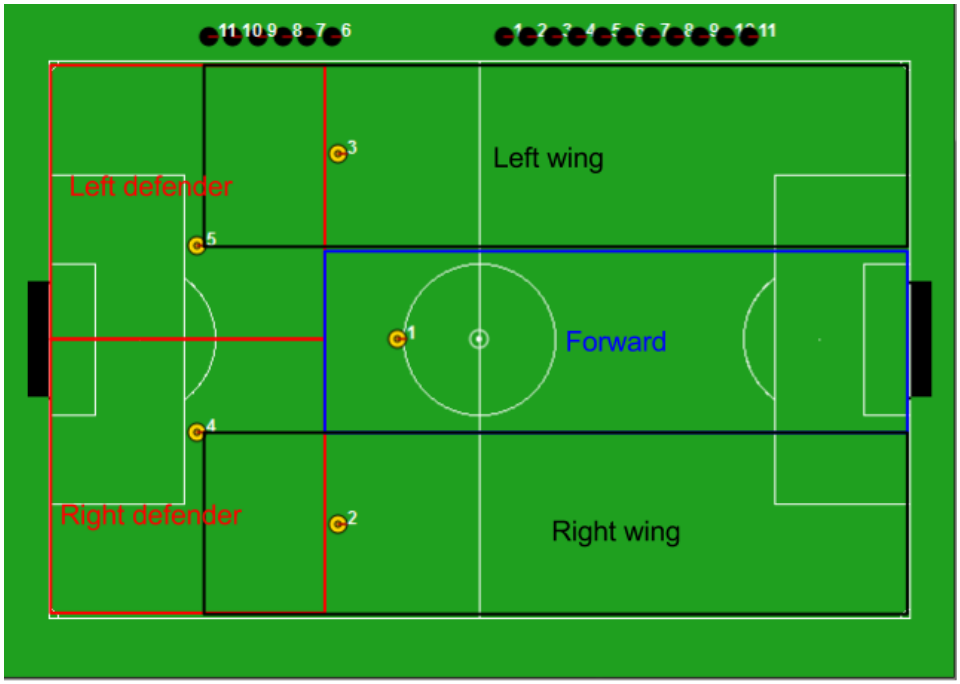

Figure 3: Initial formation for team Falcons and their bounding rectangles

3.2.2 Forward player

The forward player will most of the time be chasing the ball, but if another player is in possession of the ball, the forward player will position itself along the middle line of the field and as far back as the offside line without going over it.

3.2.3 Right and left wing players

Both the right and left wing players make the same decisions and their position on the field are essentially mirrored. When calculating their optimal position on the x-axis of the field the right and left wing players consider the position of the ball and stays 1/3 of the distance between the ball and the goal to stay in a defensive position relative to the position of the ball. If the ball is in their rectangle they follow the balls position on the y-axis otherwise they stay in the center y-position of their rectangle.

3.2.4 Right and left defensive players

Furthest back on the field there are two players playing as right and left defenders. Their job is to defend the goal as best they can. Since there is no goalie these players will take a very defensive position on the field and move closer to their own goal as the ball gets closer to it. Just like the right and left wing they will try to stay at a certain distance from the ball at all times unless they are closest to the ball or the ball is closer than the set **attackThreshold** point. Then they will attack and try to intercept and kick the ball away from the goal.

These defenders determines their position on the x- and y-axis the same way as the defenders in team Gorillas with one extra condition. If the ball is in the other defenders rectangle the player moves to the center of the field and consequently the center of the goal to prepare its defenses.

3.3 Team Ants - The communicative strategy

This strategy utilize formations with an algorithm inspired by one of the strategies in Sebbot. It differs from the previous formation algorithms in that the players do not have any fix types. Instead, the positions of the formations are calculated from the position of the ball, and all clients independently take the formation position closest to them. The player closest to the ball is not part of the formation, but advances with/runs towards the ball, or tries to steal it depending on which team is in possession of the ball.

Another element in this strategy that differ from the other ones, is that the players communicate with each other through the soccer server's built in communication commands when passing the ball. This is to try to improve the chances of receiving the pass.

3.3.1 Formation

The formation initially calculates a rectangle based on the position of the ball. The length of the rectangle is the distance from own goal to the ball, and the width is fixed to be the width of the entire football field, as shown in the figure[fig:4] below.

A defensive and offensive x-coordinate, or line, is then calculated, based on the length of the rectangle. The defensive line is set to 1/4 of the field length, and the offensive one to 3/4. On each line 2 positions are placed on either side of the midline. In the figure displayed these positions are 1/10 from the midline for the defensive positions and 1/4 for the offensive players.

During every time step, each agent except the one closest to the ball independently calculates all these positions, checks which one is the closest, and tries to go to it.

Algorithm 4 Formation positioning algorithm

```
1: myTeam ← getMyTeam()
2: myTeam.remove(playerClosestToBall)
3: positions ← calculatePositions(ball) //calculates coordinates for positions
4: targPos ← null
5: for all Position pos in positions do
6:     targPos ← p
7:     closestToPos ← myTeam.next() //Set to any so not null
8:     for all Player p in myTeam do
9:         if p.distTo(targPos) < closestToPos.distTo(targPos) then
10:            closestToPos ← p
11:        end if
12:    end for
13:    p.remove(closestToPos)
14:    if closestToPos = this.getPlayer() then
15:        break //This player knows its closest position and where to go
16:    end if
17: end for
18: gotoPosition(targPos)
```

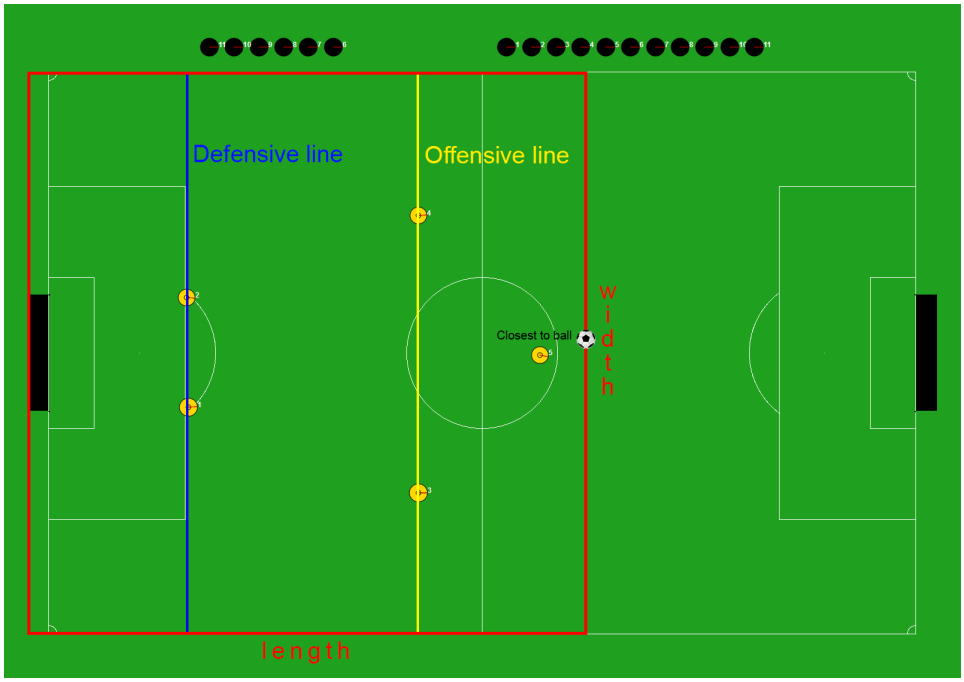

Figure 4: Formation for team Ants

3.3.2 Passing and clearing

When a player is closest to the ball, and in possession of it, he drives it forward. Each time step a check is also made to see if there are any opposing players inside a certain radius of the player. If there is, and the opponent is in front of the player, the decision to pass is made. This is similar to the previous strategies, but instead of looking at angles, only the x-coordinates are observed to determine whether the opponent is a threat or not.

When the pass action is made, it will first check if the ball is within a certain range of the own goal. If it is, a pass is not made and instead the player will only shoot the ball with maximum force away from the goal.

If the ball is far away though, the passing player will look at its teammates, and choose the closest one which does not have opponents in a certain radius of it. If there are no such teammates, a random one is chosen. Another constraint upon choosing a teammate to pass, is that if the player passing is not the one closest to the opponents goal, it will only pass players in front of it. This is to prevent back passes that sometimes resulted in own goals.

3.3.3 Communication and ball reception

The last step of the passing action, is the communication of the pass. On successfully finding a teammate to pass, the passing player will use the *say*-command, to communicate to the specific player passed to that it is about to receive a pass. All players continuously checks for a *hear*-command that tells if a pass is incoming, and if so, will run instantly run towards the ball. Finally, when arriving at the ball, the player will make a weak kick to stop the motion of the ball, increasing the control as a result.

Here follows a simplified version of how players communicate when passing.

Algorithm 5 Algorithm for commnicative passes

```
1: target ← getBestPassMate()
2: passTo(target)
3: say("Passed to " + target.getNumber)
```

Algorithm 6 Algorithm for hearing pass messages

```
1: //Always checked by all clients before other decisions
2: msg ← getHearMsg()
3: if msg ≠ null then
4:     if msg.lastChar() = this.getNumber() then
5:         gotoPosition(ball)
6:     end if
7: end if
8: return
```

4 Results

4.1 Team Gorillas vs. Team Falcons

When the Gorillas faced off against the Falcons some interesting results could be observed about both teams formation and passing techniques. The offensive players on team Gorillas chased the ball most of the time. This occurs since the Falcons often passes the ball backwards telling The Gorillas that they are not in possession of the ball. When the gorillas finally intercept the ball they will often be so close to the Falcons goal that they will attempt to score. This works well for The Gorillas initially when their stamina is high but soon becomes a huge disadvantage. Since The Gorillas most of the time stay together chasing the ball and rarely make a pass play, it is hard to evaluate their passing technique. There is however one positive side of this effect. Sometimes when they are close together and try to score, all three players try to kick the ball. This results in the ball getting a higher velocity and is harder to stop.

The Falcons are often well spread out along the field and when their players have a lot of stamina they quickly intercept the ball and try to drive it forward. However, since the right and left wing players try to stay at some distance from the ball they do not attack the ball when they probably should. Also the Falcon's passing technique does not always yield very good results due to the fact that they can pass any player on the team. When the player has decided to make a pass, the closest player is usually behind it and the forward player is far away standing by the offside line. This often results in the player making a backwards pass which tends to be counter productive. When the Falcons were playing the Gorillas they were at an advantage since they could make vertical passes to players on the right and left wing making the Gorillas running long distances in attempt to intercept the ball. This made the offensive players on Team Gorillas very tired and The Falcons could preserve their energy better.

When the Falcons finally were in shooting range of the goal they always scored a goal. The defenders on team Gorilla has a poor implementation in defensive strategy and react to slow to fend off incoming score attempts

4.2 Team Gorillas vs. Team Ants

This was by far the most uneven matchup. Team Gorillas usually managed to score initially when their stamina was high, but it was quickly depleted trying to chase the ball. The communicative passing game of Team Ants proved to be a winning strategy here, as it displayed a significant improvement in receiving success chance compared to Team Gorillas passing. Vertical passes was especially hard for Team Gorillas to handle as it made them chase the ball with all its attackers, not being able to keep up.

Another edge Team Ants had was its adaptive formation strategy, making better use of the whole team's stamina. Team Gorillas defenders essentially never drove the ball forward, and served little purpose in protecting the goal. The tired attackers of Team Gorillas were also seldom able to catch a ball cleared by a defender from Team Ants, since it could often reach the ball faster in spite of initially being further away.

Finally, the distance from which Team Gorillas attempted to shoot for goal was quite far, sometimes resulting in Team Ants being able to deflect the shot, and gaining control over the ball.

4.3 Team Falcons vs. Team Ants

Not surprisingly, this was the closest matchup. Team Ants was given a slight edge, but both teams showed properties where they excelled over their opponent.

The adaptive positioning of Team Ants still helped to distribute the workload over the team, but a flaw in the strategy could also be observed. When a player from Team Ants was driving the ball forward, and then triggered to make a pass to a player behind it, the passing player quickly moved to the position which the receiving player previously held. This limited the team's possibility to pass a player further up in the field. Team Falcons however, made great use of its forward player for this purpose, delivering it long passes forward in the field, often resulting in a goal.

When looking at passes in general though, the effectiveness of communicative passing from Team Ants was still visible. Many passes in Team Falcons never reached their target due to a combination of a slightly too weak pass and the receiver not being aware of the incoming pass.

Another problem in the way Team Falcons handled passes was observed, when a player running towards the ball reached its destination. Sometimes it would be facing towards its own goal, and despite being able to safely drive the ball forward, opponents coming from behind was caught in its *safe field of vision*, forcing a pass backwards.

4.4 Summary

In summary, it could be observed that the aggressive strategies used by team Gorillas was not successful and their passing technique could not be evaluated much due to the lack of a dynamic gameplay and positioning. The overall winner is Team Ants with their ability to communicate pass plays and effectively receive the ball. Team Falcons became a close runner up and showed some problematic results in its passing technique and uncommunicative formation strategy, that should be improved.

5 Discussion

5.1 Analysis of the results

5.1.1 Team Gorillas

The attempt to create a very aggressive team strategy was not very successful due to the fact that the attackers depleted their stamina quickly and then were chasing the ball but rarely caught up with the other team's players. The defenders on Team Gorilla was also a bottleneck for the team strategy since they rarely succeeded in stopping score attempts. Moreover, the passing technique was never utilized because the players were busy chasing the ball.

5.1.2 Team Falcons

Team Falcons implements a well formed formation strategy but is not aggressive enough and the wrong player often chases the ball. The passing technique used yields some counter productive results since a lot of backwards passes are made, setting up for an attack opportunity for the opposing team. The positioning of the forward player proved to be very successful once a pass was made to that player, setting up for a good chance of scoring.

5.1.3 Team Ants

The communication between the players in Team Ants proved to be a superior skill when making pass plays. This in conjunction with the ability to stop the ball before driving it forward resulted in quick ball control and a fast paced gameplay. The dynamic way of switching between positions on the field also let the players preserve their stamina in a way that was superior to the other two teams.

5.2 Successful skills

From the strategies experimented with in this essay a number of core skills and strategies a team could gain from utilizing can be observed.

5.2.1 The necessity for team formation

First of all, team formation should be a necessity for properly working pass plays. Formations integrating player types seems to be especially profitable for making pass plays, because positions can always be guaranteed to be filled by a player. Although the dynamic positioning used by Team Ants helped keep players from getting fatigued, it sometimes haltered progression in the field due to positions being filled by players that instead would profit from staying where they are.

5.2.2 Minimizing fatigue

Furthermore, the optimal system of keeping fatigue down ought not to be to alternate players to do heavy work, but to *minimize* work. By keeping a type based formation and relying heavily on passing instead of running players become less fatigued. One could still implement some form of role switching, but it should not be the first choice.

5.2.3 Communication

Finally, communication between agents, even a very simple one, was shown to improve the success of a strategy greatly. Even though ball receiving on Team Ants still was far from perfect, it by far outplayed the other teams. Communication should not only be able to enhance many more skills apart from ball receiving, such as smarter passing or on-the-fly strategies, but also simplify the decision making process for other tasks.

5.3 Further improvements

5.3.1 Pass plays

A top priority for improving the strategies should be to increase the number of pass plays. As an example, it could be further developed to let the team make pass plays not only when there is a close threat, but also just to drive the ball forward. This could lead to the players not depleting their stamina as fast and distributing the workload between all players. It would also make it more difficult for the opposing team since they will most likely be confused and change their actions and not fulfill their initial intent to maybe intercept the ball.

Furthermore, the way a passing player chooses its destination to pass to may be improved. For example, predicting a trajectory and insuring that the ball will not be passing through any intercepting players might be a possible decision process.

5.3.2 Ball prediction

Another improvement that would possibly increase performance could be if the players would predict where the ball is heading and how fast. This way a player would know that it should intervene and go towards the ball instead of waiting for a player behind the ball (who is closest) to catch up.

In the future it would be desirable if players could position themselves between the ball and a player on the opposing team who is expected to receive a pass, or the goal. This way, the skill of intercepting a pass or shot would be more likely to succeed.

5.3.3 Setting up for quick counters

As seen with team Falcons, positioning players further up the field can be useful since this allows for a quick counter after fending off an attack. The problem with team Falcons implementation for passing technique was that it did not force the player to try and pass the ball forward. This left the team vulnerable to attacks as stated in the previous section and is not a desirable effect. Instead it would be better if players moved to positions in front of the player currently in possession of the ball, giving more opportunities for smart passes.

5.3.4 Plausibility

All of these improvements mentioned above should not require more skill in the domain of multi agent systems nor coding skills in general. They are deemed relatively easy implementations but time consuming. In future development of a RoboCup Soccer Simulation team these implementations would hopefully yield a better result for a hand-coded team strategy in the RoboCup Soccer Simulation Leagu

6 Conclusions

From the work carried out in this project, it can be concluded that designing completely hand-coded agents for RoboCup Soccer Simulator working fairly well is quite a possible task. In this regard, Soccer Simulator is a fun and relatively easy starting ground for low level artificial intelligence design. It also serves as a logical first step for anyone aiming to later advance to the robotic leagues of RoboCup.

Trying to improve hand-coded strategies to do something more complex though, soon becomes a time consuming "rinse and repeat" process. As the code grows larger and larger, more and more manual testing will be needed, and evaluating its effectiveness can be a difficult task. This is possibly an explanation for the scarcity of hand-coded resources available. It also becomes evident why the competitive teams use machine learning methods, not only to allow agents perform skills otherwise challenging to design, but also to automate the whole process.

If RoboCup ultimately will achieve its goal in 2050 is too early to say today, but if Soccer Simulator is to serve as a gateway for the initiative as a whole, more effort should be dedicated to guiding developers starting out. At the time of writing, even though the latest server version is less than a year old, the manual is 10 years old, and provides only a brief introduction to client design. It even contains outdated, incorrect information about commands, possibly confusing readers. It should be noted though, that RoboCup Soccer Simulator League is a great platform, both in the limitless possibilities for AI design but also in the stimulating design of an international tournament. Sadly though, today it seems limited to schools and researchers that can dedicate an adequate amount of time to it.

References

[1] Formation (association football). http://en.wikipedia.org/wiki/Formation_(association_football)#4.E2.80.934.E2.80.932.

[2] *RoboCup Soccer Server Manual*, Feburary 2003.

[3] Tucker Balch. Teambots website. http://www.teambots.org/, 2000.

[4] The Robocup Federation. About robocup. http://www.robocup.org/about-robocup/, 2013.

[5] The Robocup Federation. The robocup objective. http://www.robocup.org/about-robocup/objective, 2013.

[6] Vic Ciesielsk Jeff Riley. Analysing the difficulty of learning goal-scoring behaviour for robot soccer. In *Robotic Soccer*, number ISBN: 978-3-902613-21-9. Pedro Lima, 2007.

[7] Sebastien Lentz. Go to ball and shoot. https://sebbot.googlecode.com/files/goToBallAndShoot.avi. video.

[8] Sebastien Lentz. Design of intelligent agents for the soccer game. Master's thesis, University of Liège, june 2010.

[9] Sebastien Lentz. Sebbot client. https://code.google.com/p/sebbot/, 2010.

[10] Yaser Al-Onaizan Ali Erdem Gal A. Kaminka Stacy C. Marsella Ion Muslea Milind Tambe *, Jafar Adibi. Building agent teams using an explicit teamwork model and learning. Technical report, Information Sciences Institute and Computer Science Department, University of Southern California, 1998.

[11] Gregory Kuhlmann Peter Stone, Richard S. Sutton. Reinforcement learning for robocup soccer keepaway. Technical report, Department of Computer Sciences, The University of Texas at Austin, Department of Computing Science, University of Alberta, 2010.

[12] Dr.R.Subramanian R. Geetha Ramani and M.Sindurathy. Strategies of teams in soccerbots. In *International Journal of Advanced Robotic Systems*, volume 5. 2008.

[13] Peter Norvig Stuart J. Russell. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3rd edition, 2003.