# Swarm AI: A Solution to Soccer

By: Alex Kutsenok

Advisor: Michael Wollowski

Senior Thesis
Rose-Hulman Institute of Technology
Department of Computer Science and Software Engineering
May 31st, 2004

## Abstract

It is my intent to discuss the effectiveness of using Swarm AI to control agents playing the game of soccer. Since soccer is a complex real-time strategy game, I have chosen to focus on solving 3 of the hardest sub-problems within it: deciding when to pass, moving without the ball on offense, and defensive coverage. In order to adequately test the Swarm architecture, I have designed a centralized rule-based team that was used as a foil in testing the Swarm design. Both teams share the same instructions for handling low-level soccer problems like shooting on goal and intercepting the ball. However, they differ in their ways of handling the 3 sub-problems stated above. Thus, my goal was to measure the effectiveness of a Swarm AI solution when it comes to the most difficult problems of soccer by comparing it to a more traditional approach. In the end, experiments showed that Swarm Intelligence is indeed a good solution to the soccer problems being addressed.

## I Introduction

### Why Soccer?

Soccer is the most popular sport in the world and has received a lot of attention from researchers in the last 10 years. This game is quite complex, as it involves 22 people and a ball moving around a relatively large space at the same time. Therefore, AI researchers view this sport as a problem of deciding how the individual players should behave to help their team win. The number of possible game situations is almost infinite and much greater than that of chess, which is one of the most famous AI problems. The challenge of soccer lies in being able to control a team of players so that they score as often as possible while minimizing the number of times the opposing team scores. Since the number of possible game situation is vast and the actions of the other team cannot be easily predicted, there is no way to use the computer's speed and processing power to simply look through the search space and find a solution. Indeed, there may not be a best way for a team to play, and the challenge of soccer is to find the best approximation of what the team should do.

The majority of soccer AI research is being conducted as part of the RoboCup effort. Proposed in 1992 by Alan Mackworth in "On Seeing Robots," RoboCup is a yearly competition for soccer-playing robots and artificial agents. It contains several levels of competition, one of which deals with purely simulated agents; it is called the Simulator League. Over 20 universities, many from outside of the United States, design the AI for this competition. The long-term goal of RoboCup is to eventually produce robotic teams that can compete with humans by 2050. Therefore, my work and the work of others are driven by the desire to advance the field of AI toward human-like performance. Furthermore, if artificial players can one day compete with humans in soccer, then they will also be able to perform a multitude of other tasks as well as or better than people.

### Swarm Intelligence

Swarm intelligence (SI) is a relatively new AI method. It is typically modeled after Swarm-like insects like bees and ants, whose behavior the Swarm algorithms mimic. Swarm AI involves multiple agents operating in an environment to solve problems through cooperation. It is based on the idea that it is possible to accomplish a task by having multiple agents work on it simultaneously. Instead of tackling a goal directly, one can use several aids that will work separately on smaller goals to reach the larger goal indirectly. Indeed, the individual agents may not be aware of the high-level problem they are trying to solve, and the solution emerges as a result of their joint efforts to accomplish separate tasks.

Another important principle of Swarm Intelligence is that its agents are not given a complete model of the environment. They are given an imperfect perception of the world, so that some are aware of things others are not. This makes it easier (and more efficient) for them to work since they have to handle less data.

The third crucial aspect of Swarm AI is that the agents must have some means of communicating with each other. It is here that the 'Swarm' comes into play since the agents must be able to share knowledge. No single agent can solve a problem on its own, and communication allows the efforts of agents to come together so that they develop an effective solution to a problem the swarm of agents is trying to solve. One example of communication in Swarm Intelligence is the laying of pheromones on paths by ants. Ants cannot talk to each other, but they can choose to lay some quantity of pheromones and also detect how much others have deposited. As a result, ant agents can find good solutions by placing more pheromones on better paths (Bullnheimer 1997).

## My Objective

It is my intent to determine the effectiveness of using Swarm AI to control agents playing the game of soccer. Typically, Swarm Intelligence is used in static environments like the Traveling Salesman Problem (TSP) where optimization is needed. The agents start at some random location and proceed to explore and take note of the world by making trips across it, while remembering the best results found so far. The world does not change as this is happening, giving the insect agents a chance to learn all about it. My work takes Swarm AI in a new direction, since the soccer environment is very dynamic, where the positions of the ball and players change often and quickly. There is no time for agents to build up a lot of knowledge of the situation, as they must react to the situation on the fly to prevent the other team from scoring and to score themselves.

Nevertheless, Swarm AI appears to be a promising solution to soccer, which naturally involves many individuals working together much like ants. The players are focused on doing small things like running towards the ball or becoming open to receive a pass, where no one person is in charge of winning the game for his team. Indeed, soccer is all about breaking a problem up into pieces and having the individual players work on them the way insects work together on problems like foraging for food. During the game, players often need to exchange information, which is something Swarm Intelligence also handles as it allows for communication between the autonomous agents.

To properly test how well Swarm AI works as a strategy for controlling a team of soccer agents I will compare its performance to that of a centralized rule-based architecture. I will try to solve the problems of deciding when to pass, moving without the ball on offense, and defensive coverage. Both architectures use the same methods for dealing with low-level issues like shooting on goal and intercepting the ball, but differ in their strategies for handling the 3 sub-problems stated above. Therefore, my intent is to measure the success of a Swarm AI solution in solving these soccer problems by comparing it to a more familiar approach and seeing which yields a better team of soccer players. The criteria for judging teams of soccer agents involves seeing whether they can win against each other and will be discussed in section IX of this paper.

## Previous Work and How It Relates To My Goals

As mentioned earlier, Swarm Intelligence has been applied to classic computer science problems like Graph Coloring, the TSP, and also the Vehicle Routing Problem (VRP), which is a generalization of the TSP. These efforts began in the late 90s and have had promising results. For example, the most successful Swarm algorithms for the VRP are better than neural network solutions, though they are not as good as Tabu Search and Simulated Annealing which currently yield the best results in this area of research (Bullnheimer 1997). Swarm AI has also been applied to real-world path-finding problems like network routing and crowd control.

Work that is more closely related to what I am doing deals with chess, as detailed in “When Ants Play Chess” by Alexis Drogoul. Here, SI is used to break up the game of chess into parts handled by the individual pieces that analyze the board from their own points of view. This makes it easier to make decisions since the problem each piece is solving is small, dealing with what squares it threatens and what pieces threaten it. Without discussing the details of this strategy, it is apparent that it is much simpler than trying to analyze the game while taking many pieces into account at once. This chess AI competes at the level of a competent human opponent, which is surprising given the straightforward nature of the rules governing the behavior of each chess agent (Drogoul 1993).

What is most interesting about Drogoul’s work is that he is using Swarm strategy in a game where there is competition and opponents are actively working against each other, an environment that is quite different from the ones of the classic computer science problems mentioned earlier. While soccer is different in many ways from chess, it shares the idea of teams of agents competing against each other in some space. Like Drogoul’s work and the strategies used by other Swarm researchers, my approach will focus on individual agents making decisions, rather than having a single entity look at the field and tell each player what to do.

In RoboCup Simulation league, the most successful AI solutions deal with planning and explicit models of teamwork. Systems like University of Southern California’s ISIS are very complex and require teams of students to create and improve them in preparation for the yearly competition. Furthermore, architectures like this one are built on top of other architectures, STEAM in this case, which further increases the complexity of the overall design (Tambe 1998). In RoboCup Simulation league research and independent research, there have been no serious attempts to use Swarm AI to control soccer agents.

I believe that a SI soccer system will have the advantage of simplicity and performance efficiency over the complicated architectures currently being researched. Consequently, I intend to show several things with my research. First of all, I will show that Swarm AI can be used in a dynamic real-time environment. Furthermore, I will show that a simple and efficient swarm approach can be as successful or even better than a centralized approach that is smaller in scale but similar in principle to current RoboCup architectures. Like the RoboCup approaches, the centralized strategy will use a “coach agent” that will re-analyze the situation every turn it can and provide the soccer players with directions, such that they are entirely dependant on his advice and a set of pre-programmed rules.

# II The Soccer Model

### Brief Overview of Traditional Soccer Rules

Soccer is a sport that is played on a large field divided into two halves. Goals are located on opposite sides of the field, and each team tries to kick the ball through the other team’s goal while defending it’s own goal.

In official games, the teams have 11 players per side, one of whom is a goalie whose sole responsibility is to guard the goal. The goalie may touch the ball with any part of his body, while the other players cannot touch it with their hands and arms. Players typically hit the ball with their feet, knees, and head. Frequently, soccer is played informally where there are fewer players on each team and there may not be a goalie. Official games run for 90 minutes but informal games run for as long as the players feel like playing. At the end of a game, the team that has scored the most goals wins, and the game is declared a tie if both teams scored an equal number of goals.

During the game, the players move the ball by dribbling it with their feet to keep it with them or by passing it to their teammates. When a player is close to the enemy goal, he may try to kick or head the ball through that goal. At any one time, only one of the teams has possession of the ball. This team tries to move the ball toward the enemy goal and then to shoot on goal. While this is happening, the other team plays defense by trying to prevent the first team from moving the ball towards its goal. In addition, the team without the ball tries to intercept it so that it can take possession. If a player from one team lets the ball go out of bounds, the other team is given possession of the ball.

On each team, players that run around near the enemy's goal are designated as forwards, and they are primarily responsible for scoring. Players that are usually located close to their own goal are designated as defenders, and they are primarily responsible for staying with enemy forwards and preventing them from scoring. These roles are not enforced by the rules; they are more like guidelines that players adhere to most of the time for sake of organization. Player roles are a way of distributing responsibility, and they may change in the course of the game as teams and players adjust to one another.

Formal soccer games enforce the off-sides rule, which states that a player with the ball cannot make a pass to a teammate if that teammate is horizontally closer to the enemy goal than any opposing player. This prevents forwards from always staying by the enemy goal behind the other team's defenders. If such a pass is made, the game stops and the ball is given to the other team. Soccer also does not allow players to kick or push other players in order to secure the ball or otherwise affect the game and violations may result in game stoppage, loss of possession, and other penalties.

## My Model of Soccer

For this project I created a model of soccer that makes it relatively easy to create player agents that can perform basic tasks like running and kicking the ball. I avoid dealing with realistic problems like vision and complex physics by keeping my model as simple as possible. By abstracting the game in the various ways, I keep the focus on artificial intelligence problems without worrying about other hard engineering problems. Furthermore, many of the decisions that were made in coming up with this model had been reached through trial and error and may not have obvious reasons to explain them.

My simulation supports two teams of 5 players per side playing soccer on a field of roughly standard proportions. I elected to have 5 players per side since using fewer players would make formations less meaningful, as all players would have to move all over the field. On the other hand, having more than 5 players makes it more difficult to analyze results, though it might be a good idea to simulate games with more players in the future because that would be more realistic. When the ball passes through a goal belonging to one side, the other side earns a point because a goal was scored, just like in traditional soccer. All players have identical properties, hence there are no goalies with special privileges or restrictions on where a player can or cannot go. In real soccer, some players may be faster or more skillful than others but for our purposes we are only concerned with the intellectual ability of a player, so giving certain players physical advantages would unnecessarily complicate the model.

The game starts when both teams are on their respective sides of the field and the ball is randomly placed somewhere close to the center of the field. When a goal is scored, the team that scored moves back to their side of the field while the other team is given possession of the ball. Finally, when the ball goes out of bounds, the team that touched it last is unable to move for a short amount of time. This gives the other team a good opportunity to take possession of the ball which is stopped at the spot where it left field of play, much like real soccer where the other team is given the ball at the place where it went out of bounds. My simulation does not model the off sides rule, which would be too severe on offensive players given the fact that they can't kick the ball over their opponents, as will be described in more detail below.

### The Physics

The physics of the simulation and the game of soccer have been simplified to make it easier for AI agents to make decisions while keeping the model sufficiently powerful to represent soccer. First, the field and players exist in two dimensions, so they cannot hit the ball with their head or kick the ball over each other. Next, the players do not face in a certain direction, and they never have to turn themselves. Also, they can change their velocities instantly, so there is no acceleration. Another important simplification is that the ball and the players always travel at the same constant speed, though the ball's speed is faster.

Not working with acceleration and using constant speeds makes it algorithmically simple and computationally quick to accurately calculate the time for one object to intercept another, which is crucial since the players use interception calculations quite frequently. Working in three dimensions with realistic physics would require much more sophisticated algorithms and a great deal more computational time to obtain adequate performance for simple tasks like seeing and running toward the ball. To prevent players from taking advantage of the fact that the ball travels at a constant speed by kicking it unrealistically far, the ball also has a maximum travel time. This means that when a player kicks the ball as a pass or shot on goal, the ball always travels the maximum travel time and then stops, assuming no player had intercepted it as it traveled and the ball did not go out of bounds.

A player is represented by a circle, which is that player's area of control on the field. If the ball enters a player's area of control, that player takes possession of the ball. When a player possesses the ball, the ball becomes "a part of the player" in that it takes the players location and velocity as its own as long as that player possesses the ball. When the distance between a player who possesses the ball and an opposing player is less than a certain "Collision Distance" such that their areas of control intersect, the player with the ball loses it and the opposing player takes control of the ball. All players, be they on the same team or not, can have their areas of control intersect and occupy the same space on the field; they do not push off each other as the areas of control are not physical objects. The players and the ball cannot leave the game field and are prevented from doing so if they try by the simulation's collision detection. Furthermore, the collision detection checks every turn to see which player has possession of the ball. When the ball is about to leave the field through either of the two goals, a goal is scored.

# III The Soccer Simulation

## Why A New Simulation

My experiment relies heavily on a soccer simulation that is used to test the AI of teams of players. It was necessary to create a brand new simulation for this project to fit the model of soccer described above. Since I am working with an original model of soccer, I could not make use of an existing simulation. As I have decided to focus on a certain few high-level problems within soccer, I am not concerned with low-level issues like choosing the best facing directions for players and dealing with realistic ball and player acceleration. Therefore, the simulation abstracts these elements of soccer according to the model described above so that the AI does not have to deal with them.

In addition, the number of simulations openly available in the world of academia is quite low, and I found less than 5 open-source programs. While some of them are of high quality (such as the RoboCup Soccer Server), they did not fit the model requirements described above, as I've found them to be much too complicated for my purpose because of their attention to realistic physics. For example, RoboCup Soccer Server models the game environment in 3 dimensions, allows the ball to accelerate, and permits players to turn their heads to look at their surroundings. This means that the AI engineer needs to program his AI to handle all of these tasks. Since I am not concerned with creating a comprehensive AI that can handle all aspects of soccer, I decided it would be wiser not to handle these issues.

Thus, while the RoboCup Soccer Server models the game of soccer quite realistically, the AI engineer is forced to deal with a tremendous of amount of low-level detail in order to produce a viable team. Also, failure to adequately handle those issues may undermine high-level algorithms. For instance, mistakes in calculating how much of the field a player sees or how long it will be until the ball stops rolling may prevent players from correctly executing high-level maneuvers like making runs to get open for the ball. Of course, ensuring that these low-level routines work is feasible and has been done (Stone 2000), but it is beyond the scope of this project and would detract from work on the AI aspects I am truly interested in. Consequently, I elected to work with a simple model of soccer and wrote my own simulation for it.

## General Description

A simulation run is broken up into discrete intervals (turns) that last several milliseconds; the exact number depends on the specified game speed but has no effect on the game play. Each turn, the players and the ball move around the field according to their velocities. A simulation run lasts for some large amount of turns specified by the user, typically a million turns. Screen pixels are used as units of distance, where the field is 598 pixels in horizontal length and 336 pixels in vertical length. A player's area of control circle has a radius of 6 pixels.

The players perceive the world by observing the positions of the goals, and the positions and velocities of the ball and all players. They obtain a perfect knowledge of the situation whenever they get a chance to perceive it. The players can interact with their environment (the ball and other players) in only 3 ways:

1) setting their own movement velocities, which means they decide the direction and whether the velocity is non-zero or not.
2) kicking the ball in a certain direction if they have possession of it
3) taking away the ball from enemies, which happens automatically, by running into them

To simulate the fact that soccer is a real-time game (it's not turn-based), the players are not allowed to think every turn. Instead, they have the chance to observe the situation and act upon it only once every 10 turns (the turn they are allowed to think and act is called their "shift turn"). For the other 9 turns, the players continue to move according to the velocities they set during their shift turn. The player with the ball can kick it in some direction only during their shift turn. The teams' shift turns alternate every 5 turns; the first team's players go, and 5 turns later the second team's players go, followed by the first team again 5 turns later, and so on enforced by the simulation. Experiments have shown that this system of using 1 shift turn every 10 turns is effective at simulating life-like soccer behavior in artificial agents.

Every turn, my simulation iterates through all of the players and the ball to adjust their positions based on their velocities for that turn. It also performs collision detection to determine who has the ball and prevent the players and the ball from moving off the field. The simulation recognizes when a goal has been scored or the ball goes out of bounds and adjusts the situation according to rules of the soccer model being used.

### Delays

In order to better simulate the game of soccer, the concept of being delayed has been added. In addition to out of bounds situations, as discussed above, there are other times when the simulation prevents a player from being able to act. Whenever a player kicks the ball, he cannot move, intercept a ball, or tackle an opponent for several turns- it is stunned. This simulates being out of position, and prevents unrealistic behavior such as kicking the ball into a defender and immediately taking it back. In addition, a tackled player is stunned for several turns to prevent him from immediately re-taking the ball from the opposing player that is very close to him. Finally, when a player narrowly intercepts a pass, the enemy players around him are briefly stunned so that the player has a chance to move away or pass the ball. This simulates the fact that real players can use their bodies to guard the ball for a small period of time, and rewards players for being the first to the ball by not allowing the players that were late to immediately steal the ball.

# IV The Foundation Architecture

### Introduction

Both the Swarm AI (SAI) and the centralized rule-based AI (CRAI) share the same instructions except for those concerning defensive assignments, offensive movement without the ball, and pass decisions. Therefore, both AIs handle the less interesting problems of movement with the ball, shooting, intercepting the ball, making passes, and other tasks in the same way. These basic instructions that the two AIs share are the foundations of their respective architectures that allow the two different AIs to adequately play soccer. This means that the CRAI and SAI are built on top of these existing rules, with different ways of handling the 3 soccer sub-problems this thesis addresses. These foundation rules interface with the simulation so that the more high-level algorithms don't have to work with it directly. Thus, this basic architecture is very important to the way both AIs behave and will be described in detail in the following paragraphs.

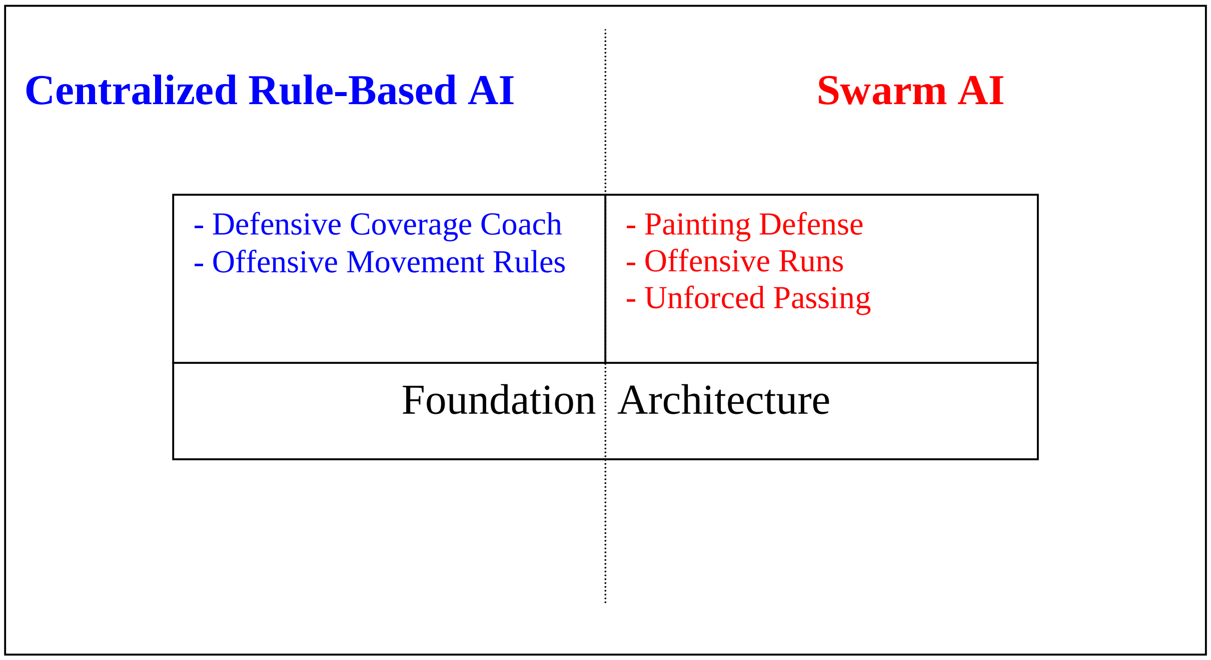


Figure 1: Architecture Diagram

**Perception and Short-Term Goal-Setting**

During a “shift turn” which happens once every 10 turns, the players of a particular team perceive, think, and act on their environment. Since the AI is given perfect knowledge of the locations and velocities of all game objects, the perception element of the AI is trivial. Each player analyzes the situation by matching the current world situation to one of the situations it has been programmed to recognize, making a generalization. For example, a player may notice that its teammate has the ball, and therefore, match this situation to the “my team has possession of the ball” situation it knows about. Next, each player sets short-term goals for itself based upon the current situation. To be more precise, a short-term goal is just the behavior a player will have for the next 10 turns until a new goal is selected during the upcoming shift turn. These goals include: pass the ball, shoot the ball, escape pursuit, intercept ball, and others. In addition, some goals are specific to CRAI or SAI, and the FA allows for them to be selected in addition to the few basic goals it provides. Figure 2 at the end of this section outlines how players select these goals.

After all the players are done setting their short-term goals, the players will act according to their goal. For example, if a player’s goal is to intercept the ball, the player will set its velocity in the appropriate direction. As a result, during the next 9 turns, that player will continue moving in that direction. The following sections describe how individual aspects of soccer are handled.

**Passing**

The Foundation Architecture (FA) addresses the problem of deciding when to pass by forcing players to pass only when they will soon be prevented from being able to dribble towards the enemy goal. In other words, the FA checks to see if there are enemies ahead or behind a ball carrier and whether those enemies will be able to intercept the ball carrier. If the situation doesn't look promising, a ball carrier will try to improve the situation by either passing the ball, running away from his pursuers, or kicking the ball towards the enemy goal to get rid of it. The best option is usually to make a pass, so a ball carrier will always consider it first and run the Directional Passing Algorithm to determine the feasibility of making a pass. Before going further, it is important to note that while this scheme is sufficient for the CRAI, the SAI has it's own take on passing which works in addition to this strategy.

The Directional Passing Algorithm (DPA) works by considering passes in the 16 directions around the ball carrier (they are 360/16 = 22.5 degrees apart). A larger number of directions could have been used, but 16 are sufficient in providing a passer with a large number of options. This is accomplished by iterating from angles 0 to 2*Pi in steps of Pi/8. For each of the 16 angles, the algorithm calculates who would get the ball if the ball were kicked in that manner. If a friendly player would intercept the ball, the algorithm computes approximately how much time that player would have until an enemy arrives at that position. These times are used in a heuristic that gives more weight to forward passes (it's usually better to advance the ball with a pass, rather than kick it towards one's own goal). Finally, the passing direction with the highest heuristic value (the heuristic favors passes that will allow the receiver more time with the ball and move the ball towards the enemy goal) is then returned and the pass is made in that direction. If a successful pass cannot be made, DPA returns False and the ball carrier will consider performing evasive maneuvers or kicking the ball towards the enemy goal.

The Directional Passing Algorithm is fairly successful since it doesn't limit passes to person-to-person passes. In fact, this algorithm doesn't consider the location of the pass recipient at the time the pass is made, but instead analyzes where the recipient of the pass will be in the future. This "passing into space" rather then passing directly to other players is very important in real soccer where players almost always make passes by estimating where their targets will be some time later.

Nevertheless, the Directional Passing Algorithm is not perfect. Often a pass is intercepted because the target did not respond to it as quickly as an opposing player. This happens because the players are allowed to change their velocities every 10 turns, and not immediately after a pass is made. The ball carrier assumes that the friendly pass target will immediately chase the ball, which might not happen. Decreasing the number of turns between shift turns can reduce these errors. In addition, sometimes a pass is made that doesn't leave its recipient with much time once it intercepts the ball. This leads to a tackle where an opposing player takes the ball away soon after it is received in by the pass target.

## Shooting on Goal

When a player with the ball is inside the penalty box of the opposite team, during the next shift turn it realizes this and sets its short-term goal to Shoot the Ball. A shooting player considers 3 possible shots: one to the center of the goal and two to the far corners. Taking into account defenders between the player and the goal, the shooting player chooses a shot that will not be intercepted. If the ball carrier decides that all 3 may be intercepted, a random shot is selected from the 3 choices.

## Interception

For a soccer AI, it is often necessary to determine if a player is able to intercept the ball or another player that is traveling with a certain velocity. For this purpose, I am using an algorithm that works by predicting where the target to be intercepted will be in the future. It calculates where the target will be x turns from now and determines if a player can reach that location in x turns. The algorithm starts at x=1 and loops while incrementing x by 1 until it decides an interception can be made. If the target is predicted to go out of bounds, then this algorithm assumes the target will stop at the field boundary and continues running until a player can reach that location in some x turns.

This algorithm always returns the correct result as long as the target maintains its current velocity. However, since the players are not allowed to think every single turn, an interesting situation can arise. One player can make a pass and use the interception algorithm to confirm that his teammate will be able to intercept it. However, that teammate may not have a chance to change his velocity until the next shift turn, which can sometimes result in him being unable to intercept that pass. As a result, the FA allows for passes that can be intercepted by one’s opponents, since it imitates the fact that there is a delay in the time between a made pass and players reacting to it.

## Setting Goals

Figure 2 below outlines what happens each shift turn in terms of goal setting and other tasks performed by a single soccer agent. This outline includes information about CRAI and SAI, which will be explained later and should be passed over by the reader at this point. This outline shows how a player decides what to do during the shift turn, depending on whether he has the ball or not and other factors. The two parts of the outlined behavior are performed in sequence each shift turn, starting with part I. It is important to note that the outline is very complex and will become clearer as one reads the rest of the paper.

**Goal Setting Rules**

Key
Normal text = FA components
*Italics* = Swarm components
**Bold** = CRAI components
Underlined = Goals

```
                              Part I:
Coach makes defensive assignments
Swarm players individually run IPDA
Swarm players individually run OSPA
                              Part II
if player has ball
    if is in enemy penalty area => Shoot on goal
    else if need to get rid of the ball
        if can make a directional pass => Make a directional pass
        else if enemy is really close (desperation) => Shoot on goal to get rid of the ball
        else => Escape pursuit which mean running in direction opposite of closest enemy
    else if  Paint Passing Algorithm returns True => Make a directional pass
    else if are not in the central vertical third => Move toward enemy goal and the central vertical third
    else => Move forward toward enemy goal (horizontally)
else  (player doesn't have ball)
    if ball is loose and you are the most likely on your team to get it=> Intercept the ball
    else if one of the following 4 things is true:
         1) Your team has ball
         2) Your team is about to have ball because your teammate will intercept it
         3) You are a Defender and the target you should be guarding is on his side of the field
         4) You are a Defender and the target you should be guarding is on his side of the field and this
         target is a player, not a zone.
      =>  if you are a Top Defender => Defensive Rubber Band movement toward top vertical third
        else if you are a Bottom Defender => Defensive Rubber Band movement toward bottom vertical third
        else if you are a Center Defender => Defensive Rubber Band movement toward center vertical third
        else if you are a Bottom Forward => Forward Rubber Band movement toward bottom vertical third
        else if you are a Center Forward =>  Forward Rubber Band movement toward center vertical third
        else if you are a Top Forward => Forward Rubber Band movement toward top vertical third
        if you are a Defender => Make Defender Runs
        else if you are a Forward => Make Forward Runs
    else => Cover your target
```

Figure 2: Goal Setting Rules for all architectures

## V  Centralized Rule-Based AI

### Introduction

Centralized Rule-Based AI approaches the 3 sub problems of soccer being addressed in this paper from the perspective of a single independent agent. This agent is responsible for solving the whole problem, be it deciding when to pass, whom everyone should guard, or where to move when one's team has the ball. In each case, the single agent does not receive any advice from other agents before making a decision. A series of "makes sense for most situations" rules that have been inspired by real soccer strategies have been implemented to help guide each agent. As their description implies, these rules are not always correct, and may force an agent to make the same mistake every time the same situation arises. Nevertheless, a team of CRAI agents plays soccer much like a team of real people, and therefore, was a good foil in judging the quality of SAI. The following sections address how CRAI handles the 3 sub problems of soccer being discussed.

## Defensive Coverage

The problem of defensive coverage deals with deciding which players on the team without the ball (the defensive team) should guard which players on the team with the ball (the offensive team). It is a good idea to have defenders in the vicinity of every offensive team's player, especially the ball carrier. As discussed in the Introduction of this paper, the objective of the defensive team is to stop the offensive team from advancing the ball towards the defensive team's goal and to take possession of the ball away from the offensive team.

The CRAI handles this problem by using an additional agent, the Coach, who thinks about the situation and tells the player agents on the field what to do every shift turn. Specifically, the Coach has a chance to perceive, think, and act before the players on his team during the same shift turn. The Coach wants to make sure the most dangerous enemy players are covered by the closest friendly players. The Coach runs the Coach Defense Algorithm (CDA) to tell friendly players who they should guard (stay close to). The following is the step by step outline of how the CDA algorithm works; an example of its use is demonstrated afterward.

1) Assign the closest friendly player to guard the enemy ball carrier if the friendly player is also horizontally between the enemy ball carrier and the friendly goal (the friendly player is between the enemy and the goal).
2) If all friendly players are behind the enemy ball carrier, have the closest friendly player guard him
3) Iterate through all remaining enemy players, starting with the one that's closest to the friendly goal and ending with the one that is farthest
    3.1) For each enemy player, assign the closest friendly player who is not guarding anyone to guard that enemy player.

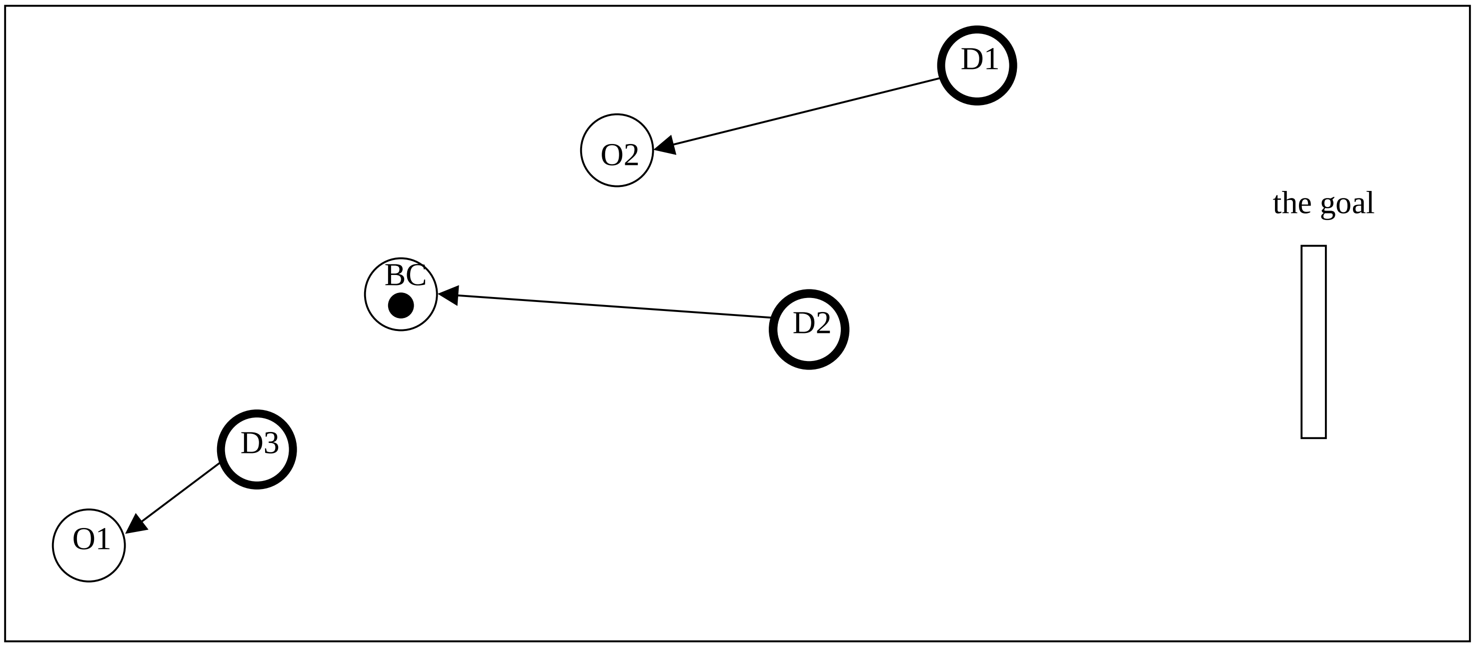


Figure 3: CDA Example

The above figure shows a simple scenario with 3 offensive players (O1, O2, and BC the ball carrier) who are represented by thin circles and 3 defensive players (D1, D2, and D3) who are represented by thick circles. In this diagram and others the players on the team that has possession of the ball (the offensive team) will always be represented by thin circles and the other team's players (the players on the defensive team) will be shown by thick circles. While the simulation supports 5 players per team, here we are only dealing with 3 to simplify things. The Coach runs the CDA algorithm to determine whom the defensive players should guard. First, D2 is assigned to cover the ball carrier since D2 is the closest defensive player to BC who is also between BC and the goal. Next, O2 is the closest enemy to the goal, so it is assigned the closest defender who is not covering anyone so far, D1. Finally, O1 is the last remaining offensive player in the scenario and D3 is the closest player to it. Note that although D3 is the closest player to the ball carrier, it is not between BC and the goal, so D2 was assigned the task of guarding BC.

One important fact to keep in mind is that the CRAI players move to satisfy the Coach's defensive coverage assignments only if they decide that their goal is to guard an enemy player during that shift turn. For example, the players will ignore the Coach's directions if their team has possession of the ball. Also, the player agents may not all have the Cover your target goal, so during one shift turn some may decide to guard enemies while others don't. These decisions are handled by the Foundation Architecture's rules shown in Figure 2. Thus, the Coach only suggests whom a player should guard in case the player chooses to guard someone during that shift turn. Every shift turn the Coach re-analyzes the situation and alters the defensive coverage scheme as the situation changes. This is a highly centralized approach to the defensive coverage problem, given that one Coach agent decides what all the player agents should do.

## Offensive Player Movement Without the Ball

Moving without the ball is an important issue for players whose team currently has possession of the ball. They must be open for a pass from the ball carrier, move towards the enemy goal, and make sure that the enemy team cannot easily score if it suddenly takes possession of the ball. The CRAI handles this task by assigning roles to each of the player agents on the field. These roles provide simple rules for agents to follow, telling them where to go if their team has possession of the ball. The two basic role types are Forward and Defender. Both the Defender and Forward roles have three variants each. These are the Top, Center, or Bottom variants, which refer to the vertical third (top, center, or bottom) of the field that player will try to stay on. Besides trying to stay on the correct vertical third of the field, the Defenders and Forwards have rules that tell them where to move horizontally, that is in parallel to the longer border of the field. These rules are outlined in Figure 2 and are described in more detail below.

A Forward player moves horizontally toward the enemy goal when his team has the ball. He moves forward in this fashion until he is a certain distance away from the ball carrier. At that point, the Forward player will move horizontally backward toward his own goal and the ball. This is called "Forward Rubber Band" movement, and it ensures that the Forward is not too far away from the ball carrier to receive a pass while still being ahead enough to receive a pass that would advance the ball towards the enemy goal. When a Forward player has the ball or the enemy team has it, the fact that the player is a Forward has no effect on his actions.

A Defender player's rules are similar to the Forward's. A Defender player moves horizontally back toward the friendly goal when his team has the ball. It is a good idea for Defenders to drop back when their team has the ball since it allows them to become open for a pass and also to be in a good position to protect the goal in case the other team steals the ball. Furthermore, a Defender moves backward until he is a certain distance away from the ball carrier. At that point, the Defender player will move horizontally forward toward the ball and enemy goal. This “Defensive Rubber Band” movement ensures that the Defender is not too far behind the ball carrier to be open for a pass.

Players are, thus, assigned a forward/backward position on the field, as well as a vertical top/center/bottom position. Therefore, the players can be spread out over the field or bunched up in the same area, depending on what role assignments are given to them. There are only five players on the field, but there are a total of 2*3 = 6 possible roles for them. While it is possible to assign the same role to two different players, this is not practical since it is a good idea in soccer for players to be spread out over the field. I have experimented with two basic formations (5-player role assignments) for the CRAI team. Formation1 has two Forwards (Top and Bottom) and three Defenders (Top, Center, and Bottom), while Formation2 has three Forwards (Top, Center, and Bottom) and two Defenders (Top and Bottom).

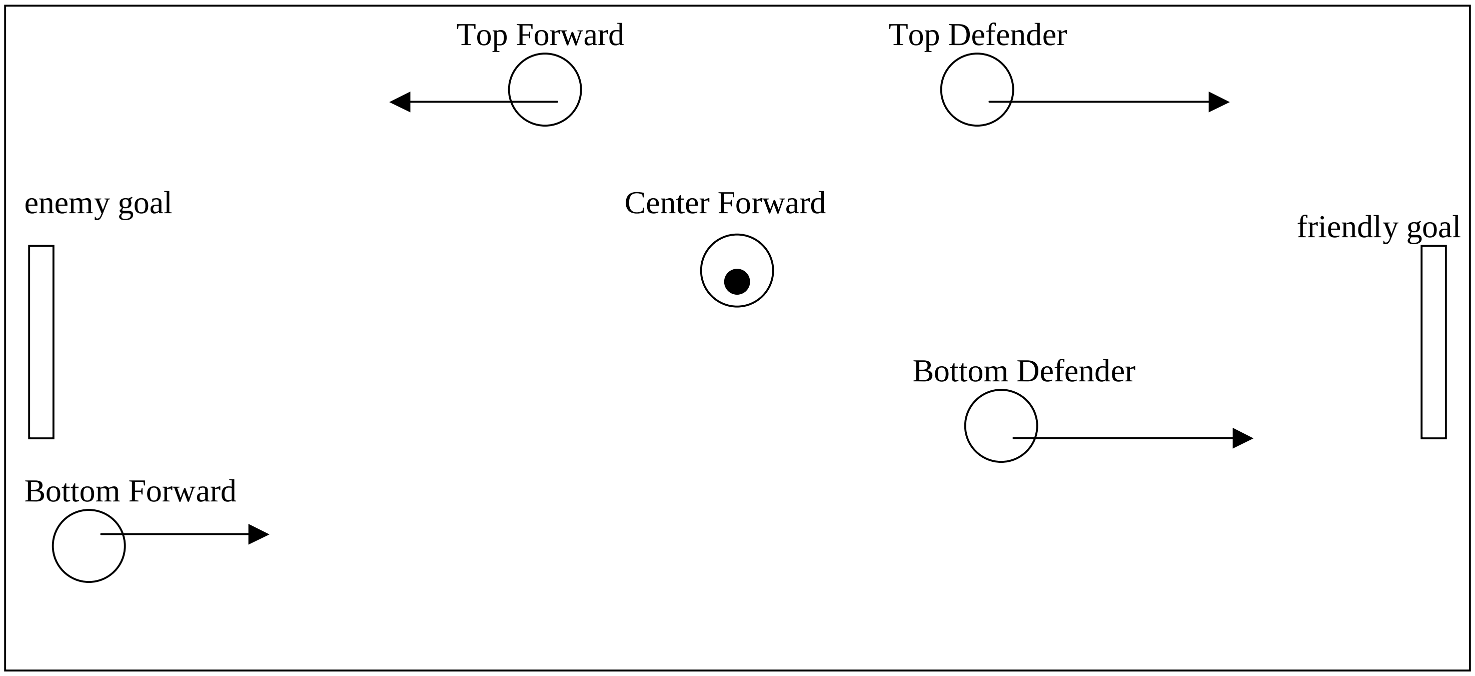


Figure 4: Offensive Movement Diagram using Formation2
(other team’s players not shown)

Figure 4 shows a situation where Center Forward has the ball. The two Defenders will move backward horizontally while the Top Forward will move forward. The Bottom Forward will move back because it is too far ahead of the ball. It is important to note that since the Center Forward has the ball, the offensive movement without the ball rules being described in this section do not apply to him. Formation2  is used here, since there are three Forwards and two Defenders.

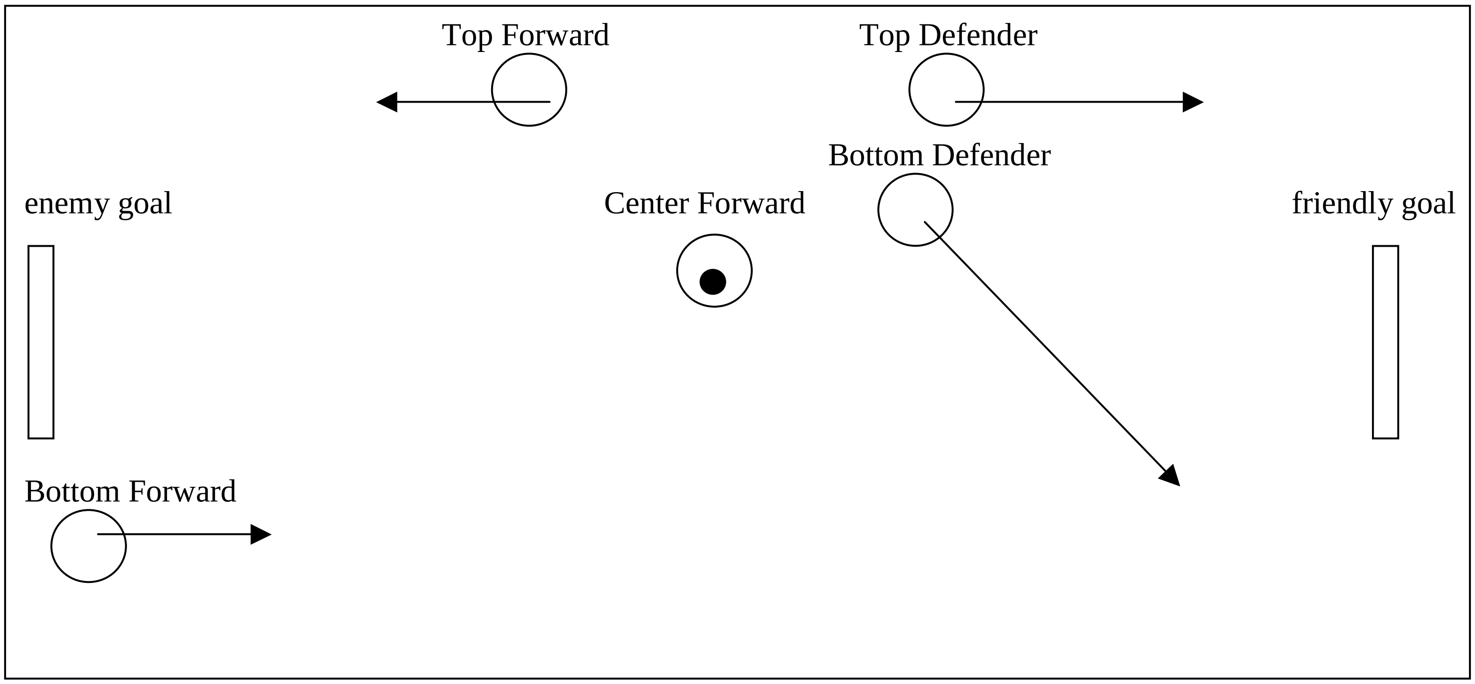


Figure 5: Another Offensive Movement Diagram using Formation2
(other team's players not shown)

Figure 5 shows a similar situation, except that the Bottom Defender is on the wrong vertical third of the field because he should be at the bottom. As a result, he will move both backwards toward his goal and downward.

A Defender's rules are somewhat more complex than the Forward's because it is useful in one situation for Defenders to act as if their team had the ball even when it does not. Specifically, when the enemy player a Defender is assigned to cover by the Coach is on the enemy side of the field, the Defender moves in the defensive rubber band fashion instead of guarding the ball. The following example demonstrates this behavior, which is also outlined in Figure 2.

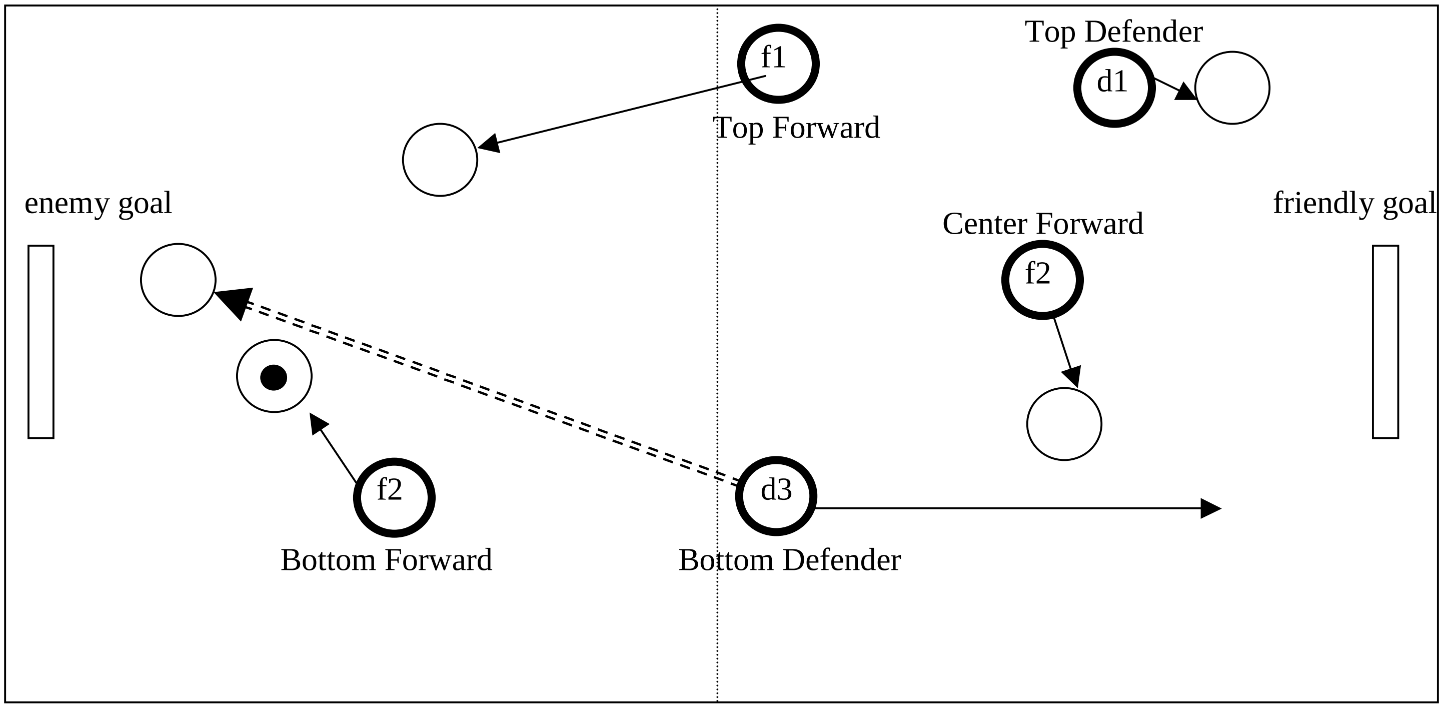


Figure 6: CDA Diagram using Formation1

In the situation shown in Figure 6, thin circles represent the offensive team's players, while the defending team is depicted with thick circles. We are most interested in the team that is currently defending, and its players are labeled. This team is in Formation1 because there are three Defenders and two Forwards. The Defender labeled d3 is told by the Coach to cover an enemy that is on the enemy side of the field (as shown by the double dashed arrow). Instead of guarding his man, d3 will move backward as if his team had the ball. Note that Forwards (like f1) will still cover their man, no matter what side of the field he is on. Testing showed that it's better for defenders to stay back in this way instead of covering players that are not dangerous at the moment. In addition, d3 will not move back too much since the rubber band effect will not let him go too far away from the ball.

**Deciding When to Pass**

CRAI uses the Foundation Architecture's pass decision mechanism without adding anything to augment it. Designing a new centralized algorithm for deciding when to make unforced passes is beyond the scope of this project and is unnecessary because the rules provided by the FA are sufficient for handling this task. In fact, the FA already analyzes the passing option from the perspective of the passer, centering on only his point of view of the situation. This way of looking at the passing problem is in line with basic principles of CRAI, though it does not support the ability to pass the ball when there is no pressure to do so.

# VI Swarm AI: Defensive Coverage

## Swarm AI: Introduction

The Swarm AI architecture works on top of the FA by providing solutions to the 3 soccer sub problems being addressed. The SAI method is based on the idea that it is better to break a problem into parts instead of having one single agent solve the entire problem. While SAI is not modeled after any particular insect behavior, it makes use of the general principles of Swarm Intelligence outlined in the Introduction: multiple agents work on the problem, they have an incomplete view of the environment, and they can communicate. These principles will become apparent in the way the sub problems area handled, in that each of the player agents makes a significant contribution to the emergent solution process. A detailed description of the Swarm AI Architecture's approach to each problem follows. All formulas described in the Swarm sections were determined experimentally through testing to maximize results.

## IPDA: Introduction

Instead of using a single Coach agent to tell each player what to do, SAI delegates the responsibility of deciding who should guard whom to the players themselves. Each shift turn, the players on the Swarm AI team individually decide what they will be covering defensively. They make this decision regardless of whether the other team actually has the ball, letting the FA decide if a player's short-term goal is to guard a target that turn as is outlined in Figure 2. Unlike the CRAI team, the RAI players have 6 choices when it comes to defensive assignments: the 5 enemy players and the penalty area zone near their own goal. Every individual makes this decision by following a process that is called the Individual Paint Decision Algorithm (IPDA).

Each player runs the IPDA before choosing a goal every shift turn, as is shown at the top of Figure 2. IPDA assesses the costs and rewards of each choice, taking into account what the other players are doing (this is where the paint comes in, but more on that later). More specifically, a Swarm player calculates the Profit of covering a certain target, which is equal to a reward number from which the cost of covering that target is subtracted. The following sections describe several heuristics (including those used for cost and reward calculations) that are used by IPDA. The algorithm itself will be described in greater detail after the helper heuristics are explained.

## The Covered Heuristic

The Covered Heuristic is used to determine approximately how open a player is in his ability to receive passes or kick the ball. A player that has no enemies within a radius of a 100 pixels is considered to be completely uncovered, and the heuristic run on him returns 0; it returns 500 for players that are maximally covered. Specifically, for each enemy inside this radius, the Covered value of a player is increased by 100 * (0.97)^(distance between the player and an enemy player). Thus, the more covered a player is, the higher his Covered value.

This formula gives more weight to nearby enemies and barely any to those who are more than 50 pixels away, while ensuring that a single enemy cannot contribute more than a 100 to the covered value. This heuristic ignores the directional position of enemies relative to the player for whom it is run. Nevertheless, it is effective in judging how much open space a player has around him, which helps defenders make decisions about who is a bigger threat.

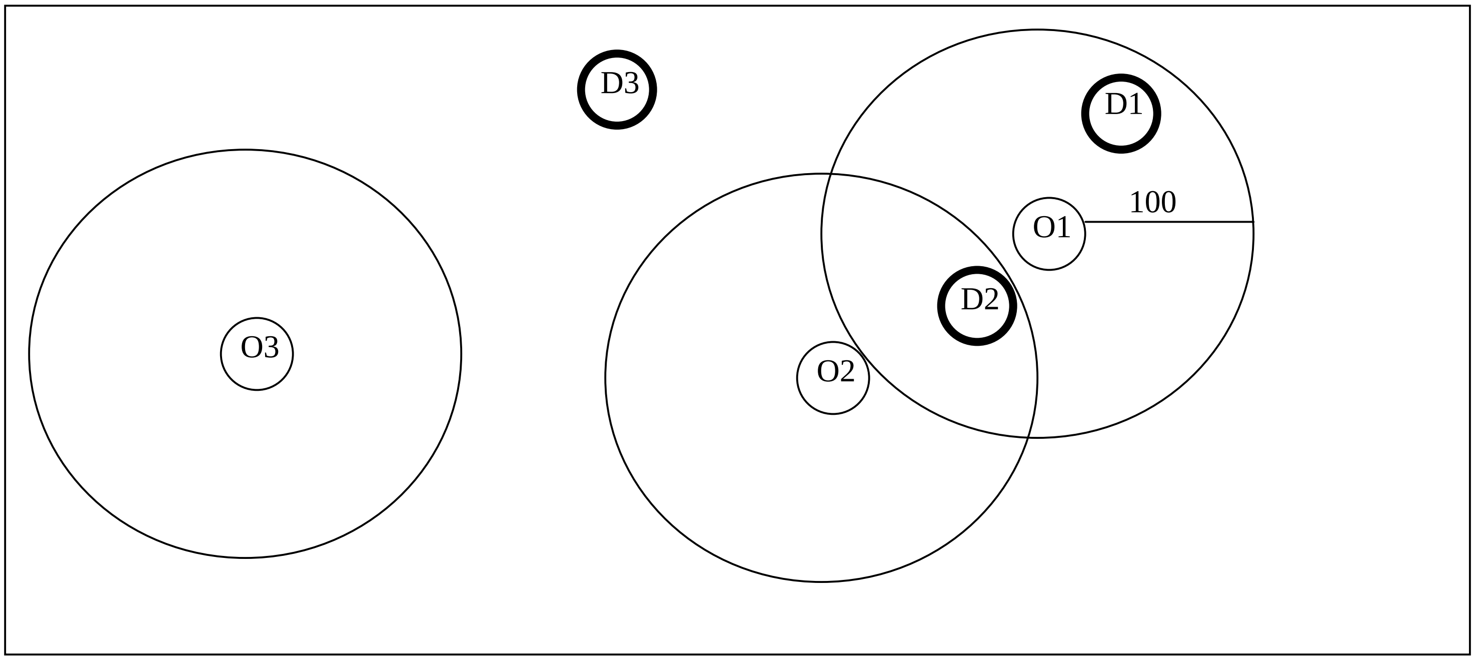


Figure 7: Covered Heuristic Example

In the situation shown in Figure 7, there are 3 offensive players (O1, O2, and O3) shown by thin circles and 3 defensive players (D1, D2, and D3) shown by thick circles. Here the Covered Heuristic is run for all 3 offensive players, and the circles around them highlight the area of concern (with radius 100) around those players. O1 would have a high Covered Heuristic value because he has two defenders in his vicinity, D1 and D2. D2 is closer to O1 than D1, so it would contribute more weight to that value. O2 has only D2 in its vicinity and D2 is not that close, so it will have a smaller Covered value than O1. Finally, O3 has no enemies nearby so its Covered value is 0. Notice that D3 is not close to any offensive players, so it does not contribute to their Covered values.

## The Reward Heuristic

The Reward Heuristic calculates the utility of covering an enemy player (let us call him Player A). This heuristic takes into account 3 factors about Player A: his openness, his distance to the opposing team's goal, and whether he is carrying the ball. This heuristic is used by Swarm defenders to decide what offensive players are most worth being guarded. In other words, the higher the reward, the more important it is to guard that player.

The reward takes into account openness by running the Covered Heuristic and inverting the results such that players with low Covered values who are very open receive high reward values. The maximum reward due to openness is 150, while this reward may be negative if Player A is not open at all (he has a high Covered value). Therefore, a higher reward is given for covering players that are more open. The exact formula used for this purpose is  Reward = 3 * (50- Covered _Heuristic(target) / 2).

The second value contributing to the reward is the distance of Player A to the other team's goal, which ranges from 200 for players close to the other team's goal to negative numbers for players that are near their own goal. This value is directly proportional to Player A's distance from the goal, such that a larger reward is granted for covering players that are dangerously close to the other team's goal. The formula used for this purpose is Reward = Reward + (200- Distance_To_Enemy_Goal (target) / 2 ).

If Player A is currently carrying the ball, he is much more capable of scoring or assisting in scoring than his teammates and should be covered as soon as possible. The reward for covering a ball carrier is taken into account by tripling the reward from the first two factors, such that they become three times larger if Player A is the ball carrier and remain unchanged if he is not. It is important to note that this factor does not add a constant value to the previous reward figure but magnifies it instead. Thus, there is a larger "ball carrier reward" for guarding wide open ball carriers that are only a short distance away from the other team's goal, than those that are close to their own goal or are already well covered. So if the target is a ball carrier,  Reward = Reward * 3.

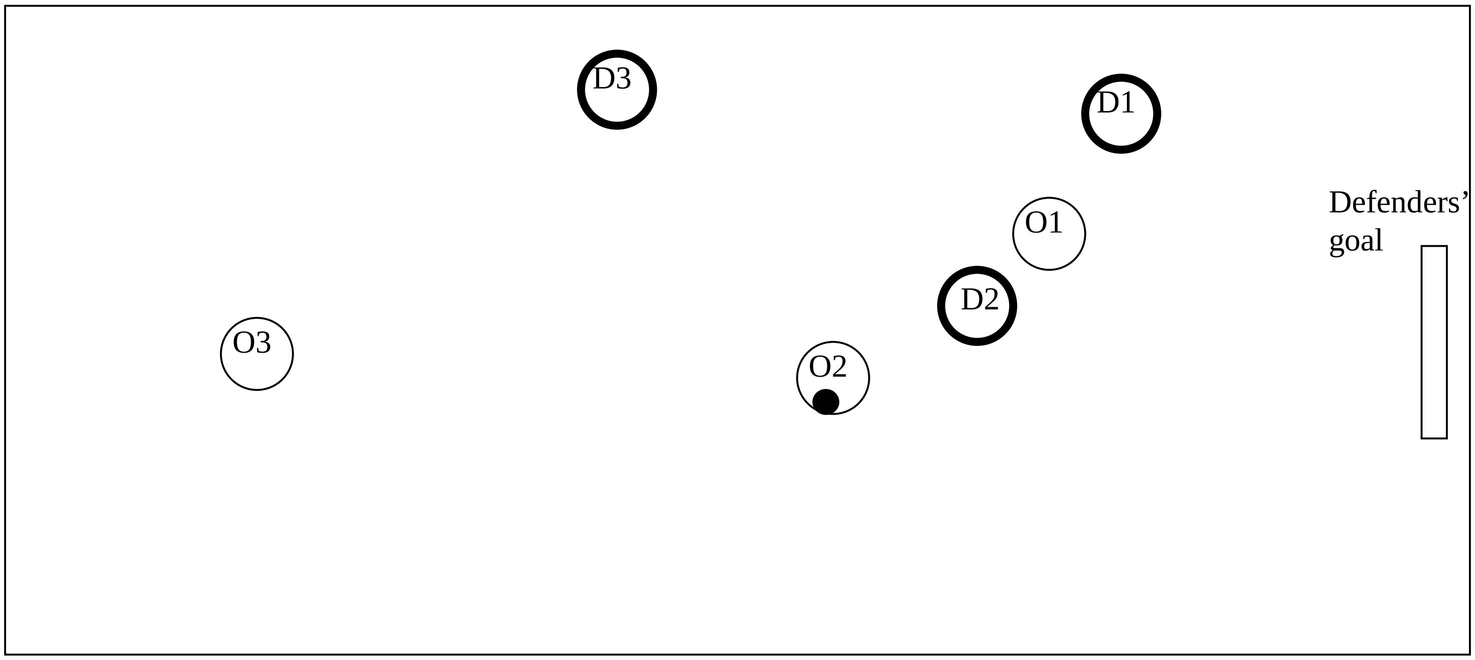


Figure 8: Reward Heuristic Example

Figure 8 shows the same situation as Figure 6, only this time there is a goal the offensive players want to score on and O2 is now a ball carrier. Let's see what happens when the Reward Heuristic is run on each of the offensive players. O1 is not very open, doesn't have the ball, but is close to the enemy goal so it will have a medium Reward value. O2 is somewhat open, not too far away from the goal, and is a ball carrier so it will have a large Reward value. Finally, O3 is very open but doesn't have the ball and is far away from the enemy goal so it will have a small Reward value because it is not much of a threat right now.

## The Cost Heuristic

This heuristic estimates how hard it will be for Player B to guard an enemy Player A by looking at about how long it will take Player B to intercept Player A. If an interception is possible within a finite amount of time, the time Player B will spend moving before he reaches Player A is returned as the cost. However, if an interception cannot occur given the current position and velocity of both players, the heuristic instead returns the following number: (distance between the two players)/(the speed of a player) which gives a less accurate and more optimistic approximation of how long it will take Player B to reach Player A. The assumption here is that Player A will stop or change directions, which will make it possible to catch him; this is a fair assumption since the players are forced to do so when they reach the boundaries of the field.

Since the directional velocity of Player A can change at any time, the cost of covering him may also change by a large amount at any shift turn. Nevertheless, this heuristic is sufficient since players typically move in straight lines, rarely changing their velocities. Ultimately, the Cost Heuristic returns large values if Player A is far away or is moving in a direction opposite where Player B is located. On the other hand, if Player B is relatively close to Player A, there is only a small cost in him guarding Player A.

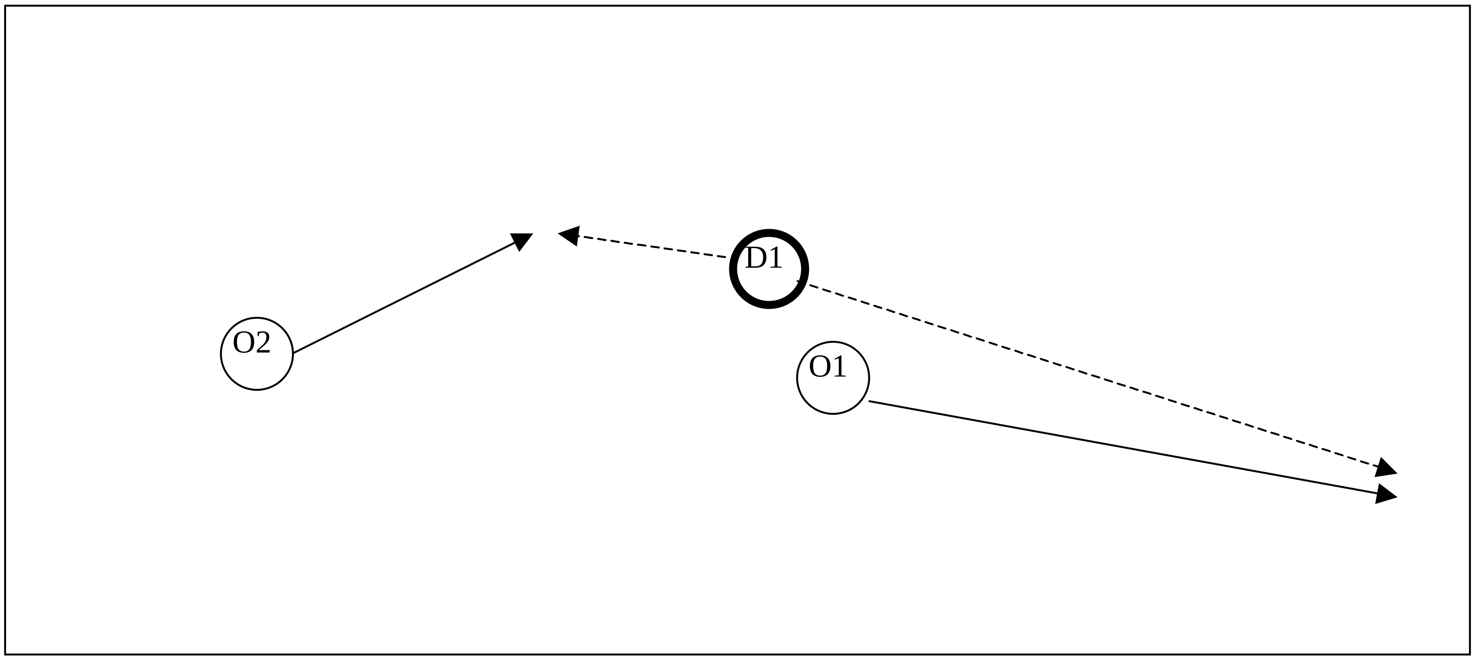


Figure 9: Cost Heuristic Example

Figure 9 shows a simple situation where the defender runs the Cost Heuristic for two offensive players. The arrows away from the offensive players show their projected paths in the direction of their current velocities, and the dashed arrows away from the defensive player show the possible projected paths of the defender for intercepting a particular offensive player. Even though O1 is close to D1, because O1 is heading away from D1 it will take a while for D1 to intercept it, so the Cost is high. On the other hand, O2 is headed toward D1 so the Cost value for guarding O2 is small.

## IPDA: the Details

For each Swarm player, the IPDA calculates the Profit of guarding each target. The possible targets are all 5 of the enemy players and the penalty area zone near one’s goal. Each target can be “painted” a certain color by a player to show his teammates that it is being covered. The presence of this “paint” on a target is a factor when the Final Profit for guarding targets is calculated, as there is less Profit for guarding a target already painted by someone else. When a Swarm player decides to cover a target, he paints that target with his color such that his teammates know who is covering the target simply by looking at the color value of its paint. When another player decides to cover an already painted target, the old paint is replaced with a new one corresponding to this other player's color. The paint does not degrade over time, but it is removed when the player it corresponds to decides to cover another target. A target can only be painted with one player's color at any one time, so the previous color is erased, which means the owner of the previous color will be less likely to guard this target the next shift turn because someone else has painted it.

The IPDA works by having the current player (the player running IPDA) first assume an Initial Profit of 100 for guarding the penalty area zone and divide that number by 4 if another player has already painted it. Next, the player calculates the Profits of guarding all the other targets. He begins by running the Reward Heuristic and subtracting from its value the number returned by the Cost Heuristic to obtain the Initial Profit. Furthermore, this Initial Profit is divided by 4 if another teammate has painted the current target and the target is not a ball carrier; the Initial Profit is only divided by 2 if the already taken target is a ball carrier. When the Final Profits have been calculated for all of the targets, the player covers the target offering the maximum Profit, painting it in his color to let his teammates know he is guarding it and removing his paint from a previous target if that is necessary. The constants in the above calculations have been determined experimentally to provide the best results.

IPDA works as described above with one exception that has a 1 in 15 chance of occurring every shift turn. Every once in a while, a player will cover and paint the ball carrier target, even though it may not offer the maximum Final Profit. This small element of randomness is important because it ensures that the Swarm team doesn't become predictable or locked in an infinite loop of behavior. Also, note that the painting procedure described above takes place regardless of what team has the ball and a player is not obligated guard an enemy he has painted. The paint is only a communication mechanism that helps other players make decisions.

| Target | Reward | Cost | Initial Profit | Ball Carrier? | Painted By | Final Profit |
|---|---|---|---|---|---|---|
| Penalty Zone | - | - | 100 | no | None | 100 |
| O1 | 200 | 40 | 160 | yes | D2 | 80 |
| O2 | 150 | 30 | 120 | no | D3 | 30 |
| O3 | 240 | 50 | 190 | no | none | 190 |
| O4 | 60 | 80 | -20 | no | none | -20 |
| O5 | 40 | 10 | 30 | no | D1 | 30 |

Figure 10: Table Showing IPDA for D1

Figure 10 is a table that shows the values calculated when IPDA is run by player D1 for some arbitrary situation where the other team has the ball. D1 needs to figure out who he should guard based on Reward and Cost values of each target, as well as what targets his teammates have painted and whether a target has the ball or not. The first target, the Penalty Area Zone, always has Initial Profit of 100, which is also the Final Profit in this case because no one has painted it. O1 has Reward 200 - Cost 40 = Initial Profit 160. But that Profit is divided by 2 since D2 has painted O1, which is a Ball Carrier. O2 has been painted by someone else as well, but its Initial Profit of 120 is divided by 4 since O2 is not a ball carrier. O5 has been painted but its Initial Profit is the same as the Final Profit since the paint belongs to D1, for whom this instance of IPDA is being run. It is possible for a target to have a negative Profit, and O4 is one such target because it is probably already well covered, is near its own goal, and is far from D1. O3 offers the maximum Final Profit, so it is the target D1 must choose to paint. As mentioned above, the FA decides if a player will actually guard a target, but IPDA decides what the target will be if the player ends up guarding something that shift turn.

### IPDA: Discussion

The IPDA works with more targets than there are players on a team (6 instead of 5), which encourages the players to spread out in their responsibilities. The penalty zone provides a constant and fairly generous profit since it is a good idea to have someone in the back in case the other team breaks through the front lines of a defense. In addition, the paint mechanism allows the Swarm players to communicate such that their decision-making is not entirely selfish and they take into account what their teammates intend to do. Thus, while every Swarm player assesses the situation from his own point of view in trying to maximize his own profit, that profit is maximized when the decisions of teammates are taken into account (i.e. who they have painted). Furthermore, by overwriting someone else’s paint, a player encourages that teammate to cover another target by decreasing the profit of guarding the original target; the profit is decreased because that teammate will see that someone else has already painted it. Thus, the Swarm players can hold these “painting dialogues” on the fly about who should be covering whom by painting the targets to express their opinions. These dialogues take place even when the Swarm team has the ball, so its players are thinking about defensive coverage before they actually need to defend anyone.

This system of dealing with defensive coverage is simpler than the centralized coach algorithm the rule-based teams use because the problem is being solved from a more narrow perspective of a single agent. Unlike the coach, that agent is not concerned with broad goals like covering every single enemy player. Furthermore, the agents don’t have to look at the entire field and determine where they have to be in relation to every other player on the field. Instead, that player is only concerned with maximizing his own usefulness, which is a much easier problem to solve. When several Swarm players are independently trying to be useful, the emergent effect is that they are actually working as a team to solve the much harder problems mentioned above. In addition, the painting system provides a simple way for players to know what their teammates are doing so that each player doesn’t have to determine for himself what all of his teammates are doing every shift turn.

At the same time, the Swarm defense is more robust in being able to handle unforeseen situations the programming of the coach doesn’t take into account. If some of the Swarm players are doing the wrong thing, some other agents can let them know about it during the next shift turn by overwriting their paint. On the other hand, if the coach makes a mistake and tells all of his players to do something, no other agent can let him know a mistake was made. The CRAI players are not autonomous and must guard whomever their coach tells them to. Therefore, the entire centralized defensive coverage system relies on the AI engineer’s design of the coach, which cannot possibly take into account all the different situations in a soccer game. The coach is forced to make generalizations, which frequently do not yield the right coverage assignments for all of his players.

# VII Swarm AI: Offensive Player Movement Without the Ball

### Making Runs

In soccer, it is a good idea for players without the ball to make runs in order to become more open and receive a pass. SAI attempts to implement this behavior for the players without the ball when their teammate has the ball by having these players make individual decisions about where to move. When a Swarm player realizes that his teammate has (or is about to take) possession of the ball and is not passing the ball to the player in question, his short-term goal is to make runs (the player would try to intercept the ball if it was passed to him). Each offensive player (each player on the team with the ball) who does not have the ball will take into account his position, the positions of other players, where he is in relation to the goals, and the position of the ball to come up with a direction vector for where that player should move. These vectors can point in any direction and offer players much more freedom in comparison to the limited movement options of CRAI players (i.e. horizontal and diagonal). However, because these vectors can point in any direction, the challenge was to come up with a way for players to pick movement vectors that would be effective.

Like CRAI, the SAI Architecture makes use of roles for its players. Again, there are just two roles: Forward and Defender; they are used to specify what kind of run players should use when they want to move without the ball on offense. This is necessary because some of the players should move toward the enemy goal while others should move back toward their own goal in case their team turns the ball over and someone needs to play defense. During a shift turn, a Swarm player can be assigned as a short-term goal to make either Defender or Forward runs, depending on his role on the team. Thus, the player's goal for the next 10 turns is to make runs, and during each of these turns the player calculates the velocity vector for movement by running either a Forward or Defender Runs algorithm.

## Forwards

In deciding where to move, SAI Forwards try to move away from enemies toward the ball and the enemy goal, running the Forward Runs Algorithm (FRA) on themselves. The following is an outline of FRA and the individual steps will be discussed below:

```
1) Reset the Summation Vector to (0,0)
2) For each of the 5 distance vectors from an enemy player to the Forward
        2.1) If the magnitude of the distance vector is greater than or equal to 200,
             move on to the next vector
        2.2) Normalize the distance vector and multiply it by (200 - magnitude of the original vector)
        2.3) Add this new vector to the Summation Vector
3) Summation Vector = Summation Vector + vector from Forward to enemy goal
4) Summation Vector = Summation Vector + vector from Forward to the ball
5) The Forward moves in the direction of the Summation Vector
```

The simplest and most effective way to move away from an enemy is to move in the direction of the vector from that enemy to the Forward. If there are several enemies, then their vectors can be added to produce a Summation Vector, which provides a direction that is optimal for moving away from all enemies. While calculating the Summation Vector, a Forward wants to push away from all enemies that are within a distance of 200 pixels from him, while ignoring those that are farther away by not adding the vectors associated with them. Also, the normalized vectors are multiplied by (200 - their magnitude) to give more weight to vectors that are smaller since a Forward wants to push away more from enemies that are close because closer enemies are more able to cover the Forward. Here, the inversion of the magnitude emphasizes the need to move away from closer enemies. The distance of 200 pixels is being used because experimentation has shown it to be a good cutoff for not worrying about enemies beyond that distance.

A Forward also wants to move toward the enemy goal so a vector toward it is added to the Summation Vector, such that the farther away a Forward is from the goal, the more he will try to move toward it. This makes sense since Forwards want to be close to where they can score and where a pass to them will advance the ball forward. Finally, a Forward wants to move toward the ball so a distance vector to it is added to the Summation Vector, such that the farther away the ball is from the Forward, the more important it is for the Forward to move toward it. This is necessary because a Forward wants to receive a pass and it is a good idea to be close enough to the ball carrier to receive it (and not be somewhere on the other side of the field away from the action around the ball). The Summation Vector produced by the FRA is then the direction in which the players move.

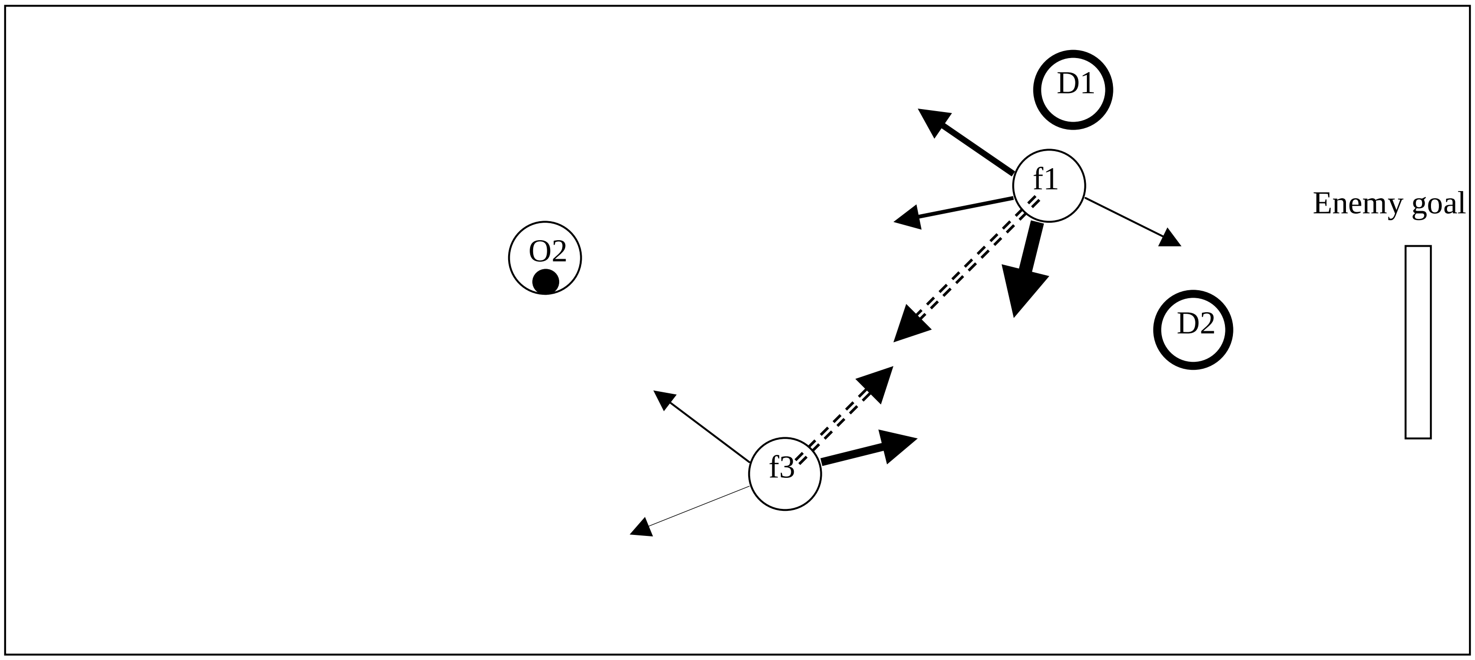


Figure 11: FRA Example with 2 Forwards

Figure 11 shows a situation where there are three offensive players (f1, O2, and f3) and two enemies (D1 and D2). f1 and f3 are Forwards; it does not matter if O2 is a Forward or Defender in this case since as a ball carrier he will not make runs and has been generically labeled O2 to highlight this fact. FRA is run for f1 and f3 separately, who are moving without the ball, which is currently possessed by O2. The single-line arrows coming out of f1 and f3 are the vectors that add up to become the Summation Vector, which is shown by the double-dashed arrow. Note that the vectors shown by thicker single-line arrows have greater magnitude and contribute more to the Summation Vector than the vectors with thinner arrows.

f1 is very close to D1 so it will try hard to go away from it as is shown by the thick arrow attached to f1 and pointing away from D1. The arrow attached to f1 and pointing away from D2 is thinner because D2 is not as close to f1 as D1, and thus, f1 wants to push away form D2 less. Both f1 and f3 have arrows pointing to the ball carrier, O2, and the enemy goal. Notice that f3's arrow toward the enemy goal is thicker than f1's because f3 is farther away from that goal. f3 does not have an arrow pointing away from D1 because D1 is too distant (more than 200) to make f3 want to run away from it. f1 and f3 will move in the direction of their Summation Vector this turn and will run FRA for themselves again the next turn provided that O2 maintains possession of the ball.

## Defenders

When SAI Defenders make runs, they move away from their own teammates, toward the ball, and toward their own goal. They calculate a Summation Vector by running the Defender Runs Algorithm (DRA) as described below:

```
1) Reset the Summation Vector to (0,0)
2) For each of the 4 distance vectors from a friendly player to the Defender
        2.1) If the magnitude of the distance vector is greater than or equal to 200,
             move on to the next vector
        2.2) Normalize the distance vector and multiply it by (200 - magnitude of the original vector)
        2.3) Add this new vector to the Summation Vector
3) Summation Vector = Summation Vector + vector from Defender to friendly goal
4) Summation Vector = Summation Vector + vector from Defender to the ball
5) The Defender moves in the direction of the Summation Vector
```

Unlike the Forwards, Defenders want to push away from their teammates instead of enemies. This ensures that the players are spread out to give the passer different passing options by not having Defenders bunch up next to other teammates where a single enemy can cover more than one of them. This behavior does not force the Defenders to move far away form enemies such that they can guard them in the near future if they need to. For Forwards, staying next to enemies for defense is not a priority, so for them it is a good idea to push away from enemy players. The way in which vectors pointed away from teammates are added up to produce the Summation Vector is similar to steps 2 through 2.3 of FRA and will not be discussed here.

While Forwards try to move toward the enemy goal, Defenders want to move toward the friendly goal. This is necessary because Defenders should stay back in case their team loses the ball and it is necessary to prevent the other team from scoring. The farther away a Defender is from his goal, the more he will want to get back because the vector toward the goal will have greater magnitude. Finally, the DRA shares step 4 with the FRA, since Defenders also want to move toward the ball to be close enough to receive a pass.

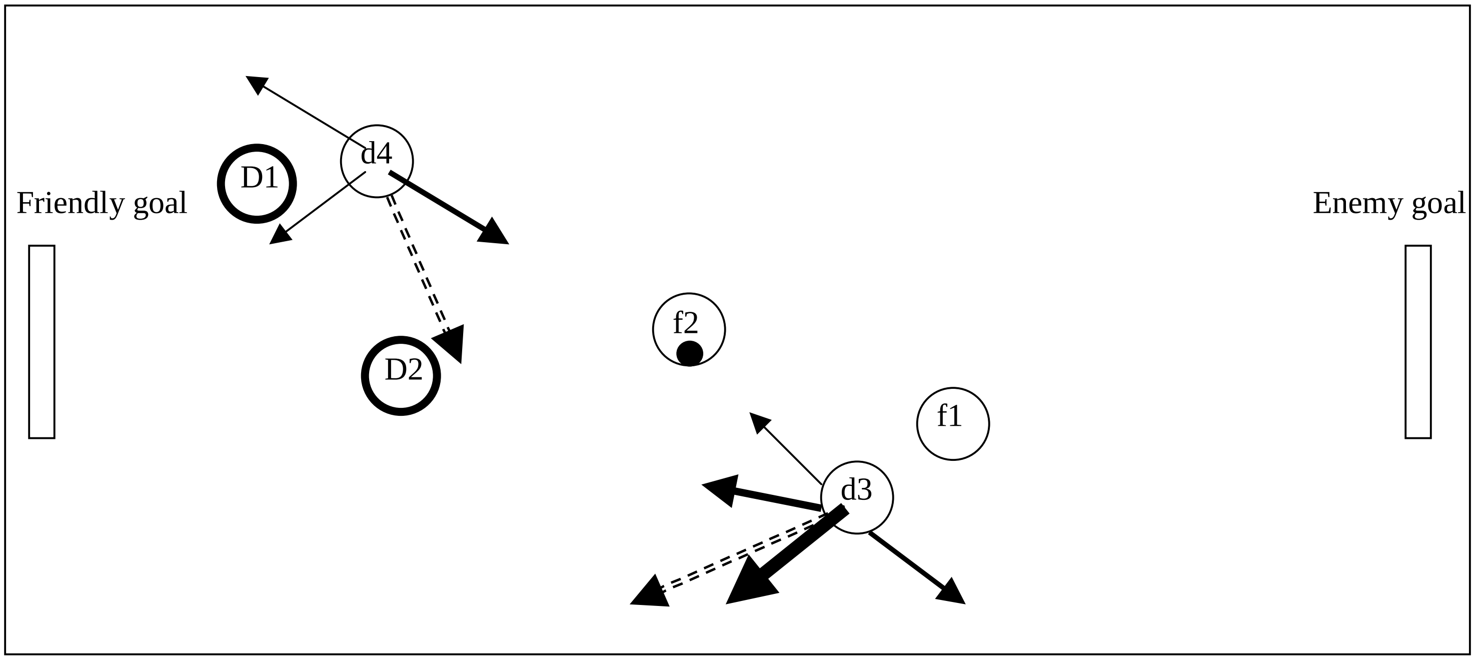


Figure 12: DRA Example with 2 Defenders

Figure 12 shows a situation with 4 offensive players (f1, f2, d3, and d4) and 2 defensive players (D1 and D2). FRA is run for the 2 Defenders on the offensive team (the team with the ball), d3 and d4. d4 is pulled slightly away from f2 because f2 is a friendly player that is far away. At the same time, d4 is pulled toward f2 because f2 has the ball, and d4 is also pulled toward the friendly goal. The Summation Vector for d4 points in a direction that would keep d4 eligible for receiving a pass while also keeping d4 ready to play defense if it is necessary. d3 is pulled away from f2 and f1, both of whom are close to it. Also, d3 really wants to move back toward the friendly goal because he is far from it. Finally, d3 wants to move toward f2 just like d4 does since f2 has the ball. The Summation Vector for d3 combines all of these intentions into a single movement vector d3 will follow this turn.

Like CRAI Defenders, Swarm Defenders don't cover their target when it is a player on the enemy side of the field, as is outlined in Figure 2. In this situation, which is quite rare because in most cases Defenders will choose to either double team a more dangerous enemy or guard the penalty zone, a Swarm Defender will make the Defender Runs being described in this section instead of guarding a target he has painted. The Defenders are made to behave this way because it keeps them from moving too far away from their own goal.

### Discussion

The making runs component is probably the least obvious, when it comes to seeing its relation to Swarm principles. Here, the Swarm approach does break up the offensive movement problem into parts by having the individual players decide how it is best for them to move. Also, SAI only forces the player agents to consider 3 factors like wanting to move toward a goal and the ball and away from one group of players, ignoring all other details about the situation. However, it is not immediately clear how the players communicate with one another. One can argue that for Defenders the communication takes place when they push off other Defenders. In this way, one Defender can tell a second move in a certain direction by causing the second to move away from him in a certain direction. However, this is a fairly indirect mode of communication and Forwards certainly do not communicate with their teammates since they push off enemies. Consequently the SAI solution to offensive movement is not entirely based on Swarm Intelligence since it somewhat neglects the communication aspect of Swarm AI. This offensive runs behavior is more closely related to flocking, which is the behavior of birds and fish but not insects.

# VIII Swarm AI: Deciding When to Pass

### Introduction

SAI passing is based on the idea that a player should be able to pass the ball to a teammate at any time, regardless of whether the ball carrier is pressured to do so. This may seem obvious, but it is not simple to implement this behavior in practice. Like CRAI, SAI makes use of forced passing as it is described in the Foundation Architecture's section. Furthermore, SAI uses what is called an Openness Painting System (OPS) to allow the ball carrier to make unforced passes. OPS introduces two components to agent behavior. First, all the Swarm players look at their own openness and paint themselves if they are open. Second, a ball carrier looks at how many of his teammates have painted themselves open and decides if he should pass the ball. Both of these behaviors are described below in detail and also listed in Figure 2

## Openness Self-Painting

Every shift turn before picking a goal, each of the Swarm players individually runs the Openness Self-Painting Algorithm (OSPA) on themselves to decide if they are eligible for receiving a pass. This can be accomplished by checking if a player can intercept one of the many possible passes from the ball carrier, but such calculations take too much time and it would be wasteful to have each Swarm player do them every shift turn when it is possible to make a simple estimation about whether a player is in a good position to receive the ball. The OSPA is the simple, low-performance cost Swarm solution to this problem, having players paint themselves to show the ball carrier that they can receive a pass. The paint used in this algorithm degrades over time such that a person's paint status ranges from 0 (not painted) to 3 (fully painted). The steps of OSPA are as follows:

```
1) If player's paint status > 0 => player's paint status decreases by 1
2) OSP Value = 50 - (Covered Heuristic(player) / 2)
3) If OSP Value > 20 =>
    3.1) If player's paint status is 0    =>  player's paint status becomes 3
    3.2) Else if player's paint status > 0  =>   player's paint status is incremented by 1
```

OSPA begins by decrementing a Swarm player's paint status by 1. The paint degrades in this fashion from 3 to 0 over several shift turns instead of simply having a binary value of 0 or 1. Next, OSPA calculates what is called an Openness Self-Painting (OSP) Value which represents how open a player is. Much like calculations for the Reward Heuristic, finding the OSP Value involves inverting the Covered Heuristic Value by subtracting it from a constant. This means that players with high OSP Values have low Covered Heuristic Values (they are very open). If the OSP Value is higher than a certain threshold (20 in this case), then the player will be painted. If the player currently has no paint, then he will be fully painted. However, if the player is currently wearing paint, then his paint value will just be incremented by 1 without necessarily restoring it to 3.

Experiments have shown that it is better to have paint degrade instead of disappear when he is not open since a player's OSP Value may be low now but the player may still be open if he was recently open. This is because the OSP Value (because it is based on the Covered Heuristic) is determined by how many enemies are in the area around a player and not on whether any enemies are between the player and the ball carrier. So an enemy might be close on right side of a Swarm player and the ball carrier on the left, and the OSP Value would incorrectly suggest that the player could not receive a pass. Since we know that player was open earlier, it is quite likely that he may still be open and the OSP Value, which is only an estimation of events, is wrong. However, after two turns we cannot assume that the OSP Value is still wrong and probably the player is well covered, so the paint degrades completely by that time.

An already painted player whose OSP Value is high enough to paint him is not automatically given a full coat of paint for several reasons. If the player was recently at 3 and is now at 2, his paint will be restored to 3 anyway by the increment of 1. However, if that player has less than 2 paint it means he was recently found to be not open so it is probably a bad idea to give him a full 3 paint status since his openness situation may be uncertain. Again, it is important to note that the OSP Value is just an estimation and the several values of paint serve to make up for discrepancies between the OSP Value and whether a player is actually open. Finally, Swarm players paint themselves even when their team doesn't have the ball so that when their team obtains it the paint status numbers reflect how open the players were the last few turns, even though their team did not have the ball then.

## Paint Passing Algorithm

As shown in Figure 2, a Swarm ball carrier will consider unforced passing as an option only if he cannot shoot on goal and doesn’t have to make a forced pass to get rid of the ball. To decide whether it is a good idea to make an unforced pass a Swarm ball carrier runs the Paint Passing Algorithm (PPA) which returns True if a pass should be made and False otherwise:

```
1) Num_Painted = How many teammates have paint status greater than 0
2) Randomly decide if a pass should be made with (Num_Painted * 5) % probability of deciding to do it.
3) If ball carrier decided to pass
    3.1) Run the Directional Passing Algorithm to see if a pass is feasible
    3.2) If a pass is feasible  =>  return True
    3.4) Else  =>   return False
4) Else  =>  return False
```

This simple algorithm involves counting how many teammates are open and making a random decision about whether to pass based on that number. The more players are open, the more likely it is for a ball carrier to make that pass. PPA runs the Directional Passing Algorithm to make sure that a pass is actually feasible despite the indications of the painted players. Usually, the teammates’ estimation of the situation is accurate and DPA is not run needlessly (as it is a more computationally intensive algorithm, considering up to 16 possible passes). If PPA returns True, then the goal of the ball carrier that shift turn is to make a pass in the direction specified by DPA. Otherwise if PPA returns False, the ball carrier continues down the Else-If ladder described in Figure 2 without making a pass.

## Discussion and Example

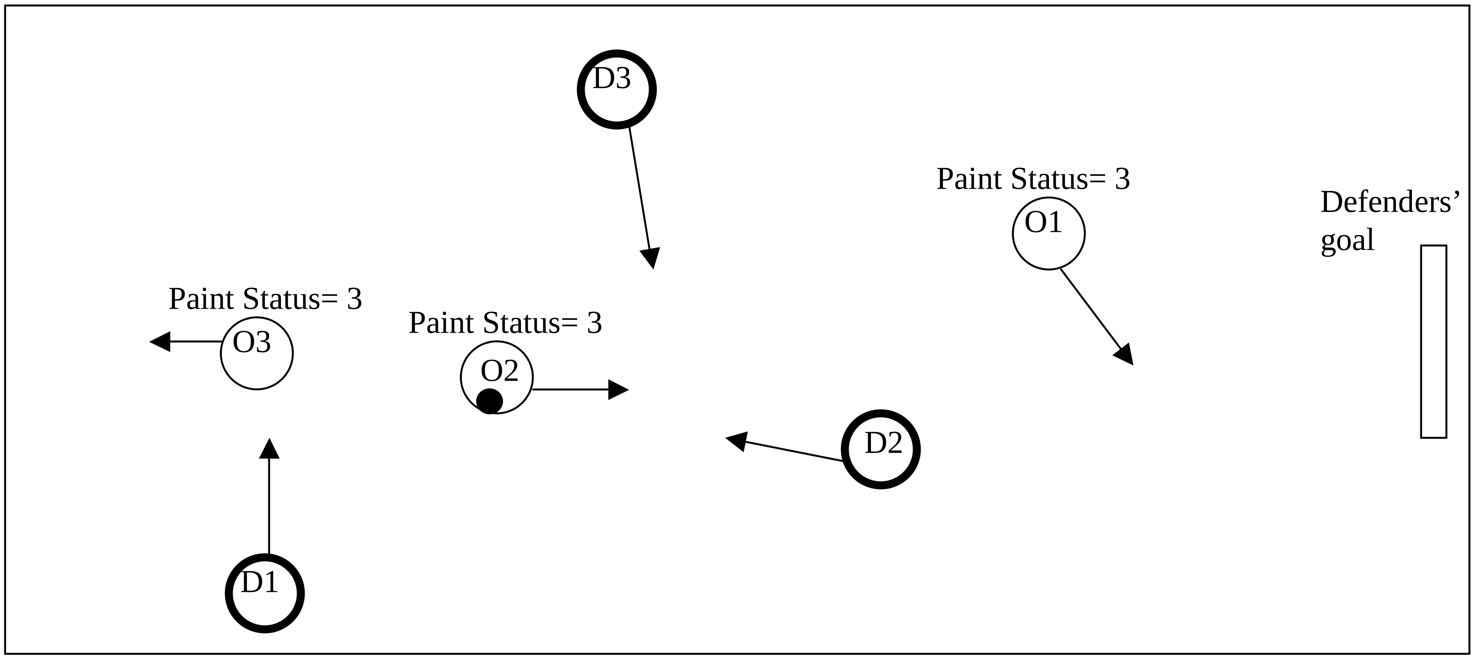
The SAI solution to passing is very much based on Swarm principles. The problem of when to pass is not simply given to the passer, but distributed among the whole team. Each player looks at his situation and decides if he is open, so that the passer doesn’t have to make that judgment for each teammate. By painting or not painting oneself, a player communicates to the passer whether he is open or not. This strategy is more effective than CRAI simply because it allows for passes when the ball carrier is not threatened. However, this additional functionality comes at a very cheap price in performance since running OSPA for each player and the ball carrier counting the number of painted teammates each shift turn takes very little work.

At this point it is important to clarify that there does not exist any relation between the painting used for passing and the defensive coverage painting. For defensive purposes, Swarm players paint enemies they intend to cover, and that kind of paint is ignored when the ball carrier is deciding whether he should pass the ball to a teammate.

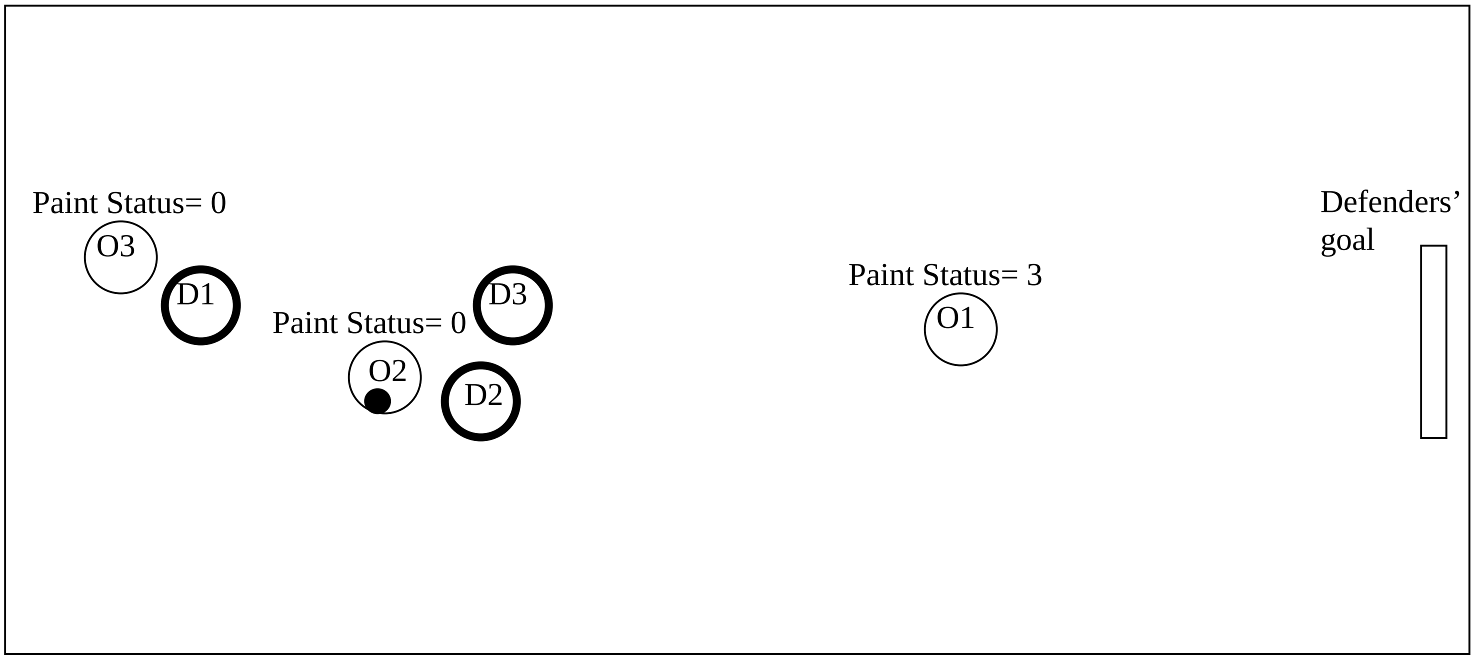
Figure 13 shows a situation where there are 3 offensive players (O1, O2, and O3) and 3 defensive players (D1, D2, and D3). The situation is first shown at shift turn 1. Then, the situation is shown at shift turn 5 assuming that unforced passes cannot be made as is the case for CRAI. Finally, the situation is shown at shift turn 5 as it would happen given that SAI supports unforced passing. O2 is the ball carrier and O1 and O3 are potential pass receivers. O1 is behind O2 while O3 starts out in a good position close to the enemy goal. At shift turn 1, all 3 offensive players are open so they paint themselves to make their paint status 3. Notice that while O2 paints himself, this will have no effect on his decision process as long as he is the passer. The arrows in the shift turn 1 diagram show the direction in which the players are moving.

O2 does not have any enemies nearby, so without unforced passing he would keep moving forward, ignoring O1 who is open close to the enemy goal. The Shift Turn 5A diagram shows the results as O2 becomes trapped when it is too late to make a pass to O3 or O1. Furthermore, O2 had to move back to avoid D3 and D2, losing the ground it gained by moving forward with the ball. In that diagram, notice that O1 is still has paint state 3, but that is of no help since O2 cannot make a pass to him.

**Shift Turn 1**



**Shift Turn 5A**



**Shift Turn 5B**

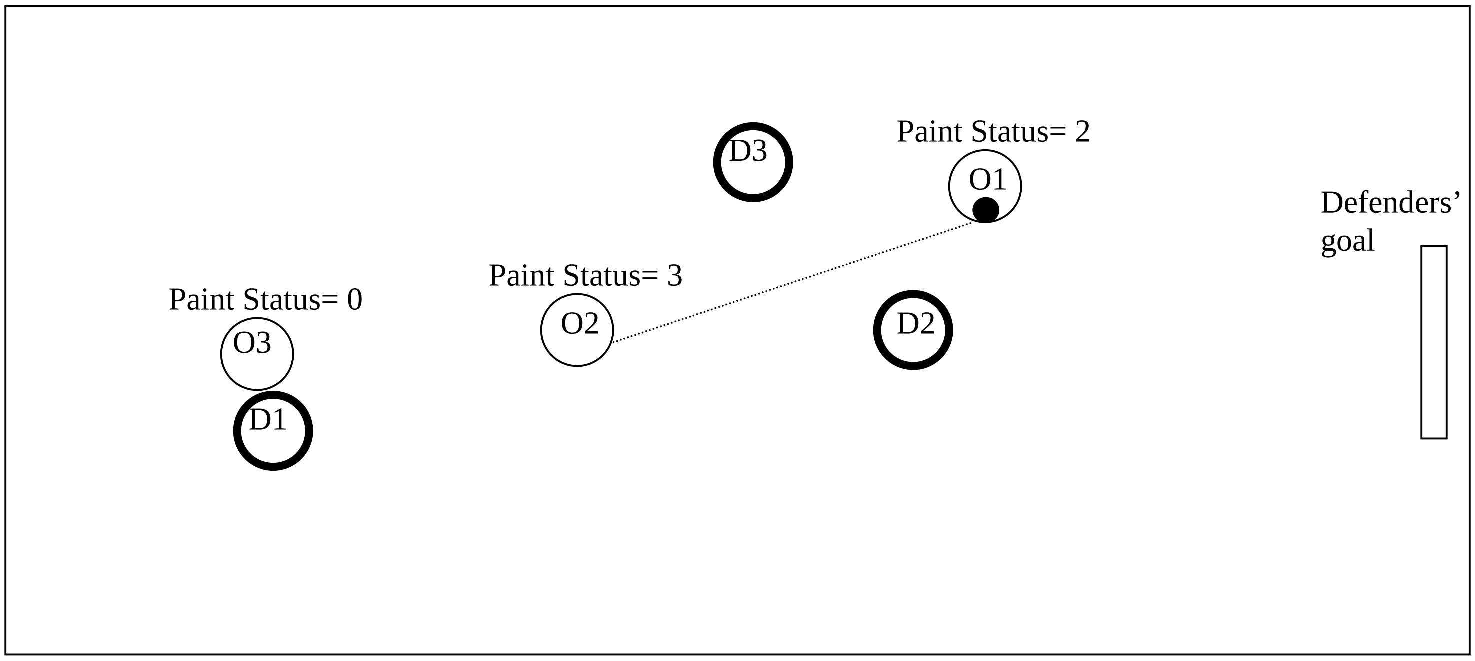


Figure 13: Unforced Passing Example: Situation Shown in Sequence for 3 Different Shift Turns

The Shift Turn 5B diagram shows what could happen for a SAI offensive team. I used the word 'could' because O2 may or may not pass to O1 or it may pass to O3. However, in this situation it is likely that the ball would be passed to O1, given that DPA favors passes that advance the ball toward the enemy goal. The dashed line shows the path of the ball as it travels from O2 to O1, between the two defenders. This pass puts the offensive team in a good position, since now O1 has the ball close to the enemy goal. Therefore, SAI's use of unforced passing clearly gives it an advantage over CRAI. Notice that O1's paint status is 2, which signifies that it has recently become not open. This number will probably degrade to 0 since as the ball carrier O1 will likely draw attention to itself. Nevertheless, this is not a bad thing since O1 is no longer a receiver and its paint status will not matter as long as it has the ball.

The example mentioned an important issue about SAI passing. Step 2 of the PPA has a strong random component, which means that SAI will not always do the right thing for some situation. In the above example, there were 2 open teammates, so there was only a 10% chance the pass would be made that shift turn, which suggests that only 10% of the time SAI would do the right thing. However, this is not the case since usually a ball carrier has more than 1 shift turn in which to make a pass. In this case, it is likely that the passing lane from O2 to O1 would still have been open for another 2 shift turns. Assuming this is true, SAI would make a mistake and not pass the ball only 0.9 * 0.9 * 0.9= 72.9% of the time. This still might seem fairly bad and one might think that increasing the probability of passing per player from 5 to a larger number might improve performance. However, experiments have shown that using larger constants for this purpose lessens a ball carrier's ability to move the ball on his own and causes more passes to be made with little payoff. As discussed in the FA section, passes can be intercepted, and therefore, it is always safer for a player to move the ball on his own than it is for him to pass the ball and risk having it intercepted. On the other hand, passing is a faster way to move the ball forward, so there are trade offs to be considered here. Experiments have shown that having a 5% pass probability per player strikes the optimal balance between risking a pass and having the ball carrier move the ball.

# IX The Experiment

**Criteria for Comparing AIs**

The most obvious way of comparing the effectiveness of the two AI architectures was to have each control a team of soccer players and to play them against each other. When two soccer teams play, an observer can tell which team is winning by looking at the score but this does not mean the winning team is better than the losing team. A score of 2 to 1 at the end of a game does not indicate that the team that won would win again if another match were played. Therefore, to test the AIs, I found it necessary to have very long games and make sure that many games were played. Each test game lasted for 1 million turns and 100 of these games were played between two teams that were being compared. For sake of convenience, I will from now on refer to the number of turns as time and measure time in 10-turns such that one 10-turn is equal to 10 turns; thus, each game lasted for 100,000 10-turns.

One simple way to see which team is better is to look at how many games each wins after playing a 100 of them. However, this result alone is not enough since the games in this experiment are very long and the better team will almost always win. Consequently, this statistic is not enough to determine how the teams compare beyond which is better. To judge how much better one team is than another, I looked at how many more goals the winning team scored at the end of the match. This takes into account both a team's offensive and defensive ability, since each team tries to raise its own score and keep the other team's score low. Thus, the score disparity shows approximately how much better the winning team was on defense and offense (but it cannot be used to compare the defenses or offenses of 2 teams separately). To ensure that the average scores are accurate from game to game, the standard deviation for each team's average score over 100 games is calculated. The standard deviation determines how far apart are actual scores from the average; if they are close to the average than the average is accurate. The standard deviation is equal to the square root of m2 (the variance), which is calculated using this following formula:

$$m_2 \equiv \frac{1}{N}\sum_{i=1}^{N}(x_i - m)^2,$$

In this formula, N is the number of games played which is 100. m is the average score and Xi is an actual score for game i.

Another useful measure of which team played better is looking at how much time the ball spent on each of the horizontal thirds of the field. Each team tries to keep the ball out of their third of the field where the enemy can score and in their opponent's third where it is dangerous to the enemy team. When the ball spends a lot of time on one team's third of the field, it means the other team is attacking a lot and the team on the defensive is unable to move the ball away from their own goal. While looking at where the ball spent its time is not as important as the score, it does show how well each team did in pushing the ball toward their enemy's goal. By pushing the ball toward the enemy third where a team can score, it also demonstrates its ability to keep the other team from scoring. The time the ball spends in the central third of the field is not important and is ignored because one cannot assume either side has the advantage when the ball is in that area.

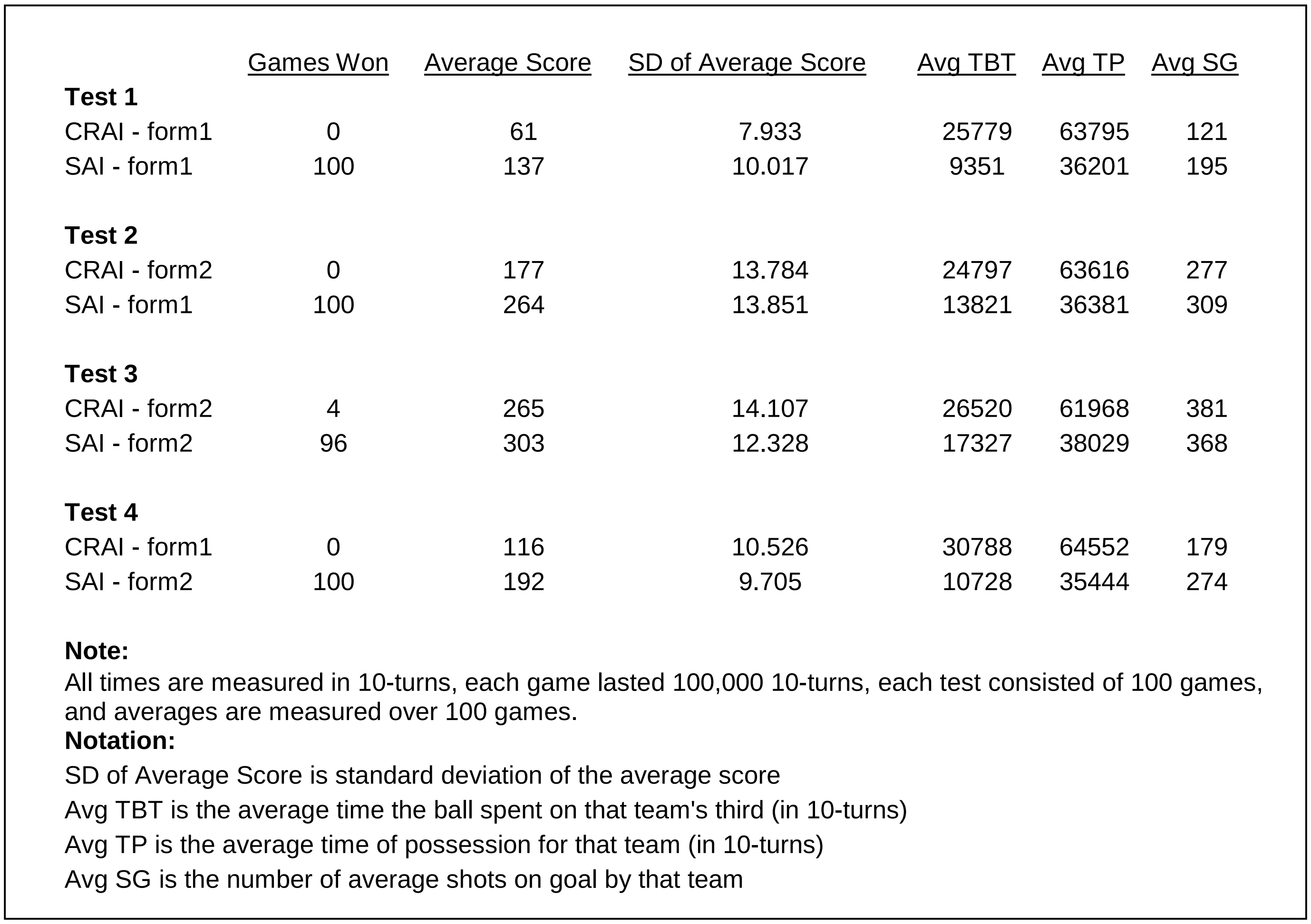
The simulation also keeps track of shots made on goal and time of possession for each team. However, I have found that these factors are harder to interpret in terms of how well a team is doing. In real soccer, a team wants to have the ball as much as possible and to shoot on goal as much as it can. However, experiments have shown that for my soccer model long times of possession are actually synonymous with losing. For example, an agent team may dominate in time of possession but only because its defenders cannot move the ball up; they may end up passing it to each other a lot (on their own third of the field) instead of attacking the enemy goal. These defenders pass to each other because they are pressured by the other team and find it hard to pass the ball to forwards who are closer to the enemy goal. Also, because of the model being used, shots on goal are not as important as they are in real soccer because here it is very common for these shots to be blocked. Also, some agents may take bad shots that have a high chance of being blocked, so that more shots may mean lots of attacking but of low quality.

## Tests and Results

As discussed in the CRAI section, the CRAI architecture supports 2 different formations. Formation1 has 2 Forwards and 3 Defenders, while Formation2 has 3 Defenders and 2 Forwards. The SAI architecture can support a variable number of Forwards and Defenders. For testing, I used two kinds of SAI teams: Formation1 with 2 Forwards and 3 Defenders and Formation2 with 3 Forwards and 2 Defenders (so both formation schemes for the AIs correspond in terms of the number of Forwards and Defenders). Note that the CRAI formations also specify a vertical position for their players (as discussed earlier), while the Swarm players have no vertical restrictions. I conducted 4 different tests comparing each of the 2 formations of CRAI with each of the 2 formations of SAI. The results of these tests are listed in the table of Figure 14.

In all 4 tests, the SAI team won all or almost all of the games (see Test 3). In Test1, the average SAI score is **2.25** times better than CRAI's average score. In Test2, the average SAI score is **1.49** times better than CRAI's average score. In Test3, the average SAI score is **1.14** times better than CRAI's average score. Finally, in Test4, the average SAI score is **1.66** times better than the CRAI average score. Clearly, all tests show that the SAI team did better than the CRAI team, scoring more goals on average. This result is valid because the standard deviations for the average scores in most of the tests are relatively low in comparison to the difference averages. The one exception is Test 3 where it is clear that some of the games were close and CRAI even won 4 of them. Nevertheless, the SAI team in Test 3 still performed better overall.

In addition, we can conclude that the formations of the teams have an effect on how much better the SAI team is than the CRAI team. When both teams played with 3 Defenders, the SAI team won with the greatest margin of victory. On the other hand, when both teams played with 3 Forwards the SAI team still won but with the lowest margin of victory. Test 4 shows that when SAI has 3 Forwards and CRAI has 2, the result for the SAI team is worse than when both teams had 2 Forwards. Test 2 shows that when CRAI has 3 Forwards and SAI has 2 Forwards, the result for the CRAI team is better than when both teams had 2 Forwards. Therefore, the CRAI team plays better with 3 Forwards and 2 Defenders while the SAI team plays better with 2 Forwards and 3 Defenders.



| | Games Won | Average Score | SD of Average Score | Avg TBT | Avg TP | Avg SG |
|---|---|---|---|---|---|---|
| **Test 1** | | | | | | |
| CRAI - form1 | 0 | 61 | 7.933 | 25779 | 63795 | 121 |
| SAI - form1 | 100 | 137 | 10.017 | 9351 | 36201 | 195 |
| **Test 2** | | | | | | |
| CRAI - form2 | 0 | 177 | 13.784 | 24797 | 63616 | 277 |
| SAI - form1 | 100 | 264 | 13.851 | 13821 | 36381 | 309 |
| **Test 3** | | | | | | |
| CRAI - form2 | 4 | 265 | 14.107 | 26520 | 61968 | 381 |
| SAI - form2 | 96 | 303 | 12.328 | 17327 | 38029 | 368 |
| **Test 4** | | | | | | |
| CRAI - form1 | 0 | 116 | 10.526 | 30788 | 64552 | 179 |
| SAI - form2 | 100 | 192 | 9.705 | 10728 | 35444 | 274 |

**Note:**
All times are measured in 10-turns, each game lasted 100,000 10-turns, each test consisted of 100 games, and averages are measured over 100 games.
**Notation:**
SD of Average Score is standard deviation of the average score
Avg TBT is the average time the ball spent on that team's third (in 10-turns)
Avg TP is the average time of possession for that team (in 10-turns)
Avg SG is the number of average shots on goal by that team

Figure 14: Experiment Results

The average times the ball spent on the respective thirds of the field of both teams also supports the notion that SAI is the better soccer architecture. In each test, the ball spent significantly more time on CRAI's third of the field where the SAI team was more likely to score. Here, the best result for the CRAI team was Test3 where CRAI's time was **1.53** times greater, and thus, worse than SAI's. Therefore, in the best case for CRAI, SAI's team was 1.53 times better in terms of keeping the ball on the other team's third of the field.

The CRAI team dominates the time of possession of the ball in each test. While this would be a positive state in real soccer, CRAI team ends up losing despite the fact that it controls the ball about 2 times as much as SAI. Finally, in Tests 1 and 4, SAI takes more shots on the enemy goal. However, in the other 2 tests both teams take a similar number of shots where neither really has the advantage in this category. In Test3 CRAI actually takes more shots, but in the end, fewer of its shots actually become goals because SAI has the higher average score for that test.

# X Conclusion

The experiment outlined in this paper shows that Swarm AI is a valid strategy for controlling soccer agents who are trying to perform certain high-level tasks (i.e. defensive coverage, passing, and offensive movement without the ball). We can conclude that SAI works because it performs better than a centralized rule-based system. Nevertheless, many other experiments are necessary to determine exactly how well Swarm AI can control a team of soccer players.

In this experiment, the SAI architecture relied heavily on the Foundation Architecture for low level and other behaviors. Since CRAI and SAI shared the same foundation, this could have resulted in more similar performance during the games than we would expect with two completely different architectures. Furthermore, since SAI made use of the FA procedures, we cannot say that the SAI architecture is completely swarm-based since a large portion of it has nothing to do with Swarm Intelligence. In fact, one would be accurate if he called the SAI architecture demonstrated here a hybrid of Swarm AI and Ad Hoc rules and algorithms. In the future, it would be a good idea to design a SAI system from the ground up that has nothing in common with the AIs it is being compared to. Originally, this was my intent, but time constrains prevented me from carrying out this objective. Hopefully, a new SAI soccer architecture will be designed to be completely swarm-based, which would shed further light on the effectiveness of Swarm Intelligence in the soccer game domain because this AI would handle not only the 3 soccer sub-problems discussed in this project but all aspects of the game.

While the CRAI approach is similar in principle to the leading soccer architectures of today, it is not nearly as complex. It does not provide for planning, learning, and other advanced AI techniques employed by soccer researchers. Also, there are various ways to improve CRAI without those advanced techniques by implementing additional rules and more sophisticated algorithms. Consequently, one cannot claim that because SAI is better than CRAI in their current states, Swarm AI will be able to play as well as the best soccer AIs of the day. However, the data shows that a simple Swarm system is more effective than a simple centralized rule-based system. This means that it is quite likely that a more sophisticated Swarm approach will be as good as or better than more advanced non-Swarm approaches. Thus, this experiment suggests that further research in Swarm AI and soccer may possibly yield a powerful new approach to controlling soccer agents.

Finally, I have shown that Swarm AI can be used effectively in a dynamic real-time environment that is not like any other where Swarm Intelligence has been used before. This implies that there may be other domains where a Swarm solution can provide good results. Those domains may not resemble the typical environments where Swarm AI has been tried before, but that should no longer be considered an excuse for not trying a Swarm approach. If Swarm behavior can be used to play soccer, there may be countless other problems Swarm Intelligence can solve. In the end, I believe Swarm AI is a very promising new tool and future work should focus on experimenting with it in new domains to see if the current best results can be improved upon.

## XI References

1. Bullnheimer, Bernd, Hartl, Richard, and Strauss, Christine. “Applying the Ant System to the Vehicle Routing Problem.” 2nd Metaheuristics International Conference (MIC-97), SophiaAntipolis, France, 1997.

2. Drogoul, Alexis. "When Ants Play Chess (Or Can Strategies Emerge from Tactical Behaviors?)." In C.Castelfranchi and J.-P.Mller , From Reaction to Cognition: Selected Papers, Fifth European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW '93), 1327.

3. Stone, Peter, Riley, Patrick, and Veloso, Manuela. “The CMUnited-99 Champion Simulator Team.” 2000. RoboCup-99: Robot Soccer World Cup III. Springer Verlag, Berlin.

4. Tambe, M., Adibi, J., Alonaizon, Y., Erdem, A., Kaminka, G., Marsella, S., Muslea, I., and Tallis, M. “ISIS: Using an explicit model of teamwork in RoboCup'97.” 1998. RoboCup'97: Proceedings of the first robot world cup competition and conferences. Springer Verlag, Berlin.