

**COLLABORATIVE AND ADVERSARIAL  
LEARNING FOR SIMULATED ROBOCUP SOCCER**

*Dissertation submitted to Jawaharlal Nehru University, in partial  
fulfillment of the requirement for the award of the Degree of*

**MASTER OF TECHNOLOGY**  
In  
**COMPUTER SCIENCE AND TECHNOLOGY**

By  
**MADHU KUMARI**

Under the Guidance of  
**PROF. K. K. BHARADWAJ**

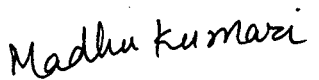


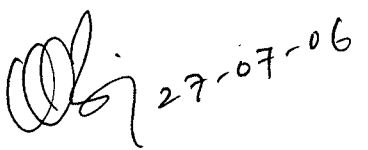
**SCHOOL OF COMPUTER & SYSTEMS SCIENCES  
JAWAHARLAL NEHRU UNIVERSITY  
NEW DELHI - 110067  
JULY 2006**


# CERTIFICATE

This is to certify that the dissertation entitled “**Collaborative and Adversarial Learning for Simulated RoboCup Soccer**”, being submitted by **Ms. Madhu Kumari** to the School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, in partial fulfillment of the requirement for the award of the Degree of **Master of Technology in Computer Science and Technology**, is a bona fide work carried out by him under the guidance and supervision of **Prof. K. K. Bharadwaj**.

The matter embodied in the dissertation has not been submitted for the award of any other Degree or Diploma.

  
Madhu Kumari  
(Student)

  
Prof. K. K. Bharadwaj  
(Supervisor)

  
Prof. S. Balasundaram  
Dean, SC&SS  
Jawaharlal Nehru University  
New Delhi-67  
India.

## **ACKNOWLEDGEMENT**

I would like to thank many people for their support, encouragement and guidance during my years as an M.Tech student here at JNU. First and foremost, this dissertation represents a great deal of time and effort not only on my part, but on the part of my supervisor, **Professor K.K. Bharadwaj**. He has helped me to shape my research from day one and encouraged me to achieve to the best of my ability. He provided a good mixture of guidance and non-interference, allowing me to explore and develop my own ideas while offering helpful suggestions when they were most needed.

I express my thanks to the Dean, Prof. S. Balasundaram for providing the necessary infrastructure to carry out this dissertation. I like to mention my sincere thanks to all of my friends for their valuable remarks and appreciations.

Finally, I express my gratitude for, my parents, sisters, brother, and grandmother who believed in me, offered guidance, and supported my decisions especially during my tough time.

**MADHU KUMARI**

# ABSTRACT

The RoboCup Federation offers a challenging and complex test bed for all kinds of scientific methods by organizing international competitions in robotic soccer. RoboCup 2D-Simulation League is one of the important competitions. It is a complex multi-agent domain which is real-time, noisy, collaborative and adversarial. Research in this domain covers a gamut of areas as networking, AI and machine learning etc., but it will remain main problem for AI at least for fifty years.

Learning defensive tactics at teamwork level is an example of collaborative and adversarial learning. This work presents a novel idea for the improvement in defensive tactics of an agent in Simulated RoboCup Soccer by using Extended Shaped Reinforcement learning under layered learning paradigm. The major contributions of this dissertation are to extend Shaped Reinforcement learning to make it workable for collaborative and adversarial domains like soccer and hence to speed up the learning process of an agent by reducing the state space to a drastically small level and by providing an efficient and implicit action selection mechanism at teamwork level.

# CONTENTS

	<b>Page</b>
<b>1. INTRODUCTION</b>	<b>1</b>
1.1 Overview of the Problem Domain	1
1.2 RoboCup Soccer	2
1.2.1 Simulation league	3
1.2.2 Small-size real robot league	3
1.2.3 Middle-size real robot league	4
1.3 RoboCup challenge in simulation	4
1.3.1 Teamwork challenge	5
1.3.2 Opponent modeling challenge	5
1.3.3 Learning challenge	5
1.4 Motivation	6
1.5 Related work	6
1.6 Outline of the Thesis	7
<b>2. Underlying Approaches</b>	<b>8</b>
2.1 Shaped Reinforcement Learning	8
2.1.1 Managing State Space Complexities	9
2.1.2 Shaping Reinforcement	9
2.2 Layered Learning	11
2.2.1 Principles	11
2.2.2 Layered Learning Formalism	13
2.2.3 Application Layered Learning to Simulated Robotic Soccer	14
<b>3. Improvement of Defensive Tactics in Simulated RoboCup Soccer</b>	<b>18</b>
3.1 Extended Shaped Reinforcement Learning	19
3.1.1 Assumptions	19
3.2 Behaviors	20
3.2.1 Stay Compact	20

3.2.2 Support1	21
3.2.3 Blocking the Lanes of Shooting	23
3.2.4 Build Up Play	26
3.2.5 Delay	31
3.2.6 Pressurize	33
3.2.7 Safe Play	34
3.2.8 Support2	35
3.3 Conditions	36
3.4 Heterogeneous Reward Function	39
3.5 Progress Estimators	40
3.5.1 Individual Performance Estimators	41
3.5.2 Progress Estimators for Main Behavior	46
<b>4. Benefits of Extended Shaped Reinforcement Learning</b>	<b>49</b>
4.1 Reduction of State Space	49
4.2 Speed up in Learning	49
4.3 Efficient action selection mechanism	50
4.4 Results	50
<b>5. Conclusion and Future work</b>	<b>54</b>
<b>Appendix 1</b>	<b>55</b>
<b>Appendix 2</b>	<b>59</b>
<b>REFERENCES</b>	<b>65</b>

# Chapter 1

## Introduction

Computing has evolved from the age of computing machines (1950-1960) to intelligent autonomous robots through various breakthroughs [Minsky 87]. Hence Distributed Artificial Intelligence (DAI) attracted special attention in the last two decades. Broadly it encompasses two main categories, Distributed Problem Solving and Multi-Agent Systems (MAS).

### 1.1 Overview of the Problem Domain

The modern approach to artificial intelligence (AI) is centered around the concept of a rational agent. An agent is anything that can perceive its environment through sensors and act upon that environment through actuators. An agent that always tries to optimize an appropriate performance measure is called a rational agent. Such a definition of a rational agent is fairly general and can include human agents (having eyes as sensors, hands as actuators), robotic agents (having cameras as sensors, wheels as actuators), or software agents (having a graphical user interface as sensor and as actuator). From this perspective, AI can be regarded as the study of the principles and design of artificial rational agents.

However, agents are seldom stand-alone systems. In many situations they coexist and interact with other agents in several different ways. Examples include software agents on the Internet, soccer playing robots. Such a system that consists of a group of agents that can potentially interact with each other is called a multi-agent system (MAS). Although according to the strong notion of agency, multi-robots are considered special case of multi-agent systems [Dudek et. al 96]. But there are inherent differences between these two systems [Farinelli et. al 2003]. Certainly there are large number of domains in which both the areas seems to be equivalent [Farinelli et. al 2003]. But research issues related to multi-robots should be enquired in the context of multi-agent systems as well as in the

light of inherently different robotics problems. An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment or an agent is a computer based system that is capable of doing independent action on behalf of its user [Wooldridge 2002].

The study of multi-robot systems (MRS) has received increased attention since the mid-1990's. This is not surprising as continually improving technology and infrastructure have made the deployment of MRS consisting of increasingly larger numbers of robots possible. With the growing interest in MRS comes the expectation that, at least in some important respects, multiple robots will be superior to a single robot in achieving a given task. The real world problems that are ideal for robotic solutions are very complex and challenging and most of them are centered on mainly three characteristics of the domains as, these domains are real-time, noisy and collaborative. RoboCup Soccer is the only problem domain of MRS which is adversarial (Other robots in the environment that have goals opposed to the team's long-term goal are the team's adversaries). It offers the rich test bed for almost every aspect of artificial intelligence.

## **1.2 RoboCup Soccer**

RoboCup (The Robot World Cup Initiative) is an attempt to promote intelligent robotics research by providing a common task for evaluation of various theories, algorithms, and agent architectures. RoboCup has currently chosen soccer as its standard task. In order for a robot (a physical robot or a software agent) to play a soccer game reasonably well, many technologies need to be integrated and a number of technical breakthroughs must be accomplished. The range of technologies spans the gamut of intelligent robotics research, including design principles for autonomous agents, multi-agent collaboration, strategy acquisition, real-time reasoning and planning, robot learning, and sensor fusion.

It's obvious that building a robot to play a soccer game is an immense challenge. The First Robot World Cup Soccer Games and Conferences (RoboCup-97) was held during the International Joint Conference on Artificial Intelligence (IJCAI-97) at Nagoya, Japan with 37 teams around the world, and the Second Robot World Cup Soccer Games and



Conferences (RoboCup-98) was held on July 2–9, 1998 at La Cite des Sciences et de l'Industrie (La Cite) in Paris with 61 teams. RoboCup-99 Stockholm will be held in conjunction with IJCAI-99 participated in by over 120 teams The idea of using soccer for robotics research is not new. In 1993, Alan Mackworth proposed in a paper titled “On Seeing Robots” [Mackworth 93] that soccer can be a good tested of robotics and AI research. Independently, several researchers have been working on the soccer domain. These efforts merged into RoboCup. A unique feature of RoboCup is that it is a systematic attempt to promote research using a common domain, mainly soccer. RoboCup is designed to require the handling of real-world complexities, though in a limited world, while maintaining an affordable problem size and research cost. RoboCup offers an integrated research task covering broad areas of intelligent robotics. Such areas include: real-time sensor fusion, reactive behavior, strategy acquisition, learning, real-time planning, multi-agent systems, context recognition, vision, strategic decision-making, motor control, intelligent robot control, and many more.

Currently, RoboCup consists of three competition tracks [Asada et al. 99] :

### **1.2.1 Simulation league**

Each team consists of eleven programs, each controlling separately each of the eleven team members. Each player has distributed sensing capabilities (vision and auditory) and motion energy both of which are resource bounded. Communication is available between players and strict rules of the soccer game are enforced (e.g., offsides). This league is mainly for researchers who may not have the resources for building real robots, but are highly interested in complex multi-agent reasoning and learning issues.

### **1.2.2 Small-size real robot league**

The field is of the size and colors of a ping-pong table and up to five robots per team play a match with an orange golf ball. The robot size is limited to approximately 15 cm<sup>3</sup>. Typically robots are built by the participating teams and move at speeds of up to 2m/s. Global vision is allowed, offering the challenge of real-time vision-based tracking of five fast moving robots in each team and the ball.

### **1.2.3 Middle-size real robot league**

The field size is of the size and color of three by three ping-pong tables and up to five robots per team play a match with a Futsal-4 ball. The size of the base of the robot is limited to approximately 50 cm diameter. Global vision is not allowed. Goals are colored and the field is surrounded by walls to allow for possible distributed localization through robot sensing.

One of the major reasons why RoboCup attracts so many researchers is that it requires the integration of a broad range of technologies into a team of complete agents, as opposed to a task-specific functional module. [Asada 98] gives a partial list of research areas which RoboCup covers:

- Agent architecture in general;
- Combining reactive approaches and modeling/planning approaches;
- Real-time recognition, planning, and reasoning;
- Reasoning and action in a dynamic environment;
- Sensor fusion;
- Multi-agent systems in general;
- Behavior learning for complex tasks;
- Strategy acquisition;
- Cognitive modeling in general.

### **1.3 RoboCup challenge in simulation**

RoboCup offers significant long term challenges which will take few decades to meet. The fundamental issue for researchers who wish to build a team of RoboCup is to design a multi-agent system that behaves in real time, performing reasonable goal-directed behavior. Goals and situation change dynamically and in real time. The challenges for simulated league must include the following issues - Machine learning in a multi-agent system, collaborative and adversarial environment, Multi-agent architectures enabling

real time multi-agent planning and plan execution in service of team work and opponent modeling.

[Kitano et al. 98] proposed three main challenges in simulated RoboCup leagues:

- Teamwork challenge.
- Opponent modeling challenge
- Learning challenge.

### **1.3.1 Teamwork challenge**

The RoboCup teamwork challenge addresses the issues of, real time planning, re-planning and execution of multi-agent team work in a dynamic and adversarial environment

### **1.3.2 Opponent modeling challenge**

It includes the research issues like modeling and reasoning about other agents goal, plans, Knowledge, capabilities, or emotions. The modeling issue in RoboCup can be broken in to three parts as on-line tracking, on-line strategy recognition and off-line review.

### **1.3.3 Learning challenge**

The objectives of RoboCup learning challenge is to solicit comprehensive learning schemes that is applicable to multi-agent learning systems which need to adapt to the situation and to evaluates merits and demerits of proposed approaches using standard tasks. It includes the research issues associated with learning aspects of RoboCup as:

- Off-line skill learning by individual agent.
- Off-line collaborative learning by teams of agents.
- On-line skill and collaborative learning.
- On-line adversarial learning.

Among all the above written challenges learning is most challenging issue because of the fact that learning has to be embodied into team and agents architecture and opponent models under the constraints of time and resources.

## **1.4 Motivation**

On line learning has been a challenging issue in Simulated RoboCup soccer especially at the teamwork level. There are number of learning tasks which involves adversarial and collaborative learning e.g. pass selection, improving defensive and offensive tactics. This work is inspired from [Mataric 97] and [Stone and Veloso 98]. This work is an effort to combine the advantages of both the approaches and make it workable to improve defensive tactics.

## **1.5 Related work**

Several people have studied collaboration among agents (commitment, co-evolution, etc.) and several people have studied adversarial multi-agent situations (game theory, Markov games, etc.). Yet there has been little effort towards studying situations in which agents reason about collaborating with other benevolent agents while at the same time trying to outwit one or more opponents. Collaborative environments have been examined in the field of MAS [Grosz 96, Sycara et al. 96, Cohen et al. 99], and adversarial domains have been considered in AI game playing systems such as for checkers [Samuel 59] and chess [Newell and Simon 72]. However these adversarial domains are turn-taking as opposed to real-time, they are not noisy, and there are no collaborative agents. Littman uses Markov games to learn stochastic policies in a very abstract version of 1-on-1 robotic soccer [Littman 94]. There have also been a number of studies of multi-agent reinforcement learning in the pursuit domain with four predators chasing a single prey in a small grid-like world. For example, [Tan 93] compares situations in which predator agents are allowed to share reinforcement information and/or policies, [Arai 97] provides agents with reinforcement for enabling successful actions by teammates and [Ono 97] equips each predator agent with different behavior modules based on how many teammates are closer than it is to the prey. Even the relatively complex backgammon [Tesauro 94] and elevator control [Crites and Barto 96] domains have much smaller state space than the

simulated robotic soccer domain. In another predator-like task, Zhao and [Schmidhuber 96] use a single run to deal with the opponents' shifting policies and ignore the opponents' policies just as we do. The effects of opponent actions are captured in the reward function. In robotic soccer, a reinforcement learning approach has been used for strategic positioning [Andou 98] in the soccer server. Introducing observational reinforcement learning, this system allows players to notice where the ball has traveled most often in the past and to adjust their positions such that they are closer to the ball's path in the future.

## **1.6 Outline of the Thesis**

Chapter 2 Introduces the learning approaches and ideas and gives the basics of shaped reinforcement learning and layer learning concepts.

Chapter 3 describes the extended shaped reinforcement learning technique along with details the complete behavior specifications, conditions, heterogeneous reward functions and progress estimators.

Chapter 4 Provides benefits of the proposed work and results.

Chapter 5 presents conclusion of the dissertation and future work.

## Chapter 2

### Underlying Approaches

Reinforcement learning offers a powerful set of techniques that allow a robot to learn a task without requiring its designer to fully specify how it should be carried out. If the task is feasible and feedback regarding how well the agent is doing is provided, several reinforcement learning techniques are guaranteed to converge to the optimal solution. The guarantees are tempered by rather strong conditions for convergence. Q-learning for example, requires all actions to be repeatedly sampled in all states.

In the last decades the suitability of reinforcement learning in variety of domains including robotics has become a methodology for learning. But Traditional reinforcement learning results in poor performance [Mataric 97] in dynamic situated multi-agent domains characterized by multiple goals, noisy perception and action, and inconsistent reinforcement. In highly dynamic and noisy environment where agents can not sense the world's state or even their own states completely traditional reinforcement learning can not work, hence some modifications and enhancements should be done in traditional reinforcement learning. Next two sections describe such learning techniques.

#### 2.1 Shaped Reinforcement Learning

[Mataric 97] gives, one of the fundamental causes of inapplicability of theoretical reinforcement learning paradigms to real time robotic systems lies in the assumption that agent-environment interaction can be modeled as a Markov Decision Process (MDP). It puts the main problems with traditional reinforcement learning in to two broad categories as:

- Managing complexity in size of the state space.
- Dealing with structuring and assigning reinforcement.

[Mataric 97] has given an intelligent way of dealing with the above written problems of reinforcement learning in situated domains, by reducing state space through the use of

behaviors and their associated conditions, and shapes reinforcement with heterogeneous reward function and progress estimators.

### **2.1.1 Managing State Space Complexities**

The environment of an agent is partially observable through its noisy sensors, which provide the input state as a combination of discrete and continuous sensory inputs, this makes learning space exponentially high. Situation becomes worse in the presence of other agents. Consequently some form of input generalization or state clustering becomes necessary for most non-trivially sized learning problems. Moreover state transitions are largely externally induced and asynchronous, and their causes can not always be sensed, and for agents action selection becomes a more difficult task. This work describes a method of state clustering through the use of behaviors and conditions, which allow agents to learn policies without assuming any world model by the inclusion of implicit domain knowledge in to various reinforcements.

#### **Conditions**

Conditions are the predicates on sensor readings that map into a proper subset of the state space. Each condition is defined as the part of the state that is necessary and sufficient for activating a particular behavior. The truth value of a condition determines when a behavior can be executed and when it should be terminated and provides a set of events for the learner's control algorithm.

#### **Behaviors**

Behaviors are goal driven control laws that achieve and/or maintain particular goals to abstract away the low level details of control. Well-designed behaviors utilize the dynamics of the system and its interaction with the world in order to achieve robust performance.

### **2.1.2 Shaping Reinforcement**

A multi-agent environment does not provide a direct source of reinforcement because of the fact that other agent can also affect the environment, this makes assignment of credits

or punishments extremely difficult and such complex environment produces a confounding credit assignment problem.

Shaping principle is based on principled embedding of domain knowledge in order to convert intermittent feedback into more continuous error signal using two types of reinforcement- heterogeneous reward functions and progress estimators. It is obvious that the more subgoals the system recognizes, the more frequently reinforcement can be applied, and the faster the learner can converge. In this system each behavior provides a goal whose achievement can be detected as an event, and can also be directly translated into reinforcement signal.

### **Heterogeneous Reward Functions**

These functions combine multi-modal feedback from the available external (sensory) and internal (states) modalities. The combination is a weighted sum of inputs from the individual event-driven functions. These reward functions, much like most typically used in reinforcement learning, deliver reinforcement in response to events, i.e. between behaviors.

### **Progress Estimators**

These are evaluation metrics relative to a current goal that the robot can estimate during the execution of a behavior. Progress estimators diminishes the brittleness of the learning algorithm by decreasing the sensitivity to noise by strengthening appropriate condition behavior correlations, by encouraging exploration by using lack of progress to terminate behaviors principally and by decreasing the fortuitous rewards.

A shaped learning algorithm learns and maintains value function that maps conditions  $c$  to behaviors  $b$  based on its appropriateness. This appropriateness is computed by the total reinforcement behavior gets over a period of time, and this total reinforcement at any point is the weighted sum of heterogeneous reward function and progress estimators. In shaped learning algorithm agent learns a matrix  $A(c, b)$  through value function to select appropriate action in the given environmental condition. The values in the matrix fluctuate over time based on received reinforcement.



In the following equations  $R_H(c, t)$ ,  $R_i(c, t)$  is heterogeneous reward function and progress estimators respectively.

$$A(c, b) = \sum_{t=1}^T R(c, t),$$

$$R(c, t) = uR_H(c, t) + \sum_{\forall i} w_i R_i(c, t)$$

Where  $u, w_i > 0$  and  $u + \sum_{\forall i} w_i = 1$

## 2.2 Layered Learning

Layered learning is a bottom up hierarchical learning approach to the agent behaviors that allows machine learning at various levels and it is applied to the tasks for which learning a direct mapping from input to the output is intractable with the existed learning methods. [Stone and Veloso 98a] gives a framework of layered learning. Layered learning is designed for the domains that are too complex for learning a mapping directly input to the output representation. Instead the layered learning approach consists of breaking of a problem down into several task layers. At each layer, a concept needs to be acquired. A machine learning algorithm abstracts and solves local learning tasks

### 2.2.1 Principles

Layered learning is defined by four principles. In this section, I identify, motivate, and specify these four principles.

#### Principle 1

Motivated by robotic soccer, layered learning is designed for domains that are too complex for learning a mapping directly from an agent's sensory inputs to its actuator outputs. We assume that any domain that fits in the following criteria limited communication, real-time, noisy environments with both teammates and adversaries is too complex for agents to learn direct mappings from their sensors to actuators.

Instead, the layered learning approach consists of breaking a problem down into several behavioral layers and using machine learning (ML) techniques at each level. Layered learning uses a bottom-up incremental approach to hierarchical task decomposition. Starting with low-level behaviors, the process of creating new ML subtasks continues until reaching high-level strategic behaviors that deal with the full domain complexity.

### **Principle 2**

The appropriate behavior granularity and the aspects of the behaviors to be learned are determined as a function of the specific domain. The task decomposition in layered learning is not automated. Instead, the layers are defined by the ML opportunities in the domain. Layered learning can, however, be combined with any algorithm for learning abstraction levels. In particular, let A be an algorithm for learning task decompositions within a domain. Suppose that A does not have an objective metric for comparing different decompositions. Applying layered learning on the task decomposition and quantifying the resulting performance can be used as a measure of the utility of A's output.

### **Principle 3**

Machine learning is used as a central part of layered learning to exploit data in order to train and/or adapt the overall system. ML is useful for training behaviors that are difficult to fine-tune manually. It is useful for adaptation when the task details are not completely known in advance or when they may change dynamically. In the former case, learning can be done off-line and frozen during actual task execution. In the latter, on-line learning is necessary since the agent needs to adapt to unexpected situations, it must be able to alter its behavior even while executing its task. Like the task decomposition itself, the choice of machine learning method depends on the subtask.

### **Principle 4**

The key defining characteristic of layered learning is that each learned layer directly affects the learning at the next layer. A learned subtask can affect the subsequent layer either (i) by providing a portion of the behavior used during training or (ii) by creating

the input representation of the learning algorithm. In general, machine learning algorithms - e.g. neural networks, Q-learning [Watkins 89], and decision trees [Quinlan 93] require an input and output representation, a target mapping from inputs to outputs, and training examples. The goal of learning is to generalize the target mapping from the training examples which provide the correct outputs for only a portion of the input space.

In summary, layered learning is a machine learning paradigm designed to allow agents to learn to accomplish a goal in a complex environment. Layered learning allows for a bottom up definition of agent capabilities at different levels in a complex domain. Machine learning opportunities are identified when data is available or the task is unpredictable and hand coded solutions are too complex to generate. Individual learned behaviors are organized, learned, and combined in a layered fashion, each facilitating the creation of the next.

### 2.2.2 Layered Learning Formalism

Consider the learning task of identifying a hypothesis  $h$  from among a class of hypothesis  $H$  which maps a set of state feature variables  $S$  to a set of output  $O$  such that based on the set of training example,  $h$  is most likely (of the hypothesis  $H$ ) to represent un seen examples.

When using the layered learning paradigm, the complete learning task is decomposed into hierarchical subtask layers  $\{L_1, L_2, \dots, L_n\}$  with each layer defined as

$$L_i = (\bar{F}_i, O_i, T_i, M_i, h_i)$$

Where

$\bar{F}_i$  is the input vector of state features relevant for learning for subtask  $L_i$ .

$$\bar{F}_i = \langle F_i^1, F_i^2, \dots \rangle. \forall j, F_i^j \in S$$

$O_i$  is the set of outputs from among which to choose for subtask  $L_i$ .  $O_n = O$

$T_i$  is the set of training examples used for learning subtask  $L_i$ . Each of  $T_i$  consists of

Correspondence between an input feature vector  $\vec{f} \in \vec{F}_i$  and  $o \in O_i$ .

$M_i$  is the machine learning algorithm and used at layer  $L_i$  to select a hypothesis mapping

$\vec{F}_i \mapsto O_i$  based on  $T_i$ .

$h_i$  result of running  $M_i$  on  $T_i$ .  $h_i$  is a function from  $\vec{F}_i$  to  $O_i$ .

As set out in **Principle 2** of layered learning the definitions of the layers  $L_i$  are given a priori. **Principle 4** addressed via the following stipulation.  $\forall i < n$ ,  $h_i$  directly affects  $L_{i+1}$  at least one of the three ways:

1.  $h_i$  is used to construct one or more features  $F_{i+1}^k$ .
2.  $h_i$  is used to construct elements of  $T_i$ ; and/or
3.  $h_i$  is used to prune the output set  $O_i$ .

### 2.2.3 Application Layered Learning to Simulated Robotic Soccer

One tempting way to approach any new agent-based domain is to try to learn a direct mapping from the agent's sensors to its actuators. However, a quick consideration of the robotic soccer domain is enough to convince oneself that it is too complex for such an approach: the space of possible sensory inputs is huge, there are many possible actions, and there is a large amount of hidden state. Such complexity is an important characteristic of the domain for the purposes of this thesis, since robotic soccer is meant to represent other domains which are too complex for the straightforward approach. [Stone and Veloso 98a] give implementation of the various layers of agents learning in simulated robotic soccer.

Fig. 1.1 A sample task decomposition within the layered learning framework in a collaborative and adversarial multi-agent domain. Layered learning is designed for use in domains that are too complex to learn a mapping straight from sensors to actuators. We use a hierarchical, bottom-up approach

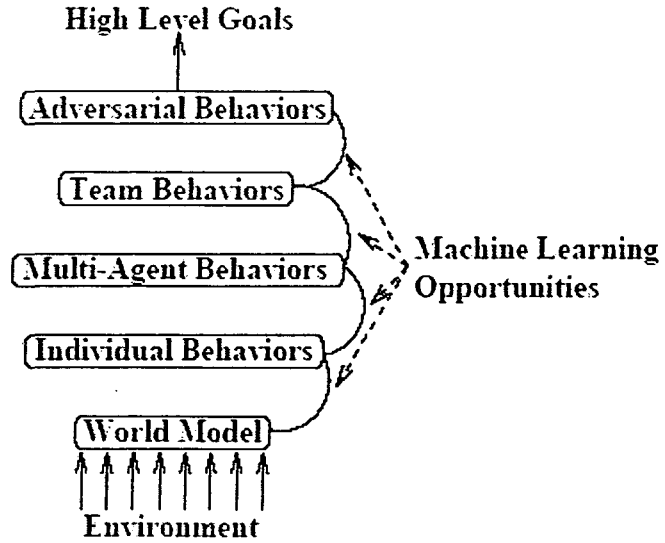


Fig. 1.1 A sample task decomposition within the layered learning framework, in a collaborative and adversarial multi-agent domain.

Table 1.1 illustrates a possible set of learned behavior levels within the simulated robotic soccer domain that correspond to the abstract task decomposition represented in Fig. 1.1.

**Table 1.1**

Layer	Strategic level	Behavior type	Example
$L_1$	robot-ball	individual	ball interception
$L_2$	one-to-one player	multi-agent	pass evaluation
$L_3$	one-to-many player	team	pass selection
$L_4$	team -formation	team	strategic positioning
$L_5$	team-to-opponent	adversarial	strategic adaptation

### **Layer1 -Ball Interception – An Individual Skill**

First, the agents learn a low-level individual skill that allows them to control ball effectively. While executed individually, the ability to intercept a moving ball is required

Due to the presence of other agents: it is needed to block or intercept opponent shots or passes as well as to receive passes from teammates. As such it is a prerequisite for most of ball manipulation behaviors.

### **Layer2 Pass Evaluation – A Multi Agent Behavior**

Second, the agents use their learned ball interception skill as a part of behavior for training a multi-agent behavior. When an agent has the ball and has the option to pass to a particular teammate, it is useful to have an idea of whether or not the pass will actually succeed if executed: will the teammate successfully receive the ball such an evaluation depends not only the teammate's and opponent's positions, but also their abilities to receive or intercept the pass. Consequently, when creating training examples for the pass evaluation function, the intended pass recipient is equipped with the previously learned ball interception behavior.

### **Layer3 Pass Selection –A Team Behavior**

Pass selection represents the third and highest-level behavioral layer within layered learning implementation .Pass selection is a behavior that must be adaptable. Since it depends on the behaviors of teammates and opponents, agents must be able to adjust their decisions based on the empirical results of past decisions. Thus, pass selection is an appropriate behavior for learning because of the possibility of exploiting data to adapt to a shifting concept. In the robotic soccer context, [Stone and Veloso 99] used TPOT-RL to learn pass selection, taking advantage of the learned pass-evaluation capability described in Chapter 6 to construct the input representation for learning.

### **TPOT-RL**

TPOT-RL is an effective technique for enabling a team of agents to learn to cooperate towards the achievement of a specific goal. It is an adaptation of traditional RL methods that is applicable in complex, non-stationary, multi-agent domains with large state spaces and limited training opportunities. TPOT-RL enables teams of agents to learn effective policies with very few training examples even in the face of a large state space with large

amounts of hidden state. In short, TPOT-RL applies in domains with the following characteristics:

- There are multiple agents organized in a team.
- There are opaque state transitions.
- There are too many states and/or not enough training examples for traditional RL techniques to work.
- The target concept is non-stationary.
- There is long-range reward available.
- There are action-dependent features available.

## Chapter 3

### Improvement of Defensive Tactics in Simulated RoboCup Soccer

- Defense and offence can be considered as mirror image to each other, which necessitates the need of improvement in defensive tactics for better performance. There are several definitions of defense, some are based on the possession of ball, e.g. the moment a team loses possession, it is in defense, while others are based on location of ball in the field, e.g. if ball is in the defensive half of the team, then it is a defensive play. We have considered defense in this report, as a situation when ball is in the defensive half of a team and it does not have possession of ball. In any teamwork individual skills are the deciding factor in the overall performance of the team, likewise defensive tactics require strong individual skills (interception, passing, dribbling etc.) as a prerequisite. Learning defensive tactics can be considered as one of the good example of collaborative and adversarial learning like pass selection in [Stone and Veloso 98b], because it involves collaboration between the teammate agents to deal with the issues like forcing opponents in one direction along with the knowledge of opponents positions and behaviors.

Learning method used in this work is based on layered learning technique described in chapter 2, in which output of a layer can influence working of a layer above it in three possible ways, it can be a part of input space, it can help to reduce the search space and speed up the learning process. Since defensive tactics are higher level team behaviors so the learning algorithm used here can form a layer or a part of layer which can be above pass evaluation or pass selection. The core algorithm used here is inspired from the Shaped Reinforcement learning technique [Mataric 97], which is one of the intelligent techniques to reduce the search space of complex domains to a manageable level.



## 3.1 Extended Shaped Reinforcement Learning

Reinforcement learning has become an obvious choice for the domains which involves on-line and unsupervised learning but has indicators of progress through the environmental feedback. But it requires some modifications when domain has very large search space, and interaction of environment and learners, and interaction among the learners is limited e.g. Simulated RoboCup Soccer (Appendix 2).

In this work we have tried to embed defensive tactic in a simulated soccer team by using extended form of shaped reinforcement learning described in chapter 2. The extension is based on the fact that shaped reinforcement learning concept is used for the multi-robot's foraging domain [Mataric 97], which hardly has any notion of teamwork. Hence we have focused to extend this learning technique for teamwork (defensive tactics) through progress estimators.

Like the framework of shaped reinforcement learning, in the proposed work there are well defined moves, situations, heterogeneous reward functions, progress estimators and a control algorithm. All of these components are described in the following sections along with their full specifications.

### 3.1.1 Assumptions

The learning algorithm considered here is based on the following assumptions

- Every teammate agent is well equipped with the low level behaviors and skills e.g. passing, clearing, dribbling, kicking etc.
- Every teammate agent has learned the teamwork up to pass evaluation [Stone and Veloso 98b].
- There is no explicit opponent modeling or opponent's acquisition mechanism, however some implicit knowledge (positional aspects) about opponents is used in the algorithm.
- An agent is able to know complete information about its other teammates present in its vicinity through direct or indirect communication.
- Learning environment and terminologies used in this work are taken from the simulation environment provided by RoboCup Soccer Simulator (Appendix-2).

## 3.2 Behaviors

There are six main behaviors and two subordinate behaviors used in this algorithm. The main behaviors broadly cover the defensive tactics like, delaying opponents, pressurizing opponents, blocking the shooting lanes, how to cope up with the situation like dangerous play (safe play), what to do when team gains possession of ball in the defensive half. While the subordinate behaviors include two main key ideas of defense, how to support a teammate who is doing actual defensive task and how far or near a teammate agent should place itself (staying compact) so that it can provide a good support. Behaviors discriminate among team members on the basis of their distance and direction from the ball (vicinity of the ball). Except the behavior Support2, every behavior considers only the team members present in the vicinity of ball.

### 3.2.1 Stay Compact

It is a subordinate behavior to help the behaviors like delay and pressurizing. This behavior allows defender to be closed enough to support each other and distant enough not to interfere each others working. It considers agents in the vicinity of ball. During the execution of this behavior agent maintains a minimum and maximum distance from every other player in the vicinity and it has a single role for agents.

$d_{ij}$  is the distance of player-  $i$  from another player-  $j$  present in vicinity and  $d_{ij} \in [d_{\min}, d_{\max}]$ , where  $d_{\min}$  is the distance, which an agent can travel in one simulation cycle and  $d_{\max}$  is the average distance an agent can cover in three simulation cycle.

## Algorithm for Stay Compact

$A_i$  is any agent in a team;

IF ( $A_i$  has teammates in its vicinity)

THEN it maintains  $d_{ij}$  from all  $A_j$  teammate present in its vicinity;

### 3.2.2 Support1

It is a subordinate behavior like Stay Compact, to support the players who are doing actual defensive task. This behavior can have a direction of its execution e.g. support FD in forward direction. It has two types of roles for agents, one is to handle opponents and another is to perform covering of attacking space behind the player who is doing actual defensive task. This behavior uses **Pass Evaluation** layer's output in order to predict potential receivers of a pass. It considers the players in the vicinity of ball. The used in this behaviors are as:

#### Context Player

A player is said to be context player if it is tackling, challenging or a First Defender.

#### Important Opponents

An important opponent is the opponent agent which can be potential receiver of a pass.

#### Handling Opponents

An agent locates itself near an opponent so that it can block or intercept or makes reception of pass difficult for this opponent. It can be achieved as, agent moves itself near context player in a given direction (optional in Support1). It turns body and face so that it can view both important opponent and ball holder.

#### Covering

An agent who performs covering task is the nearest player to the goal's centre in the vicinity of ball. It locates itself in the covering area as shown in the Fig. 3.1 and covers



the attacking space left behind the FD. It keeps itself at some distance from FD where more chance of attack or the portion of attacking space left behind the FD.

This behavior can be achieved in the following ways:

- Agent can become Second Defender (SD) or Third Defender (TD).
- Agent can cover the area behind the Context Player.
- Agent can handle important opponents in the vicinity of Context Player.
- Agent can cover and handle both at a time.

It uses the following **low level Skills**:

- Ball Interception
- Passing
- Tracking
- Turning Body

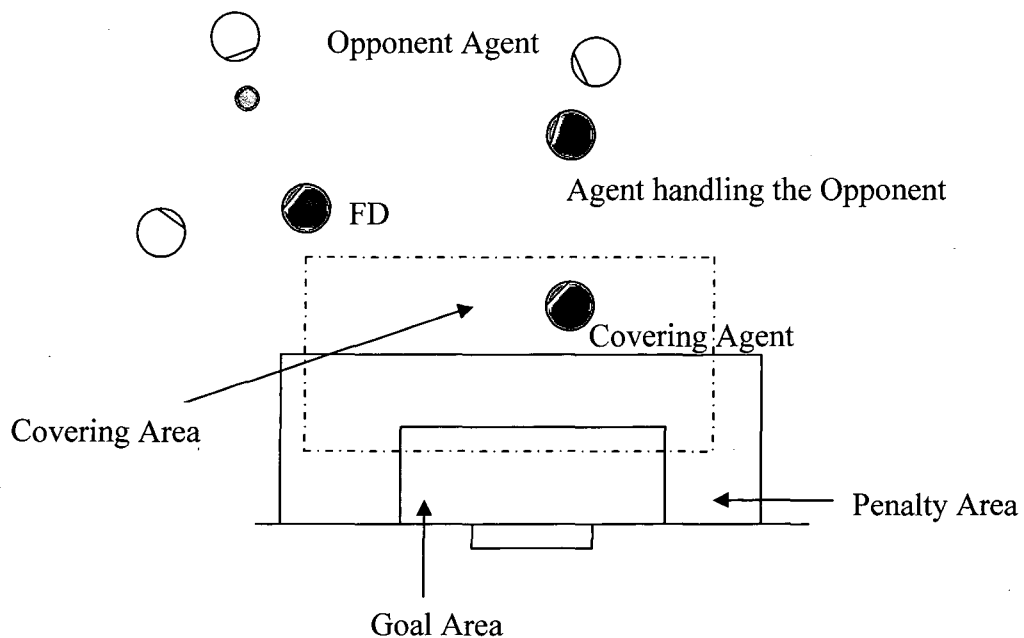


Fig. 3.1 Covering Area in defense

Some times depending on the situation an agent has to perform both the roles handling opponent and covering e.g. when the number of players in the defense is low.

Let  $i, j$  is a context player and supporting players respectively.

### Algorithm for Support1

```
IF (  $d_{ij} \in [d_{min}, d_{max}]$  and agent has an Important Opponent near it)
  THEN Handle the Important Opponent;

ELSE
  { IF (agent is at least distance from goal's centre)
    THEN {
      IF ( agent has an important agent )
        THEN Cover and Handle the Important Opponent;

      ELSE Cover;
    }

  IF (ball comes to the agent OR poor pass OR incorrect pass)
    THEN {
      Get the ball control;
      Make Correct and Useful Pass;
    }
```

### 3.2.3 Blocking the Lanes of Shooting

This move allows an agent to position itself near or in the penalty area so that it can block the shot to the goal, when ball comes to the agent, it must clear or pass it to the teammates present in its vicinity. Normally this move is executed by the agent when it

has least or no support of other teammates and ball movements can turn into the goal. Hence agents have single role in this move (block the chance of shooting towards goal) and are in the vicinity of ball.

It uses the following **low level Skills**:

- Ball Interception
- Clearing
- Passing
- Tracking

In order to achieve this behavior agent must move in the lane of shooting as shown in Fig.3.2. The agent must face towards ball and locate itself at an appropriate distance on Line1 (Fig.3.2) near or in the penalty area depending upon conditions based on ball's distance from the centre of goal.

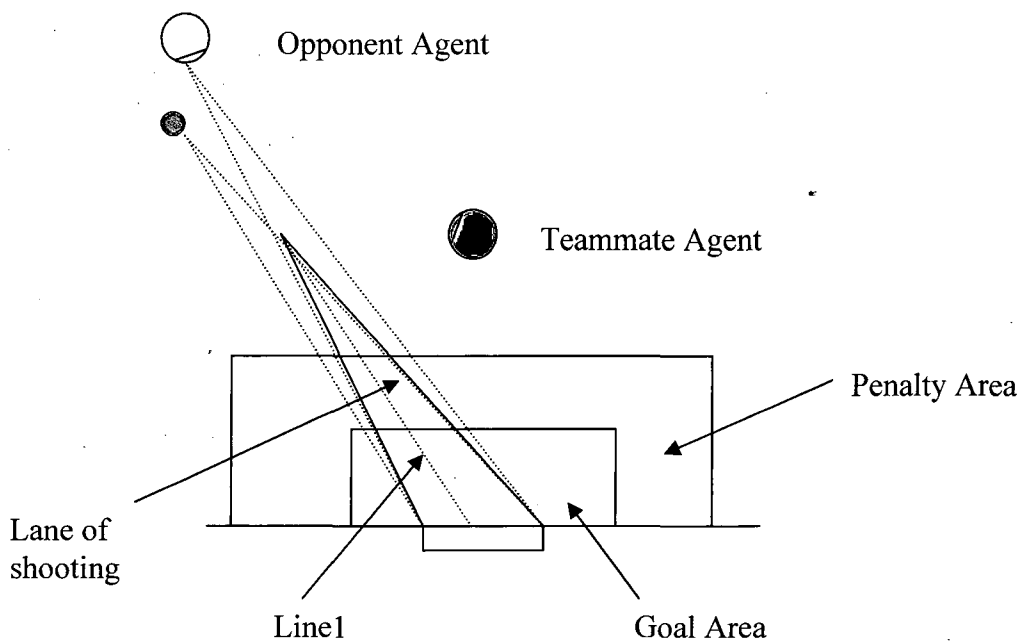


Fig.3.2 Lane of Shooting

### **Proper Positions**

Any agent who is exhibiting this move must locate itself to the appropriate positions so that it can execute its role according to the following rules:

1. IF (ball is far) THEN agent locate itself on points of Line1 which are above or on the boundary of penalty area.
2. IF (ball is in the middle) THEN agent locate itself on points of Line1 which are between penalty area and goal area.
3. IF (ball is near) THEN agent locate itself on points of Line1 which are inside the goal area.

This move uses the **Pass Evaluation** technique in order to achieve the better utilization of ball possession.

### **Algorithm for Blocking the Lane of Shooting**

```
Agent finds the Lane of Shooting and Line1;  
It faces directly towards the ball;  
Do  
{  
    Agent locates itself at Proper Positions;  
    IF( ball is approaching towards it)  
        THEN it intercepts the ball and uses Pass Evaluation to  
            Clear or Pass the ball;  
}  
While ( if further modification in positions are possible)
```

### 3.2.4 Build Up Play

Build Up Play means gaining and maintaining possession with the intention of goal. This move is normally executed when defense has sufficient number of players. Its execution has two phases, first is gaining possession and second phase includes maintaining possession with the intention of goal. This move considers only those agents which are in the vicinity of ball, out of them one or two do actual defensive tasks and rest of them act as supporting teammates.

#### A. Gaining Possession

Gaining possession means if an opponent was having **ball control** in previous simulation cycle and now (current simulation cycle) a teammate agent has ball control by **tackling** and **challenging** the ball. The algorithm for gaining possession as described below includes **Support1** as a subordinate behavior, which is explained in next few sections of this chapter. It also assumes that all players in the vicinity of ball can view ball.

It uses the following **low level Skills**:

- Ball Interception
- Marking
- Tracking



## Algorithm for Gaining Possession

```
IF (Teammate agent A1 is nearest to the ball)
  THEN {
    IF (ball is free) THEN take ball control;

    IF (ball is in kickable area & A1 is in the sides of the
        opponent controlling ball) THEN Tackle;

    ELSE Challenge the ball;
  }
ELSE
  {
    Other Teammate agent B executes Support1;

    IF (opponent having ball makes a pass near B)
      THEN {
        IF (pass is incorrect or poor)
          THEN take ball control;

        ELSE A1=B;
      }
  }
```

## B. Maintaining Possession

This behavior is executed after gaining ball possession and includes maintenance of ball control by the teammates when opponents are around, keeping in the view that ball must progress towards the opponent's goal (intention of goal). Like gaining possession, it considers only those agents which are in the vicinity of ball. It has two types of the role for agents, one who is having ball control and other who are supporting ball holder. This can be done in the following ways:

- A ball holding agent must make correct and useful passes when chance of losing possession or can execute turn ball behaviors (Appendix 1) if there is a chance of tackling or challenging.

- Other agents must support the ball holding agent in the forward direction (towards opponent's half).

This behavior includes **Support1** as subordinate behavior, **Turn Ball1**, **Turn Ball2** as low level behaviors and predicates like Chance of Tackling (predicates related with chance of losing possession ) etc.

It uses the following **low level Skills**:

- Clearing
- Ball Interception
- Dribbling
- Passing.
- Turning Ball and Body

### **Correct Passes**

A pass is said to be correct if it reaches the intended player at the proper positions where it can intercept it and can gain possession. In order to achieve this it uses **Pass Evaluation** to find out intended player and if it is not possible to make correct pass, then agent must clear or dribble according to **Pass Evaluation**.

### **Usefulness of Passes**

Once the agent has chosen number of correct passes, usefulness of the passes should be calculated so that agent can make an important pass. Usefulness of any pass will depend upon the number of opponents in a circle of radius  $R$  of the receiving agent and the ball progress. Ball Progress can be measured as distance  $d$  of ball from the centre of team's own goal and  $R$  is the maximum distance an agent can travel in simulation cycle.

### **Chance of Possession (CP)**

In order to capture the effect of number of opponents present in the circle of radius  $R$  of the receiver agent chance of possession is used. It calculates the likelihood of retaining possession if ball is being passed to the chosen agent. CP is inversely proportional to the number of opponents present near the receiver.

In the following computations  $nop_1$  denotes the number of opponents present within a circle of radius  $R$  from the centre of receiver, who are facing towards ball's current location or current ball holder, and  $nop_2$  denotes the number of opponents present within a circle of radius  $R$  from the centre of receiver, who are not facing towards ball's current location or current ball holder (see Fig. 3.3).

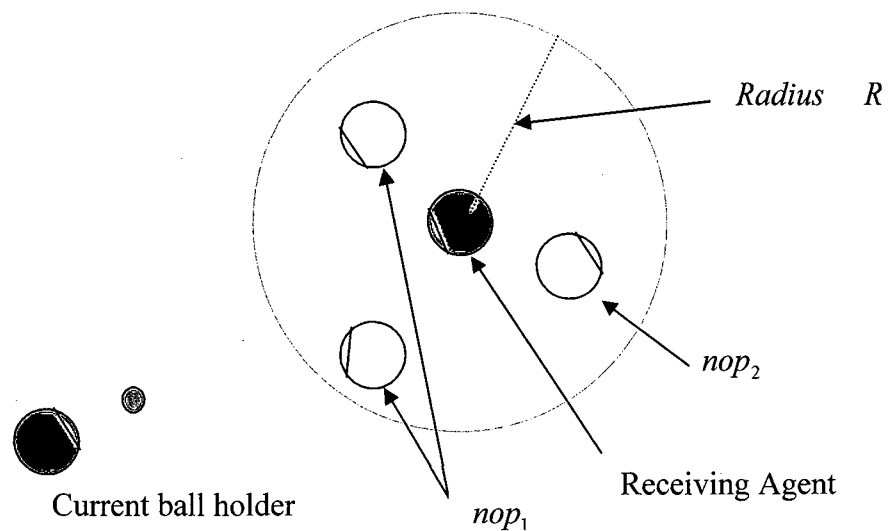


Fig 3.3 Area around a ball receiving agent

Cumulative effect of these opponents can be computed as:

$$X = w \times nop_1 + nop_2$$

Where  $w > 1$ , a weight associated with  $nop_1$ , as they can be major contributor in taking possession than  $nop_2$ .

CP can be computed as:

$$CP = \begin{cases} 1 & \text{if } nop_1 + nop_2 = 0 \\ \frac{.9}{w * nop_1 + nop_2} & \text{otherwise} \end{cases}$$

Now we can compute Usefulness of a Pass in terms of **CP** and *d*:

$$\text{Usefulness} = \begin{cases} 0 & \text{if } CP \leq .25 \\ d * CP & \text{if otherwise} \end{cases}$$

### Algorithm for Maintaining Possession

```

IF (Agent is controlling ball)
{
  IF (Chance of Tackling or Challenging)
  THEN {
    IF ((One side Tackling & Challenging) or One side
      Tackling or Challenging) THEN Turn Ball1;

    IF (Two side Tackling & Challenging) THEN Turn Ball2;

  }

  Make Correct and Useful passes;
}

ELSE
  Agent executes Support1 in forward direction;

```

Now the algorithm for Build Up Play can be written as:

## Algorithm for Build Up Play

```
While (Team's Possession = No)
    Team tries to Gain Possession;
```

```
IF (Team's Possession = Yes)
    THEN Maintain Possession;
```

### 3.2.5 Delay

Delaying means to stop the opponent's Progress in the field, so that other teammates can reach near the ball when defense does not have sufficient number of players. This can be achieved by blocking vicinity passes or direct shoot towards the goal. Like other behaviors it also considers only those players who are in the vicinity of ball. It has two roles for agents, first is for the First Defender (FD) and another one is for remaining players to support the FD agent.

This behavior includes **Support1** and **Stay Compact** as subordinate behaviors. It can be accomplished in a way that FD locates itself at the **Proper Position** in order to do the actual defensive tasks e.g. when ball comes to attacking area (shown in Fig. 3.4) it intercepts it. While other players execute Support1 and Stay Compact in order to help FD.

It uses the following **low level Skills**:

- Ball Interception
- Marking
- Tracking
- Turning Body

### Proper Position

In order to be at the proper position FD should locate itself on Line1 at angles  $\theta_1$  and  $\theta_2$  ( $0 \leq \theta_1, \theta_2 \leq 45^\circ$ ) from ball and current ball holding opponent respectively and at a minimum distance from ball (see Fig. 3.4).

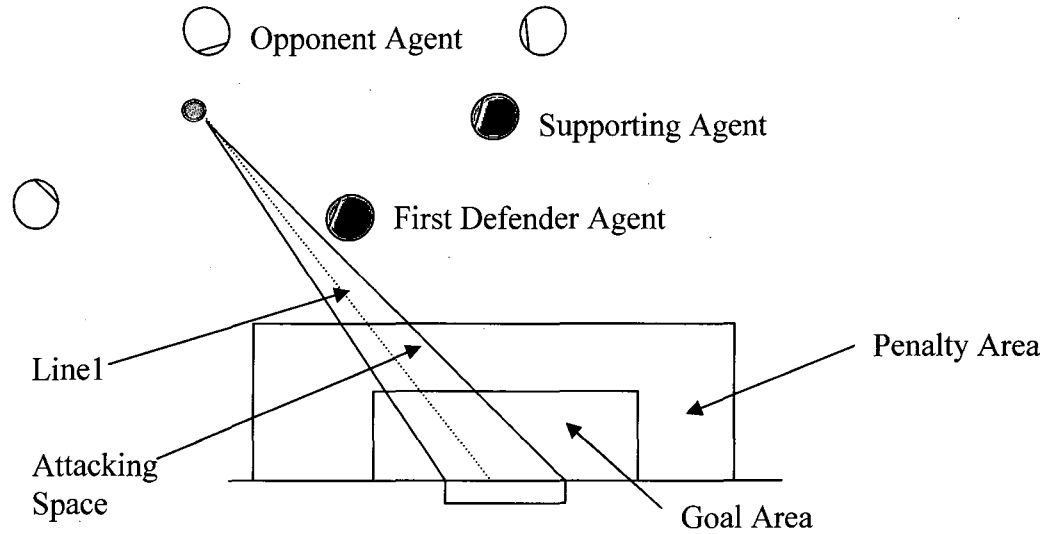


Fig. 3.4 Attacking Space

### Algorithm for Delay

```
IF ( Agent is a FD)
  THEN {
    Agent locates itself at Proper Position;

    IF (Ball comes to it)
      THEN Make Correct and Useful pass;
  }

ELSE
  Support1 FD towards the goal and Stay Compact;
```

### 3.2.6 Pressurize

It is a defensive tactic which is used when defense has at least two players. It is used to reduce the amount of space and time in which opponents can handle the ball, this can result into the possession of ball depending upon the situations. This behavior may or may not have specific direction of execution e.g. pressurizing towards flanks. This can be achieved in the following ways

- by blocking all potential passes in the vicinity
- FD challenges ball.

Like Delay it also has two roles for agents, FD agent and remaining players have to support FD according to the situations. It includes two subordinate behaviors Support1 and Stay Compact, and uses Pass Evaluation layer's output to determine vicinity passes.

It uses the following **low level Skills**:

- Ball Interception
- Marking
- Passing
- Tracking
- Turning Body

#### Algorithm for Pressurizing

```
IF (Agent is FD)
  THEN{
    Challenge the ball;

    IF( Ball comes to it or it gets ball control)
      THEN make correct and useful passes;
  }
ELSE
  Support1 FD and Stay Compact;
```

### 3.2.7 Safe Play

This behavior is executed when ball is near to the team's goal. It is used to nullify the chance of goal by opponents or it is used to convert dangerous play into normal play. Like other behaviors it considers only those agents who are in vicinity of ball. This can be achieved in the following way

- Agents can cover width of goal according to the situation in the goal area.
- Whenever any agent gets a chance to intercept the ball or gets ball then it makes Proper Clear or Useful pass.

It uses the following **low level Skills:**

- Ball Interception
- Passing
- Tracking
- Turning Body

#### Nearest Agents

These are those agents which are present in the Nearest Area as shown in Fig. 3.5

#### Proper Clear

Proper clear means clearing ball in a direction where no or least number of opponents are present.

#### Proper Location

If two or more nearest agents are present in the nearest area, then nearest area is distributed among them.

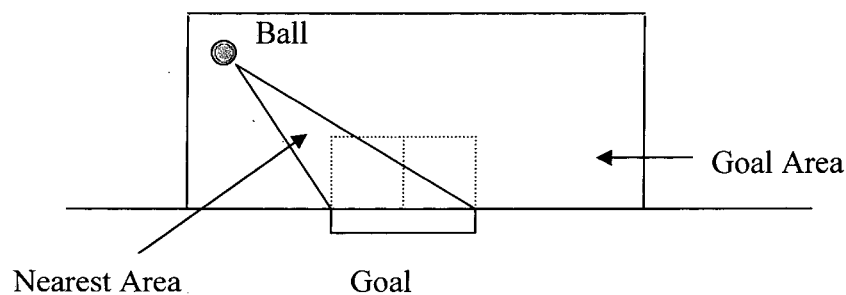


Fig. 3.5 Extended view goal



### Algorithm for Safe Play

```
IF (there is any Nearest Agent)
  THEN {
    IF (number of Nearest Agents =1)
      THEN it locates itself on the mid point of line joining
           centre of goal and ball;

    ELSE
      The agent locates itself at Proper Location;

    IF (ball comes to the agent)
      THEN Proper Clear
  }
ELSE
  {
    Agent moves towards Nearest Area;

    IF (ball comes to the agent or able to intercept ball)
      THEN Proper Clear or Useful pass;
  }
```

### 3.2.8 Support2

This is a behavior which is executed by the agents who are not in the vicinity of ball. In this agents first look for the ball in the field then try to come near to the ball, but if there are sufficient number of the players around the ball then it places itself near the vicinity of the ball depending upon the situations e.g. if the ball is in penalty area then it will move towards the goal area in order to support the players who are in the vicinity of ball. It has single role for the agent to play. It uses dashing and turning body as low level behaviors.

## Algorithm for Support2

```
Agent turns and moves to locate ball in the field;  
Agent moves towards ball;  
IF (sufficient number of players in the ball's vicinity)  
    THEN it locates itself near to the balls vicinity's boundary;  
ELSE  
    it locates itself in the balls vicinity;
```

### 3.3 Conditions

Like Shaped Reinforcement Learning, this work has a set of predefined conditions and they are clusters of various subconditions of the environment. In this work as working environment of the agents is chosen from RoboCup Soccer Server hence these conditions has direct relationship with the playing modes of the simulator. Soccer simulator has eleven playing modes [Foroughi et. al 2004] out of which ten playing mode cane described with the hand coded rule easily, situations in these modes are set e.g. kick\_offs, goal\_kick. But play\_on is the most important, which depicts the normal game situations apart from the set plays. Situations in play mode are highly dynamic which makes it extremely difficult to be captured into hand coded rules (because of large search space). Hence the situations used in this work reflect the play on mode of the simulator. [Nakashima et. al 2004 ] and [Konur et. al 2004] have divided the field of soccer in to the different zones based on the location of the ball on the field , similarly I have divided the defensive half of a team in to five Zones as shown in the Fig. 3.6.

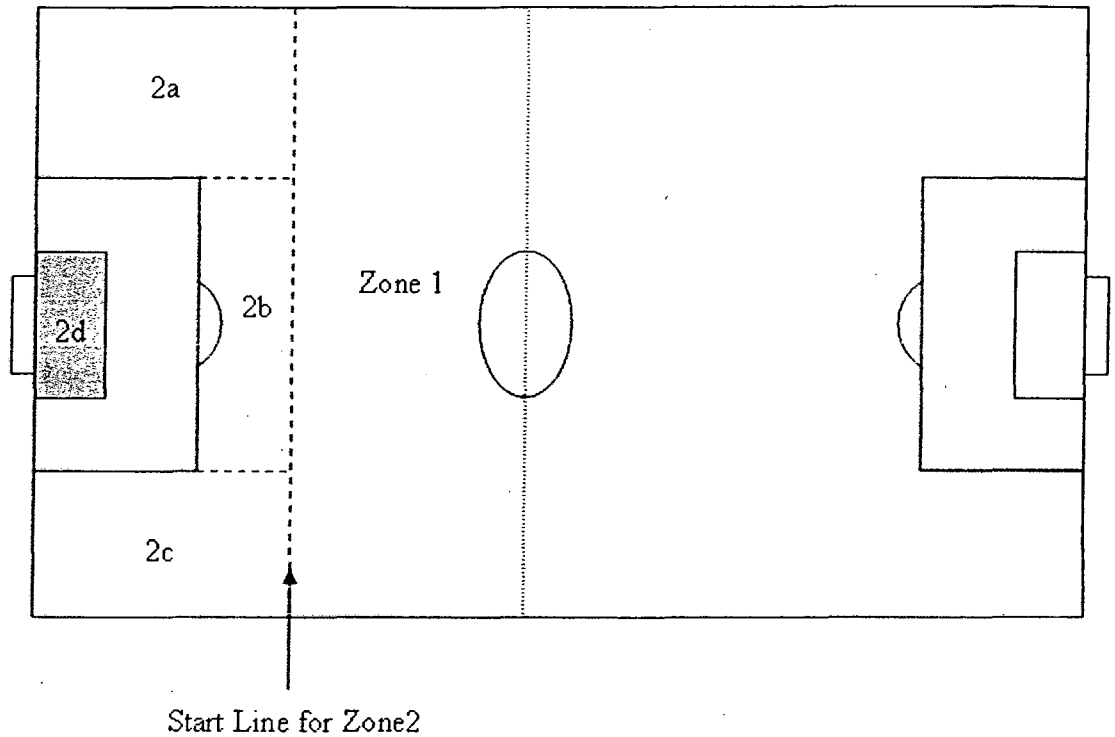


Fig 3.6 Defensive Zones

As soccer is a dynamic game, so the situation will not merely dependent on the ball's position on the field but also on the teammates and opponents present near the ball which can affect the ball, moreover agents considered here have limited sensing (seeing and dashing) (Appendix -2). In order to capture the real situations of soccer field the concept of vicinity is used . Hence defensive conditions chosen in this work are dependent on the following factors:

- Balls location on the field (one out of five zones in Fig. 3.6).
- Number of the teammates present in the vicinity of ball.
- Number of opponents present in the vicinity of ball.

These defensive conditions are based on the fact that as soon as team gains possession defense is over. Hence here only those conditions are included when team does not have

possession. Depending on the Fig. 3.6 these situations can be divided broadly into five main categories as:

1. ball is in zone 1
2. ball is in zone 2a
3. ball is in zone 2b
4. ball is in zone 2c
5. ball is in zone 2d

Again these main conditions are divided in to hierarchy of sub-conditions depending upon the number of teammates and opponents present in the balls vicinity as given below:

Sub-conditions based on the number of opponents present in the vicinity of ball are as:

1. No opponent present.
2. Moderate (one or two) number of opponents is present.
3. Higher (three or four) number of opponents is present.
4. Highest (five or more) number of opponents is present.

Sub-conditions based on the number of teammates present in the vicinity of ball are as:

1. No teammate is present.
2. One teammate is present.
3. Two teammates are present.
4. Three teammates are present.
5. Four teammates are present.
6. Five teammates are present.

Every defensive condition considered in the proposed work has one main condition (based on the ball's location on the field) and two sub-conditions from the sub-conditions related with number of opponents and sub-conditions related with number of teammates present in the vicinity of ball. In this way complete defense can be described by 120 ( $5 \times 4 \times 6$ ) conditions.

### 3.4 Heterogeneous Reward Function

As heterogeneous reward functions are multimodal feedback available from external and internal states, in order to find out these feedbacks events related with behaviors achievement should be known. These events can trigger in any condition and any point of time. Hence I have chosen following events:

$E_{gs}$  - Opponents has scored a goal

$E_{ob}$  - Ball goes out of bound

$E_{ck}$  - Corner kick is awarded to the opponents

$E_{pk}$  - Penalty kick is awarded to the opponents

$E_{fk1}$  - Free kick is awarded to the opponents

$E_{fk2}$  - Free kick is awarded to the team

$E_{gk}$  - Goal kick is awarded to the team

As events are the signals which shows achievement of a any behavior, but all the agents are not equally contributing towards any event because all are not executing the same behavior. In order to incorporate this fact, unlike shaped reinforcement explained in chapter 2 we have considered two types of Heterogeneous Reward Functions, one for those agents who have major contribution on the occurrence of the event and another who are less responsible for the event's occurrence.

Let  $A_i$  be any team member agent who has been in the ball's vicinity during three simulation cycles before occurrence of the event.

Heterogeneous Reward Function for  $A_i$  agents

$$R_E^1(c,t) = \begin{cases} gs_1 & \text{if } E_{gs} \\ ob & \text{if } E_{ob} \\ ck & \text{if } E_{ck} \\ pk & \text{if } E_{pk} \\ fk_1 & \text{if } E_{fk_1} \\ fk_2 & \text{if } E_{fk_2} \\ gk & \text{if } E_{gk} \\ 0 & \text{otherwise} \end{cases}$$

Where  $gs_1, ob, ck, pk, fk_1 < 0$  and  $fk_2, gk > 0$

Heterogeneous Reward Function for agents who are not  $A_i$

$$R_E^2 = \begin{cases} gs_1 & \text{if } E_{gs} \\ 0 & \text{otherwise} \end{cases}$$

Where  $gs_1 < 0$

### 3.5 Progress Estimators

Progress estimators are evaluation metrics relative to the current goal of an agent hence it captures the local progress of a behavior. In this problem domain every individual agent has to perform its role and top of this it's performance in the role is going to contribute for teamwork. Hence performance estimators in this work has two level i.e. first level is to evaluate the performance of the individual agent in a defensive role and at the second level there are performance estimators for behaviors which are derived from first level. All constants used in the performance estimator's formulae are positive real numbers and predicates are given in Appendix-2.

### 3.5.1 Individual Performance Estimators

Since individual level defense is based on the agent's role in the behaviors progress estimators at individual level are divided into three categories as given below:

#### a. Progress Estimators for Actual Defense

If an agent is in the ball vicinity and performing one of the following tasks, tackling or challenging (the progress estimators for challenging are same as of tackling so we have the progress estimators for tackling only) or FD then it is doing actual defensive task.

#### Progress Estimators for First Defender

Positional correctness estimator  $R_{FD_1}(c, t)$  of the first defender is given below:

$$R_{FD_1}(c, t) = \begin{cases} A & \text{if } \theta_1, \theta_2 \leq 45^\circ \\ -A_{error} & \text{if } \theta_1, \theta_2 > 45^\circ \\ pa & \text{if } |\theta_1 - \theta_2| \text{ is reducing} \\ -pa & \text{if } |\theta_1 - \theta_2| \text{ is increases} \\ pd & \text{if } d_{ball} \in [d_{min1}, d_{max1}] \\ -pd & \text{if } d_{ball} \notin [d_{min1}, d_{max1}] \\ 0 & \text{otherwise} \end{cases}$$

Where  $\theta_1, \theta_2$  are agent's relative angles from ball and the ball holding opponent and  $d_{ball}$  is the distance of the agent from ball.

Ball handling efficiency estimator  $R_{FD_2}(c, t)$  of FD can be formulated as:

$$R_{FD_2}(c, t) = \begin{cases} -bm_1 & \text{if } ball\_miss1 \\ bc_1 & \text{if } gained\_ball\_control1 \\ 0 & \text{otherwise} \end{cases}$$

Total progress estimator  $R_{FD}(c, t)$  of a FD is computed as:

$$R_{FD}(c,t) = R_{FD2}(c,t) + K_1 R_{FD1}(c,t) \text{ , Where } K_1 > 1$$

### Progress Estimators for Tackling Agent

Positional correctness estimator  $R_{Tack1}(c,t)$  of an agent who is tackling is computed as:

$$R_{Tack1}(c,t) = \begin{cases} da & \text{if agent dashes at } \theta_1 = 0 \\ -da & \text{if } \theta_1 \neq 0 \\ k_i & \text{if kicking angle } \theta_k \in [0,45^\circ] \\ -k_i & \text{if kicking angle } \theta_k \notin [0,45^\circ] \\ 0 & \text{otherwise} \end{cases}$$

Where  $\theta_k$  is kicking angle at which agent kicks the ball.

Ball handling efficiency estimator  $R_{Tack2}(c,t)$  of an agent who performs tackling is given below:

$$R_{Tack2}(c,t) = \begin{cases} -bm_2 & \text{if ball\_miss2} \\ bc_2 & \text{if gained\_ball\_control2} \\ 0 & \text{otherwise} \end{cases}$$

Total progress estimator  $R_{Tack}(c,t)$  for an agent who performs tackling is computed as:

$$R_{Tack}(c,t) = R_{Tack2}(c,t) + K_2 R_{Tack1}(c,t) \text{ where } K_2 > 1.$$

### Progress Estimators for Blocking the Lanes of Shooting

Positional correctness estimator  $R_{BL1}(c,t)$  of the agent blocking lanes of shooting is formulated as:



$$R_{BL1}(c,t) = \begin{cases} A & \text{if } \theta_1 \text{ and } \theta_2 \leq 45^\circ \\ -A_{error} & \text{if } \theta_1 \text{ or } \theta_2 > 45^\circ \\ pa & \text{if } |\theta_1 - \theta_2| \text{ decreases} \\ -pa & \text{if } |\theta_1 - \theta_2| \text{ increases} \\ dg & \text{if distance from the goal decreases} \\ -dg & \text{if distance from the goal increases} \\ 0 & \text{otherwise} \end{cases}$$

Ball handling efficiency estimator  $R_{BL2}(c,t)$  of an agent who performs block the shooting lanes is given below:

$$R_{BL2}(c,t) = \begin{cases} -bm_1 & \text{if ball\_miss1} \\ bc_1 & \text{if gained\_ball\_control1} \\ 0 & \text{otherwise} \end{cases}$$

Total progress estimator  $R_{BL}(c,t)$  for an agent who performs block the shooting lanes behavior is computed as:

$$R_{BL}(c,t) = R_{BL2}(c,t) + K_3 R_{BL1}(c,t) \text{ where } K_3 > 1.$$

## **b. Progress Estimators for Team Support**

Any agent is in the team support if it is handling an opponent or covering the area behind FD.

### **Progress Estimators for Handling Opponents**

Positional correctness estimator  $R_{OH1}(c,t)$  of an agent  $A_i$  handling opponents is formulated as:

$$R_{OH1}(c,t) = \begin{cases} A & \text{if } 0 \leq \theta_1, \theta_2 \leq 45^\circ \\ -A_{errur} & \text{if } \theta_1, \theta_2 > 45^\circ \\ pa & \text{if } |\theta_1 - \theta_2| \text{ is reducing} \\ -pa & \text{if } |\theta_1 - \theta_2| \text{ is increases} \\ pd & \text{if } d_{io} \in [d_{\min 2}, d_{\max 2}] \\ -pd & \text{if } d_{io} \notin [d_{\min 2}, d_{\max 2}] \\ 0 & \text{otherwise} \end{cases}$$

Where  $d_{io}$  is the distance of agent  $A_i$  from its nearest opponent.

Ball handling efficiency estimator  $R_{OH2}(c,t)$  of an agent who handles opponent is given as:

$$R_{OH2}(c,t) = \begin{cases} -bm_3 & \text{if } ball\_miss3 \\ bc_3 & \text{if } gained\_ball\_control3 \\ 0 & \text{otherwise} \end{cases}$$

Total progress estimator  $R_{OH}(c,t)$  for an agent who performs handling opponent behavior is computed as:

$$R_{OH}(c,t) = R_{OH2}(c,t) + K_4 R_{OH1}(c,t) \text{ where } K_4 > 1.$$

### Progress Estimators for Covering

Positional correctness estimator  $R_{COV1}(c,t)$  of covering agent is formulated as:

$$R_{COV1}(c,t) = \begin{cases} A & \text{if } \theta_1 \text{ and } \theta_2 \leq 45^\circ \\ -A_{error} & \text{if } \theta_1 \text{ or } \theta_2 > 45^\circ \\ pa & \text{if } |\theta_1 - \theta_2| \text{ decreases} \\ -pa & \text{if } |\theta_1 - \theta_2| \text{ increases} \\ dg_1 & \text{if distance from the goal decreases} \\ -dg_1 & \text{if distance from the goal increases} \\ pos & \text{if agent is in uncovered attacking space} \\ -pos & \text{if agent is not in uncovered attacking space} \\ df & \text{if } d_{ifd} \in [d_{min,3}, d_{max,3}] \\ -df & \text{if } d_{ifd} \notin [d_{min,3}, d_{max,3}] \\ 0 & \text{otherwise} \end{cases}$$

Where  $d_{ifd}$  is the distance of agent  $A_i$  (covering agent) from the FD.

Ball handling efficiency estimator  $R_{COV2}(c,t)$  of an agent who performs covering is given as:

$$R_{COV2}(c,t) = \begin{cases} -bm_3 & \text{if ball\_miss3} \\ bc_3 & \text{if gained\_ball\_control3} \\ 0 & \text{otherwise} \end{cases}$$

Total progress estimator  $R_{COV}(c,t)$  for an agent who performs covering behavior is computed as:

$$R_{COV}(c,t) = R_{COV2}(c,t) + K_5 R_{COV1}(c,t) \text{ where } K_5 > 1.$$

### c. Progress Estimators for the Agents not in the vicinity of ball

Total progress estimator  $R_{NIA}(c,t)$  for agents who are not in the vicinity of the ball is computed as:

$$R_{NVA}(c,t) = \begin{cases} m & \text{if distance agent from vicinity of the ball is reducing} \\ -m & \text{if distance agent from vicinity of the ball increases} \\ 0 & \text{otherwise} \end{cases}$$

### 3.5.2 Progress Estimators for Main Behavior

Now, after knowing individual's progress estimators, their cumulative effect can be used to compute the progress of a behavior. Any behavior's performance can be measured on the basis of three factors, the time opponents get to control the ball, space opponents get to control the ball and ball progress e.g. the purpose of delay behavior is to stop ball progress and to reduce the space opponents get to control the ball. Progress estimators of main behaviors are discussed below:

In order to compute the amount of space ball control for opponents all the positional correctness estimator of the agents who are in the vicinity of ball are considered as:

$$X = \alpha \sum_{\forall i} R_{i1}(c,t) + (1-\alpha) \sum_{\forall j} R_{j1}(c,t)$$

Where  $A_i$  are agents who are doing actual defensive tasks,  $A_j$  are supporting agents and  $\alpha$  is parameter and  $.5 \leq \alpha \leq 1$ . Variable  $X$  indicates the cumulative effect of positional correctness of all agents present in the vicinity of all and inversely proportional to the amount of space opponent get to control ball.

$R_{SB}(c,t)$  estimates the behavior's performance for ball progress and a space opponent get to control ball. In order to calculate the progress estimators for time of ball control the behavior is observed up to five simulation cycles and average of number of times the team has gained or lost the ball possession calculated to decide the reward for the behavior.

$$R_{SB}(c,t) = \begin{cases} -sp & \text{if } X \text{ is reducing} \\ sp & \text{if } X \text{ increases} \\ l & \text{if progress of ball is decreased} \\ -l & \text{if progress of ball is increased} \\ 0 & \text{otherwise} \end{cases}$$

$R_{TC}(c,t)$  estimates the behavior's efficiency towards reducing the time of ball control opponents gets in defense when defensive team is executing one of above written main behaviors.

$$R_{TC}(c,t) = \begin{cases} tc & \text{if time of ball control is reducing} \\ -tc & \text{if time of ball control increases} \\ 0 & \text{otherwise} \end{cases}$$

### **Total Reward**

Total reward for an agent while executing a behavior  $R(c,t)$  is computed as the weighted sum:

$$R(c,t) = uR_E^i(c,t) + vR_{SB}(c,t) + wR_{TC}(c,t)$$

Where  $u, v, w > 0$  and  $u + v + w = 1$ .

This algorithm learns a value function that maps conditions  $c$  to behavior  $b$ . This function is used by behavior selection algorithm to choose most appropriate behavior for each condition in which agent finds itself in. The learning system is matrix based and maintains a matrix  $A(c,b)$  estimates whose entries reflect a normalized sum of reinforcement  $R$  received for each condition-behavior pair over time  $t$ :

$$A(c, t) = \sum_{t=1}^T R(c, t)$$

The values in the matrix varies over a time based on received reinforcement, these are collected during the execution of behavior, and updated and normalized when behaviors are switched.

## Chapter 4

### Benefits of Extended Shaped Reinforcement Learning

Extended Shaped Reinforcement learning meets some of the on line learning challenges given in [Kitano et al. 98]. It provides great scope for the agents to have adaptations in various defensive conditions than the other existing learning techniques at defensive teamwork level. Since the learning algorithm includes more intermediate reinforcement hence it allows agents to acquire more knowledge about their environment and this gained knowledge makes learning algorithm to converge fast. The major benefits of extended shaped reinforcement learning algorithm are given below.

#### 4.1 Reduction of State Space

There are  $10^{198}$  external states for an agent in Simulated RoboCup Soccer field which makes learning space extremely large to be managed by Q-learning or any other traditional reinforcement learning algorithm. [Stone and Veloso 99] give a mechanism to reduce the state space, in which TPOT-RL learning algorithm reduces number of states by exploiting action dependent features to create smaller vector space which is very similar to the state space. TPOT-RL reduces the number of states to 2816. If half the number of states belongs to defense i.e. 1408 states, which almost 11 times the number of the conditions (120) proposed in this work, this reduction is achieved through off line selection of important situation which are clusters of the states of a given environment .

#### 4.2 Speed up in Learning

The more subgoals the system recognizes, the more frequently reinforcement can be applied, and the faster the learner can converge [Mataric 97]. In this work, there are several progress estimators for various defensive roles in a behavior. If we consider TPOT-RL, it has three types of the rewards for every agent. Since the proposed framework gives a clear distribution of reward/punishment among the agents, it allows the learning algorithm to converge faster and allows agents to adapt in arbitrary defensive

conditions because of the inclusion of more domain knowledge. The proposed learning scheme allows converting intermittent and delayed feedback to continuous error signals.

### **4.3 Efficient action selection mechanism**

Action selection is one of the major problems in reinforcement learning techniques especially when agent has large learning space i.e. large action space and state space. The action selection mechanism in reinforcement learning works on the trade off between explorations vs. exploitations [Sutton and Barto 98] . A learning algorithm first explores the action space to choose useful actions that is the exploration rate is kept higher and at later stages exploration rate increases at the cost of exploitation rate. TPOT-RL used exploitations with probability  $p$  and exploitations with  $1-p$  probability, where  $p$  gradually increases from 0 to .99. But in the proposed work there is implicit and efficient action selection mechanism as because the  $A(c, b)$  the matrix to be learned has at any point of time has cumulative effectiveness of the actions which reduces the overhead of having explicit action selection mechanism.

### **4.4 Results**

The learning algorithm given in this work learns a value function which maps conditions to the behaviors. After applying the learning framework given in chapter 3 the agents will learn the metrics given in Tables 4.1 to 4.4 depending on the ball's positions in the defensive field (ball can be in one of the five zones, Zone1 and Zone 2a – 2d, as given in Fig. 3.6).

In the following tables X is the number of teammates present in the vicinity of ball and Y is the number of opponents present in the vicinity of ball



**Table 4.1 Mappings learned when ball is in Zone 1**

	Y=0	0<Y<3	3 ≤ Y < 5	Y ≥ 5
X=0	No Operation	No Operation	No Operation	No Operation
X=1	Build Up Play with other players in defense	Delay	Delay with other players in defense	Block the Lane of Shooting
X=2	Build Up Play	Pressurize	Delay	Delay with other players in defense
X=3	Build Up Play	Build Up Play	Pressurize	Pressurize
X=4	Build Up Play	Build Up Play	Build Up Play	Build Up Play
X=5	Build Up Play	Build Up Play	Build Up Play	Build Up Play

**Table 4.2 Mappings learned when ball is in Zone 2a**

	Y ≤ 1	Y=2	Y=3	Y=4	Y ≥ 5
X=0	No Operation	No Operation	No Operation	No Operation	No Operation
X=1	Delay	Block the Lane of Shooting	Block the Lane of Shooting	Block the Lane of Shooting	Block the Lane of Shooting
X=2	Build Up Play	Pressurize towards Zone 1 (Flank side)	Delay	Delay	Block the Lane of Shooting
X=3	Build Up Play	Build Up Play	Pressurize towards Zone 1 (Flank side)	Delay	Delay
X=4	Build Up Play	Build Up Play	Build Up Play	Build Up Play	Pressurize towards Zone 1 (Flank side)
X=5	Build Up Play	Build Up Play	Build Up Play	Build Up Play	Pressurize towards Zone 1 (Flank side)

**Table 4.3 Mappings learned when ball is in zone 2b**

	$Y \leq 1$	$Y=2$	$Y=3$	$Y=4$	$Y \geq 5$
X=0	No Operation	No Operation	No Operation	No Operation	No Operation
X=1	Block the Lane of Shooting	Block the Lane of Shooting	Block the Lane of Shooting	Block the Lane of Shooting	Block the Lane of Shooting
X=2	Pressurize towards Zones 2a or 2c	Pressurize towards Zones 2a or 2c	Delay	Delay	Block the Lane of Shooting
X=3	Build Up Play	Build Up Play	Pressurize towards Zones 2a or 2c	Delay	Delay
X=4	Build Up Play	Build Up Play	Build Up Play	Pressurize towards Zones 2a or 2c	Pressurize towards Zones 2a or 2c
X=5	Build Up Play	Build Up Play	Build Up Play	Pressurize towards Zones 2a or 2c	Pressurize towards Zones 2a or 2c

**Table 4.4 Mappings learned when ball is in Zone 2d**

	$Y \leq 1$	$Y=2$	$Y=3$	$Y=4$	$Y \geq 5$
X=0	No Operation	No Operation	No Operation	No Operation	No Operation
X=1	Block the Lane of Shooting	Block the Lane of Shooting	Block the Lane of Shooting	Block the Lane of Shooting	Block the Lane of Shooting
X=2	Safe Play	Safe Play	Safe Play	Safe Play	Safe Play
X=3	Build Up Play	Build Up Play	Pressurize towards Zones 2a or 2c or 2b	Safe Play	Safe Play

X=4	Build Up Play	Build Up Play	Pressurize towards Zones 2a or 2c or 2b	Pressurize towards Zones 2a or 2c or 2b	Safe Play
X=5	Build Up Play	Build Up Play	Build Up Play	Pressurize towards Zones 2a or 2c or 2b	Safe Play

This is to be noted that the mappings to be learnt when ball is in Zone 2c are the same as that of Zone 2a (Table 4.2)

## Chapter 5

### Conclusion and Future work

This dissertation discusses improvement of defensive tactics as its key learning task which can be in any learning layer above pass evaluation or pass selection in layered learning framework. The major focus of this work is to extend shaped reinforcement learning to capture the ethics of teamwork and this accomplished by splitting progress estimators in order to get reinforcement at individual level and at teamwork level, and multiple heterogeneous reward functions.

This novel approach which is used to improve defensive tactics of agents at teamwork level. The major benefits of the proposed approach are to enhance learning rate, reduction in the number of states and hence the reduction of learning space by off line clustering of situations, and to provide an implicit and convenient action selection mechanism.

#### **Future directions**

Although the focus the proposed work is on the defensive tasks but it can be efficiently applied to the tasks of simulated soccer like improvement in offensive tactics, pass selection etc. The proposed framework of learning is general can be applied effectively in any complex learning domain. There is a scope to make learning even faster in this work by adding internal biases to the agents e.g. having probable actions for the conditions .This work is carried out under the assumption that An agent is able to know complete information about its other teammates present in its vicinity though direct or indirect communication. By omitting this assumption it will be extremely difficult to see the collective effect of the individual's performances. One of the future research directions will be to carry out learning without above mentioned assumption and to extend the proposed framework to a full fledged RoboCup Soccer team.

# Appendix 1

This appendix gives a brief description of defensive soccer terms and predicates used in chapter 3.

## **Actual Defensive tasks**

Marking, tracking, challenging or tackling are known as actual defensive tasks.

### **Ball control**

An agent is having a ball control iff

- Ball is in Kickable area.
- Agent's facing direction is least relative angle to the ball among the other agents present in the vicinity of the ball.
- It is at least distant to the ball.

### **Challenging**

If an opponent is having the ball in the current simulation cycle and an agent try to get ball control by making direct run towards the ball provided that ball is at reachable distance.

### **First Defenders (FD)**

First defender is the nearest defender to the ball from goal side.

### **Marking**

Guarding the goal side and ball side to prevent the marked opponent to from turning the ball or from moving with the ball towards the goal.

### **Second Defender (SD)**

The defender close enough to support the First defender by covering the space behind the first defender or by handling the opponents.

### **Tackling**

Making the runs in the sides of opponent handling the ball and kicking the ball in the desired direction to get the ball control.

### **Third Defender (TD)**

All other defenders other than the first and second defenders are third defenders.

### **Tracking**

Running with the marked opponent to prevent it from gaining possession of the ball in the available space behind or beside other defenders.

### **Turn Ball**

Turning the ball at  $90^0$  angle from the current ball position.

## **Turn Ball2**

Turning the ball at  $180^{\circ}$  angle from the current ball position.

## **Vicinity of ball**

An agent is said to be in the vicinity of the ball if it can view the ball in current simulation cycle and it can reach to the ball in next five simulation cycles.

$$d_{\min 1}$$

It is the minimum distance an agent as a first defender must keep from the ball and can travel in two simulation cycles with its full stamina.

$$d_{\min 2}$$

It is the minimum distance of a supporting agent (performing opponent handling) from its nearest opponent. It is the distance which agent can cover in one simulation cycle with its full stamina.

$$d_{\min 3}$$

It is the minimum distance of a covering agent (performing team support) from the first defender and agent can travel this distance in two simulation cycles with its full stamina.

$$d_{\max 1}$$

It is the maximum distance of an agent as a first defender from the ball and can travel in four simulation cycles with its full stamina.

$$d_{\max 2}$$

It is the maximum distance of a supporting agent (performing opponent handling) from its nearest opponent. It is the distance agent can travel in two simulation cycle with its full stamina.

$$d_{\max 3}$$

It is the maximum distance of a covering agent (performing team support) from the first defender and agent can travel this distance in three simulation cycles with its full stamina.

Predicated which are being used in chapter3 are as:

**ball is free**

When there is no agent present in the vicinity of ball then ball is said to be free

**ball is far**

If the distance of the ball from an agent is more than the average distance agent can travel in five simulation cycles.

**ball is in the middle**

If the distance of the ball from an agent is in proximity of the distance agent can travel in two to three simulation cycles with its full stamina.

**ball is near**

If the distance of the ball from an agent is less than or equal to the distance agent can travel in one simulation cycle with its full stamina.

**ball\_miss1**

If the opponent having ball control passes or shoots the ball towards the goal and first defender can't intercept the ball.

**gained\_ball\_control1**

If the opponent having ball control passes or shoots the ball towards the goal and first defender manages to get the ball control.

**ball\_miss2**

If an agent who is tackling or challenging miss the ball before kicking because opponent holding the ball can turn the ball.

**gained\_ball\_control2**

If an agent who is tackling or challenging gets the ball control after the kick.

**ball\_miss3**

If an agent is executing team supporting defensive activities and miss the ball because of poor pass by the opponents or because of the agent's inability to intercept the pass.

**gained\_ball\_control3**

If an agent is executing team supporting defensive activities gains the ball control because of the incorrect pass by opponent or because of agent's capability to intercept the ball.

**Chance of Tackling**

If a teammate agent has ball control, opponents are in the kickable area of ball, opponents are in the sides of the ball holder and approaching towards the ball.

**Chance of Challenging**

If a teammate agent has ball control, opponent (nearest opponent to the ball) is present in the vicinity of ball, directly facing towards the ball and approaching towards the ball.

**One side Tackling**

If a teammate agent has ball control, only one opponent is in the kickable area of ball, it is in the either side of the ball holder.

**One side Tackling & Challenging**

If a teammate agent has ball control, one opponent tries to tackle from either side of the ball holder and one opponent tries to challenge the ball.

**Two side Tackling & Challenging**

If a teammate agent has ball control, two opponents try to tackle form both the sides of the ball holder and one opponent tries to challenge the ball.



## Appendix 2

### RoboCup Soccer server

In this Appendix, the most important features of the RoboCup Server Simulator have been described briefly the more detailed information is in [Foroughi et. al 2004]. This is a challenging simulator which can be compared to reality in many situations. The RoboCup Soccer Server contains many real-world complexities and is a very challenging and realistic base that the agent must handle.

The RoboCup Soccer Server is almost a real-time system that works with discrete time intervals of 100ms. During one cycle an agent receives different information from the sensors surrounding him and the field. That is why the agent must respond immediately in each cycle in order to complete an action. This requires real-time decision making. When an agent decides what to do, the action will not be executed earlier than in the end of the cycle. At the same time the server updates the state of the environment. Therefore the server uses a discrete action model while, on the other hand, the agent decides in real-time.

#### A2.1 Overview

Each agent is a separate client program that works on its own. It can not communicate directly but communicate through the server only. All the agents on the field use the only communication channel which has low bandwidth and is also very unreliable. The sensors provided for each agent are: the aural sensor, the body sensor and the visual sensor. The body sensor provides the agent with information such as the stamina or the current speed, physical information. It also provides information on how many actions the agent has performed. The aural sensor provides the agent with information about detected messages sent by other player objects. It has a limited range and capacity and one agent can only hear one message from a nearby team-mate every second simulation cycle. Then we have the visual sensor which provides the agent with information about the objects in his current field of view information such as distances, directions etc. It also works as a proximity sensor that can “feel” objects that are very close but behind the agent. The information the agent receives is relative from his perspective and can also be converted into global coordinates using landmarks, flags. The noise is added by quantizing the information, sent by the server.

The player object, also called the agent, can perform different types of actions, which are divided into two categories, primary actions and concurrent actions. During one cycle only one primary action can be executed, whereas multiple concurrent actions can be

performed simultaneously with any primary action. Only one primary action can be performed even if there is more than one being sent.

Acting and sensing in the soccer server are asynchronous. This means that visual information arrives at 150ms intervals and an agent can only perform a primary action once every 100ms. This means that in some cycles the agent must act without receiving new visual information, and this requires the ability to predict the current world state based on past states.

When the soccer server simulates object movement, the velocity of an object is added to its position. The velocity decays by a certain rate and increases by the acceleration of the object resulting from certain action commands. Noise is added to the movement of all objects to reflect unexpected movements of objects in the real world. The soccer server prevents the player objects from keeping constant maximum speed by assigning a limited stamina to each of them. When a player is performing a dash, the action consumes some of his stamina, but it is also slightly restored in each cycle. Then the player objects can be divided into different type of players, so called heterogeneous players. Whenever a new game or period is started, each team can select from several different types with different characteristics.

## **A2.2 Soccer server and clients**

The system providing simulation is the soccer server. It is a central point of connection for a number of players, partitioned into two teams, playing against each other. The server delivers sensing information to the players such as auditory, visual, and body-sensing information. Each player analyses the receiving information and decides what to do such as kick or turn, then sends commands back to the server. The server is associated with soccer monitors that display the pitch and players on computer screens, designed for visualization purpose.

Soccer clients such as players and coach connect with the server through UDP/IP sockets. Each client is an independent process, exchanging information with the server through a specific port (the default is 6000). A team can have up to 12 clients including 10 fielders, a goalie and a coach. A player client is an agent, surrounded by the environment, which is simulated by the server. Players are provided with information, but partial and noisy about the environment (A2.4 *Players*). Their main tasks are to make decision on which action to take such as kick, turn, and dash.

## **A2.3 How soccer simulation performs**

Internally in the soccer server, the simulation is divided in a sequence of time steps, called simulation cycles. At the beginning of each cycle, the server receives action commands performed by all players connected with it. Then, it calculates the effect of these actions to the environment such as ball direction, player's stamina and vision. At the end of that cycle, it returns environment information to each player as player's sensing information. The interval for each cycle can be customized (the default is 100 milliseconds).

Information passing back and forth between the server and clients is under the form of text messages. These messages are well formatted like commands with parameter values. For example, when a player wants to kick to the ball with the power of 50 straight to its body direction, it sends this text command “(kick (50, 0))” to the server. It is important that the server should receive these commands before next simulation cycle. Otherwise, the server does not wait to receive player’s messages, resulting that players will miss the opportunity to act, therefore, badly impact its performance.

## **A2.4 Players**

Virtual players are simulated with two important abilities: perception of the world and action to affect the world. Perception information arrives under the form of aural and visual information, created by simulated sensors. Based on their perception, players make decision on which action they should take to affect the play. A player connects to the soccer server from a specific port (the default is 6000). The maximum number of players for each team is 11 including 10 fielders and a goalie. Players send and receive information under the form of text commands with associated parameters. Perception information is noisy. In each simulation cycle, the server adds noise to the ball’s and player’s movement. Therefore, players cannot know the world exactly nor can they affect the world exactly the way they want.

### **a. Soccer pitch layout**

The soccer pitch composes of stationary objects including lines and flags as shown in Fig. A2.1. There are both horizontal and vertical lines, defined by a fixed name such as “(line t)” for the top horizontal line. Flags are particular points, uniquely named, such as “(flag c)” for a central point of the pitch. Both soccer server and players use line and flag objects. The soccer server provides players with visual information about line and flag objects around them so that players can approximately figure out their relative position in the pitch.

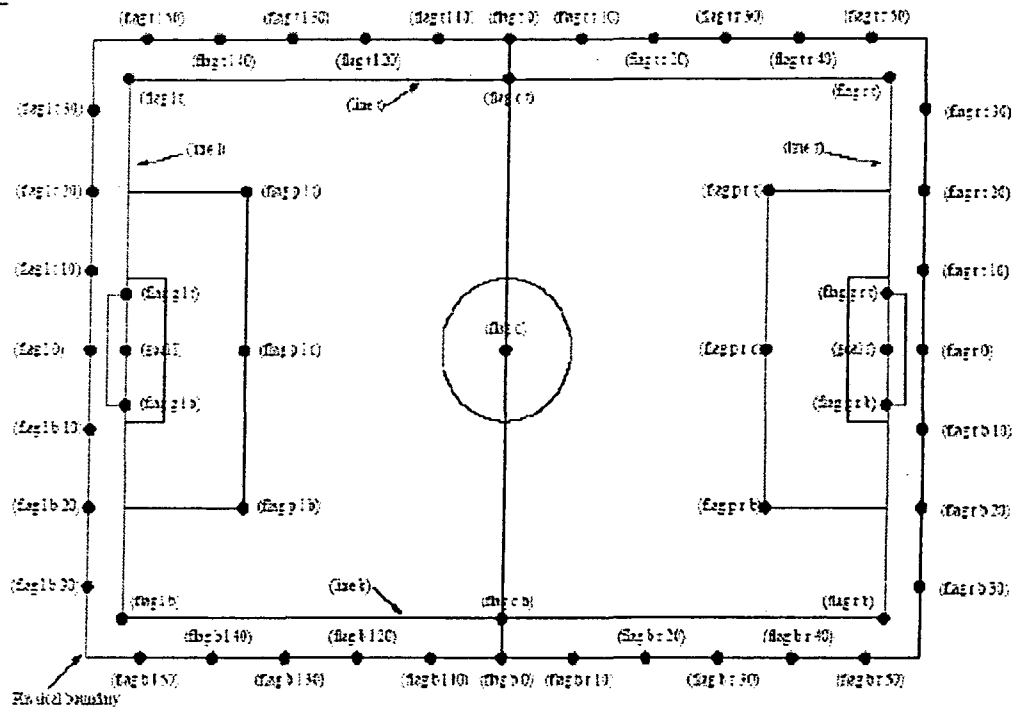


Fig. A2.1 Soccer pitch layout [Soccer Simulation Manual, 2003]

### b. Player's perception

Player's perception information comes from simulated sensors. There are 3 main types of sensor information: auditory, visual and body sensing. Players receive auditory information when communication between players happens. The two others are sent regularly to the players. Noise is added randomly in every simulation cycle that makes visual information noisy to the players.

Visual information arrives in terms of information about objects that a player currently sees. As default, visual information comes every 150 milliseconds. Visual objects can be stationary objects including lines and flags, as well as movable objects like other players. Around the player, visual ranges are defined in order to confine player's vision capability. The closer to a surrounding object a player is, the more detailed visual information it receives. Around a player's standpoint, vision area is divided into circular ranges, and narrowed by their view angle, which specifies how broadly it can see. It is depicted in Fig. A2.2 that player's vision is explained as:

1. Object *a* is in visible distance, but not in view angle. Therefore, only object type of *a* can be seen, such as a ball, a player, or a flag. No exact object name is given, such as(flag) but not (flag c).
2. Player has no visual information about object *b*, and *g* because they are either far from visible distance or out of view angle.
3. Object *c* and *d* are in *unum\_too\_far\_length* area, which is applied for player objects. If both are players, they are seen with their uniform number, but *d* with higher certainty whereas *c* with less certainty.

4. Object *e* is in *team\_too\_far\_length* area, which is applied for player objects. If *e* is a player, its team name is seen with some degree of certainty.
5. Object *f* is only seen as an anonymous player.

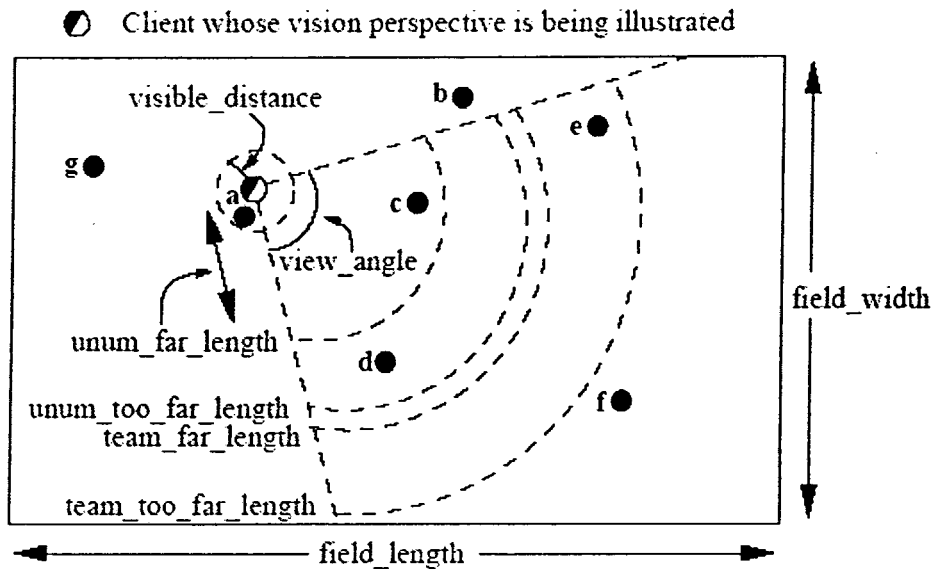


Figure A2.2 Vision area of a player. The player is shown as two-semi circles, the light circle is its body-facing direction. Objects are around it, from *a* to *f*. Player's vision parameters are *visible\_distance*, *view\_angle*, and so on.

Body sensor provides players with information about themselves, such as their speed, number of kicks, stamina, and number of turns they made, and so on. Body-sensing information is sent every 100 milliseconds as default.

### c. Player's action

Players should always act; otherwise, their team will lose advantage. When a player takes an action, it should send to the server a command corresponding to the action it takes. The command is text-formed. For example, a command like "(turn 50)" is to turn 50 degrees to the right relatively to its body direction. There are a number of commands which were predefined in soccer simulation. Body commands are for movement control like turn, dash, kick, and catch which are specific to player's movement. Communication commands are for players to talk and hear from their teammates. In each simulation cycle, only one movement command is accepted to execute. For example, if a player decides to kick the ball first and then run to the ball, kick and dash commands should be separated in two different cycles.

### A2.5 Coach

Coach is different from players in terms of privileges and therefore, different in actions. A coach is a client which has specific commands used to assist players. Coach connects

with the soccer server at other port than that of players (the default is 6002 for online coach and 6001 for trainer). There are two types of coaches: online coach and trainer (also called as offline coach). The basic difference of online coach and trainer is that they are created for different roles: the former is for the real game and the later is for training. Therefore, the commands available for online coach are more restricted than that of trainer.

#### **a. Online coach**

Online coach has the ability to know the global information about the match so that it can give strategic-level advices to players by communication. It knows exact position of every player and other objects in the pitch. It can change the role of players depending on how it wants the team to play, for example, defensive or offensive strategy. Therefore, online coach is useful for game analysis, and strategy adjustment.

Communication between online coach and players are restricted in order to prevent the overuse of centrally controlling every single player's action. The restriction is basically based on minimum time interval (the default is 300 milliseconds) and maximum messages a coach can send to players. In every interval, a coach can send only one message to players. As well, the number of messages is limited for each match.

#### **b. Trainer**

Trainer is similar to online coach. Like online coach, a trainer knows exactly the position of players and other objects on the pitch. The difference is that a trainer can execute training specific commands for controlling players more directly such as moving a player to a certain position. It is only appropriate for training, not for real games. Therefore, trainer is useful in developing players, which are equipped with certain machine learning methods. It helps automate training process.

## References:

- Tomohito Andou. *Refinement of soccer agents' positions using reinforcement learning*. In Hiroaki Kitano, editor, *RoboCup-97: Robot Soccer World Cup I*, pages 373{388. Springer Verlag, Berlin, 1998. (pp. 97, 167, 206).
- Sachiyo Arai, Kazuteru Miyazaki, and Shigenobu Kobayashi. *Generating cooperative behavior by multi-agent reinforcement learning*. In Sixth European Workshop on Learning Robots, Brighton, UK, August 1997. (p. 167).
- M. Asada (Ed.), *Proc. 2nd RoboCup Workshop*, The RoboCup Federation, 1998.
- Minoru Asada , Hiroaki Kitano , Itsuki Noda , Manuela Veloso *RoboCup: Today and tomorrow —What we have learned*, Artificial Intelligence ,110 (1999). pp. 193–214
- Philip R. Cohen, Hector J. Levesque, and Ira Smith., *On team formation*. In J. Hintikka and R. Tuomela, editors, *Contemporary Action Theory*. Synthese, 1999. (pp. 20,53, 55, 88, 89).
- G. Dudek, M. Jenkin, E. Miliotis, and D. Wilkes. "A Taxonomy for Multi-Agent Robotics". *Autonomous Robots*, 3(4):375-397, 1996.
- Barbara J. Grosz. *Collaborative systems*. *AI Magazine*, 17(2):67{85, Summer 1996. (pp. 20,53, 88).
- Robert H. Crites and Andrew G. Barto. *Improving elevator performance using reinforcement learning*. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Processing Systems 8*, Cambridge, MA, 1996. MIT Press. (pp. 166, 167).
- A. Farinelli, L. Iocchi, and D. Nardi, "An Analysis of Coordination in Multi-Robot Systems", *System, Man and Cybernetics* , 1487-1492 ,IEEE International Conference 2003.
- Fernando fernndez and Daniel Borrajo, "Applying Vector Quantization to reinforcement learning" , In *Robocup-99: Robot Soccer World Cup III*, Springer Verlag ,2000 .
- Foroughi, F. Heintz, S. Kapetanakis, K. Kostiadis, J. Kummeneje, I. Noda, O. Obst, P. Riley, and T. Steffens. *RoboCup Soccer Server User Manual: for Soccer Server version 7.07 and later*, 2001. URL: <http://sourceforge.net/projects/sserver>. Verified 7<sup>th</sup> of February 2004.
- H. Kitano, M. Tambe, P. Stone, M. Veloso, S. Coradeschi, E. Osawa, H. Matsubara, I. Noda, M. Asada, *The RoboCup synthetic agent challenge 97*, in: H. Kitano (Ed.), *RoboCup-97: Robot Soccer World Cup I*, Lecture Notes in Artificial Intelligence, Vol. 1395, Springer, Berlin, 1998, pp. 62–73.

- S. Konur, A. Ferrein, and G. Lakemeyer. *Learning decision trees for action selection in soccer agents*. In Proc. of Workshop on Agents in dynamic and real-time environments, 2004.
- A. Mackworth, *On seeing robots*, in: Computer Vision: Systems, Theory, and Applications, World Scientific Press, Singapore, 1993, pp. 1–13.
- P. Maes and Rodney A. Brooks, “*Learning to Coordinate Behaviors*”, in Proceeding, AAAI-90, Boston, MA, pp. 796-802, 1990.
- M. J. Mataric, “*Reinforcement Learning in the Multi-Robot Domain*”, Autonomous Robots, 4(1), Mar 1997, 73-83, 1997.
- Michael L. Littman. *Markov games as a framework for multi-agent reinforcement learning*. In Proceedings of the Eleventh International Conference on Machine Learning, pages 157{163, San Mateo, CA, 1994. Morgan Kaufman. (pp. 167, 188, 189, 199, 203, 206).
- M. Minsky, “*The Society of Mind*”, Heinemann, London, 1987
- Nakashima, T.; Takatani, M.; Udo, M.; Ishibuchi, H.; *An evolutionary approach for strategy learning in RoboCup soccer*, Systems, Man and Cybernetics, 2004 IEEE International Conference on Volume 2, 10-13 Oct. 2004 Page(s):2023 - 2028 vol.2
- Allen Newell and Herbert A. Simon. *Human Problem Solving*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1972. (pp. 20, 98).
- Norihiko Ono and Kenji Fukumoto. *A modular approach to multi-agent reinforcement learning*. In Gerhard Weiss, editor, Distributed Artificial Intelligence Meets Machine Learning, pages 25{39. Springer-Verlag, 1997. (p. 167).
- J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993. (pp. 93, 95, 116, 119, 122)
- A. L. Samuel. *Some studies in machine learning using the game of checkers*. IBM Journal of Research and Development, 3:211{229, 1959. Reprinted in E. A. Feigenbaum and J. Feldman, editors, Computers and Thought, McGraw-Hill, New York 1963. (p. 20).
- Jurgen Schmidhuber. *A general method for multi-agent reinforcement learning in unrestricted environments*. In Adaptation, Coevolution and Learning in Multiagent Systems: Papers from the 1996 AAAI Spring Symposium, pages 84{87, Menlo Park, CA, March 1996 AAAI Press. AAAI Technical Report SS-96-01. (pp. 184, 186).
- P. Stone and M. Veloso, “*A Layered Approach to Learning Client Behaviors in the Robocup Soccer Server*”, Applied Artificial Intelligence, 12: 165-188, 1998.



Peter Stone and Manuela Veloso. *Using decision tree confidence factors for multiagent control*. In Proceedings of the Second International Conference on Autonomous Agents, 1998. (p. 125)

Peter Stone and Manuela Veloso. *Team-partitioned, opaque-transition reinforcement learning*. In Minoru Asada and Hiroaki Kitano, editors, RoboCup-98: Robot Soccer World Cup II. Springer Verlag, Berlin, 1999. Also in Proceedings of the Third International Conference on Autonomous Agents, 1999. (p. 135)

K. Sycara, K. Decker, A. Pannu, M. Williamson, and D. Zeng. *Distributed intelligent agents*. IEEE Expert, December 1996. (pp. 20, 195, 198).

K.Sycara. "Multiagent systems". AI magazine 19(2):79-92, Summer 1998.

Ming Tan. *Multi-agent reinforcement learning: Independent vs. cooperative agents*. In Proceedings of the Tenth International Conference on Machine Learning, pages 330-337, 1993. (pp. 167, 195, 196, 199).

Gerald Tesauro. *TD-Gammon, a self-teaching backgammon program, achieves master-level play*. Neural Computation, 6(2):215-219, 1994. (pp. 139, 167).

Christopher J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK, 1989. (pp. 93, 138)

Michael Wooldridge "An Introduction to MultiAgent Systems", John Wiley & Sons, 2002.

