



An Approach to Enhance
Reinforcement Learning in Agent

by

Mr. Thanadee Ujjubandh

Submitted in Partial Fulfillment of
the Requirements for the Degree of
Master of Science
in Computer Science
Assumption University

March, 2002

An Approach to Enhance Reinforcement Learning in Agent

By

Mr. Thanadee Ujjubandh

The logo of Assumption University of Thailand is a circular emblem. It features a central shield divided into four quadrants: top-left (blue with a white star), top-right (white with a blue cross), bottom-left (white with a blue star), and bottom-right (red with a white cross). The shield is flanked by golden laurel branches. Above the shield is a golden crown. Below the shield is a golden banner with the Latin motto "LABOR OMNIA VINCIT". The entire emblem is encircled by the text "ASSUMPTION UNIVERSITY OF THAILAND" in a semi-circle.

**Submitted in Partial Fulfillment of the
Requirements for the Degree of**

Master of Science

In Computer Science

Assumption University

March, 2002

The Faculty of Science and Technology


Thesis Approval


Thesis Title An Approach to Enhance Reinforcement Learning in Agent


By Mr. Thanadee Ujjubandh
Thesis Advisor Dr. Jirapun Daengdej
Academic Year 2/2001


The Department of Computer Science, Faculty of Science and Technology of Assumption University has approved this final report of the **twelve** credits course. **SC7000 Master Thesis**, submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science.

Approval Committee:

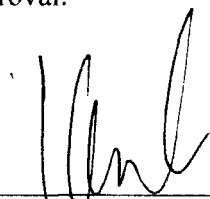

(Dr. Jirapun Daengdej)
Advisor



(Asst. Prof. Dr. Thotsapon Sortrakul)
Committee Member


(Dr. Thitipong Tanprasert)
Committee Member


(Asst. Prof. Dr. Surapong Auwatanamongkol)
Representative of Ministry of
University Affairs

Faculty Approval:


(Asst. Prof. Dr. Tang Van To)
Director


(Asst. Prof. Dr. Pratit Santiprabhob)
Dean

March / 2002

ACKNOWLEDGEMENTS

I would like to express my most sincere appreciation and thanks to Dr. Jirapun Daengdej, my thesis advisor for many valuable advice and encouragement throughout the thesis. If without his great support, I cannot get to this point.

I also thank Dr. Pham Hong Hanh, for her valuable guidance and comments. I would like to thank her for teaching me the art of managing project. And also thanks to Mr. Worasing Rinsurongkawong, who is the best project partner.

Finally, I thank my family for their encouragement and support during my MSCS study period.



ABSTRACT

Reinforcement learning addresses the problem of how autonomous agent gathers information and performs action in its environment to achieve its goal. Q learning is a kind of reinforcement learning that can acquire optimal control strategies form delayed reward.

This thesis proposes a learning technique based on separating long-term goal to small subtasks, using function approximator to generalize the similar state and transferring knowledge between agents for reducing the exploration environment of Q learning. This technique will be simulated in soccer games. The environment in game is 3 dimensions, uncertainty environment and the agent will get only partial information from the sensor.

As a result, by using the technique, the autonomous agent will reduce time to explore the environment. However, subtasks were divided and they may or may not be the optimal subtasks.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	i
ABSTRACT	ii
LIST OF FIGURES	v
LIST OF TABLES	vi
CHAPTER 1: INTRODUCTION	1
1.1 Introduction	1
1.2 Objective and Scope	1
CHAPTER 2: THEORETICAL BACKGROUND	3
2.1 Reinforcement learning	3
2.2 Model of reinforcement learning	3
2.3 Q learning	5
2.4 Issue in reinforcement Learning	7
2.5 JAVA	8
2.6 Neural network	9
CHAPTER 3: RELATED WORKS	11
3.1 Approaches to speed up the Q learning	11
CHAPTER 4: THE PROPOSED TECHNIQUE	13
4.1 Environment	13
4.2 Training Process	16
CHAPTER 5: EXPERIMENTATION AND EVALUATION	20
5.1 Rewarding Mechanism	20
5.2 Transferring Knowledge Mechanism	23

5.3 Training a neural network	24
CHAPTER 6: DRAWBACK AND FUTURE RESEARCH	27
CHAPTER 7: CONCLUSION	29
BIBLIOGRAPHY	31



LIST OF FIGURES

Figure 2-1	The standard model of Reinforcement Learning	3
Figure 2-2	Interaction between agent and environment	4
Figure 2-3	Explanation of Q-learning Algorithm	6
Figure 2-4:	Explanation of Backpropagation Algorithm	10
Figure 4-1	Soccer Field	13
Figure 4-2	The screen shot of Soccer Manager game	16
Figure 4-3	A data collected from reinforcement learning	17
Figure 4-4	A Training Data	19
Figure 4-5	Training neural network process	19
Figure 5-1	Assigning rewards algorithm	23
Figure 5-2	Transfer knowledge between agent	23
Figure 5-3	Training neural network process	24
Figure 5-4	Training neural network process	25
Figure 5-5	The number of the knowledge which the agent derives from Q-learning	26

LIST OF TABLES

Table 4-1	Soccer field area description	14
Table 4-2	Soccer player’s skills	14
Table 4-3	Soccer player’s action ID	15
Table 4-4	Input Node	17



Chapter 1

Introduction

1.1 Introduction

A goal of artificial intelligence is a creation technique for constructing autonomous agent that can perform good performance in the real world [1]. Reinforcement Learning is a kind of learning technique in artificial intelligence. It has been researched in many areas including game theory, control theory and robotics. Reinforcement learning is a way of programming agents by reward and punishment without need to specify how the task is to be achieved [6]. The characteristics of reinforcement learning is a learning behavior through trial-and-error interactions with a dynamic environment. In regard to reinforcement learning, many systems use the delayed rewards technique such as Q-learning, to achieve optimal control strategy [8]. Unfortunately, delayed rewards have a weak point that is *“if the agent cannot reach the goal, it cannot learn anything”*. This thesis focuses on how to build the multi-agent system which works cooperatively and based on Q-learning, to reduce exploring steps in the large, dynamic and uncertainty state space by separating long-term goal to subtasks and using neural networks as a function approximator.

1.2 Objective and Scope

This thesis discusses in issue of multiple goals Q-learning to make the autonomous agent, learn from interacting with the environment. By using this technique, the autonomous agents can easily achieve their sub-goals, which result in decreasing steps in training. The proposed technique is demonstrated using a soccer

game, which is considered as one of difficult problems domain, since each of the player has to deal with dynamically changing and uncertain environment.

The environment in this game will be represented in 3D workspace. With 3D workspace, the ball in this game can move in the air, however it cannot go out of the soccer field's boundaries. Once the ball hits the field's boundaries, it will bounce back into the field. All soccer robots have to learn from their environment and adapt themselves in order to achieve their goals. They have to understand all important areas in the soccer field. However, there are some constraints on this thesis:

- ❖ This is an offline training.
- ❖ Goalkeeper will not be trained.
- ❖ All autonomous agents will not communicate with each other.
- ❖ In regard to detecting of all agents, it can perceive only in partial information.
- ❖ The environment is not certain (the same action in the same stage may lead to the different result and different reward).

Chapter 2

Theoretical Background

2.1 Reinforcement Learning

Reinforcement Learning is the learning of mapping from situations to actions and to maximize a scalar reward or reinforcement signal. The learner does not need to be told directly to take which actions, but it must discover which actions yield the most reward by trying them. All reinforcement learning requires a particular combination of search and memory [13]. Search is required to find good actions, and memory is required to remember what actions worked well in which situations in the past.

Reinforcement learning is different from supervised learning because the agent is not told which is the best action. Instead, after choosing an action, the agent is told how well it performed.

2.2 Model of Reinforcement Learning

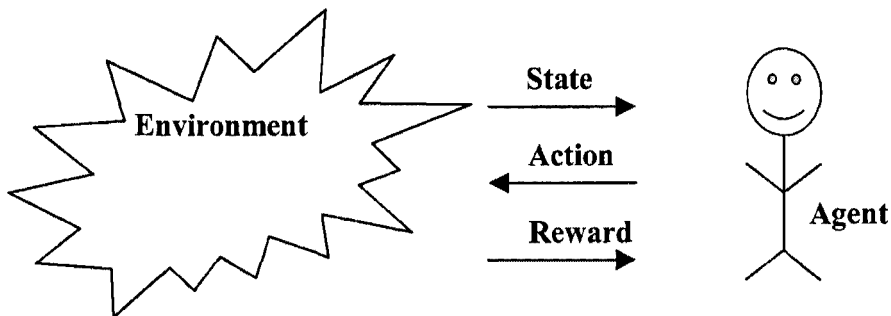


Figure 2-1: The standard model of Reinforcement Learning

In the standard reinforcement-learning model, an agent is connected to its environment via perception and action, as shown in Figure 3-1. On each step of interaction, first the agent receives current state and then it selects action, finally it receives reward [6].

Formally, the model consists of

- ❖ A discrete set of environment states S ;
- ❖ A discrete set of actions A ;
- ❖ A set of scalar reinforcement signals R ;

The following dialog is an example of relationship between environment and the agent.

Environment: You are in state 10. You have 21 possible actions.

Agent: I will perform action 4.

Environment: You have received a reinforcement of 7 units.

Environment: Now you are in state 50. You have 21 possible action

Agent: I will perform action 16

Environment: You have received a reinforcement of 5 units.

Environment: You are in state 3 and so on.

$$S_0 \xrightarrow{a_0, r_0} S_1 \xrightarrow{a_1, r_1} \dots$$

Figure 2-2: Interaction between agent and environment

The agent's job is to find out a policy π for mapping states to actions that maximize some long-run measure of reinforcement. The environment can be non-

deterministic, which takes the same action in the same state and in different situations, may receive different result or reward. However, the dealing with the non-deterministic environment, the agent applies a probabilistic to justify which policy should be selected.

2.3 Q-learning

The reward function may provide some scalar value for the states where the task has been achieved, so the agent must learn how to accomplish its task through the reward function. To accomplish the task, the agent gets the reward, called the Q-value of executing an action from a given state. The Q-value, $Q(s, a)$ is defined as the expected discounted reward of executing action a , from state s , and the following the policy π . A policy is a function that determines which action to execute in any given state.

Q-learning is designed to maximize expected future discounted reward. This quantity is called in return, and is defined as

$$r(t) = \sum_{i=0}^{\infty} \gamma^i R_{t+i}$$

R_{t+i} is the immediate reward received at time $t+i$ and γ is a discount factor with magnitude between 0 and 1. The Q-learning attempts to learn a policy that gathers the highest reward. After learning process, the policy is defined to execute the action with the highest Q-value in the current state.

Initialize the table entry $Q(s, a)$ to zero

Repeat forever

```
{  
  Observe the current state  $s$   
  Select action  $a$  and execute it  
  IF goal is reached  
  {  
    Receive immediate reward  $r$   
    Observe the new state  $s'$   
    Update sequences of episodes of table  
    entry  $Q(s, a)$  by this formula  
    
$$Q(s, a) \leftarrow r + \gamma \max_{a'} Q(s', a')$$
  
  }  
}
```

Figure 2-3: Explanation of Q-learning Algorithm

According to algorithm shown in Figure 3, the Q table can be initially referred to random number or zero. The agent repeats observe current state and perform action until it reaches the goal. Then Q-value has to calculate in the current state and propagate back to the last sequence of states. In Figure 3 is Q-learning algorithm for deterministic environment. If the environment is non-deterministic, the training rule

$$Q(s, a) \leftarrow r + \max_{a'} Q(s', a')$$

has to modify to [8]

$$Q(s, a) \leftarrow (1 - \alpha_n)Q_{n-1}(s, a) + \alpha_n[r + \max_{a'} Q_{n-1}(s', a')]$$

$$\alpha_n = \frac{1}{1 + \text{visits}_n(s, a)}$$

When the agent has to select the action from its knowledge, it looks at the Q table and then chooses the action that has maximized the Q-value.

In an infinite number of time steps, Q-learning can be shown to converge to an optimal policy [15]. Anyway, without exploration, the agent would stay with the first solution found; even though there are many ways to improve the way to reach the goal (optimal policy). The agent must search for an infinite number of time steps in order to provably find the optimal solution. Clearly this is not possible in practical applications. It is possible, in the agent that limits its exploration and learning does not find an optimal solution. In this case, the agent's sub-optimal solution can be used to achieve the goal, but it might lose efficiency.

Q-learning is an attractive algorithm, because it requires only little prior knowledge of the domain. However, as the agent's task and domain increases, the learning time also has a lot of increasing.

2.4 Issue in reinforcement learning [8, 7]

- ❖ **Delayed rewards:** In reinforcement learning the trainer will provide only a sequence of immediate reward values (not explicitly training information to the agent) as the agent executes its sequence of action. Therefore, the agent faces the problem of temporal credit assignment. Consequently, back propagating of the reward values can reduce efficiency of a system for a long-term goal.
- ❖ **Exploration:** In reinforcement learning, after the agent performs a sequence of actions and a goal has been reached, it will get the training data only. As a

result, the agent faces a tradeoff in choosing whether to favor exploration of unknown states and actions (to gather new information), or exploitation of states and action that it has already learned in order to maximize its cumulative reward.

- ❖ **Partially observable states:** In the real world problem, the agent can get only some part of the whole environment, and it has to perform action with partial information. It may be necessary for the agent to consider its previous observation in contrast with the current observation in order to choose an action. While this kind of problem can occur in any type of intelligent system, the problem also requires attention when using the reinforcement learning technique.

2.5 JAVA

Java is a kind of programming language that is used to implement the proposed technique. Because the concept of platform is independent, it can run any platform after finishing compilation only once. Java also supports both stand-alone computer (application) and small program that can transfer and run on browser via the network (applet).

Java has all properties of object-oriented paradigm; encapsulation, polymorphism, dynamics binding and inheritance. This paradigm is easy for code reusable. If there is some addition to the system later, the additional parts can be added with very little effect to overall system.

Java has an exception handling mechanism to deal with the abnormal situation. Java has an automatic garbage collector, which runs as a low priority thread.

The garbage collector will de-allocate memory automatically not like C++ that the programmer has to de-allocate memory by himself.

2.6 Neural Network

To train neural network will use standard back propagation to adjust the weights, which has the following sigmoid function [5]

$$f(x) = \frac{1}{1 + e^{-x}}$$

$$f'(x) = f(x)[1 - f(x)]$$

This sigmoid function has range of (0, 1).

❖ Back propagation Algorithm

- Create neural network
- Select learning rate and error that we want to stop training
- Random all weight in each node.

Step 1: Feed forward

$$Z_{in} = W_0 + \sum_{i=1}^n x_i W_i$$

$$Z = f(Z_{in})$$

Z_{in} = input of sigmoid function

Z = output of sigmoid function

W = weight

X = input

Step 2: Back-propagation

For each output node

$$\delta = (t - y)f'(z_{in})$$

t = target

y = output from output node for each hidden node

δ = Portion of error correction weight adjustment

$$\delta_{in} = \sum_{k=1}^m \delta_k W_k$$

$$\delta = \delta_{in} f'(z_{in})$$

Then for each node

$$\Delta Bias = \alpha \delta$$

$$\Delta W = \alpha \delta x_i$$

α = Learning Rate

Step3: Adjust weight and bias for each node

$$W_{(new)} = W_{(old)} + \Delta W$$

$$Bias_{(new)} = Bias_{(old)} + \Delta Bias$$

Repeat Step 1 to Step 3 until stop condition [5].

Figure 2-4: Explanation of Backpropagation Algorithm

Neural networks will stop calculating the weight when absolute mean error is less than 0.15. At this point, the neural network has been created and will be used as a thinking engine of the soccer player

Chapter 3

Related Works

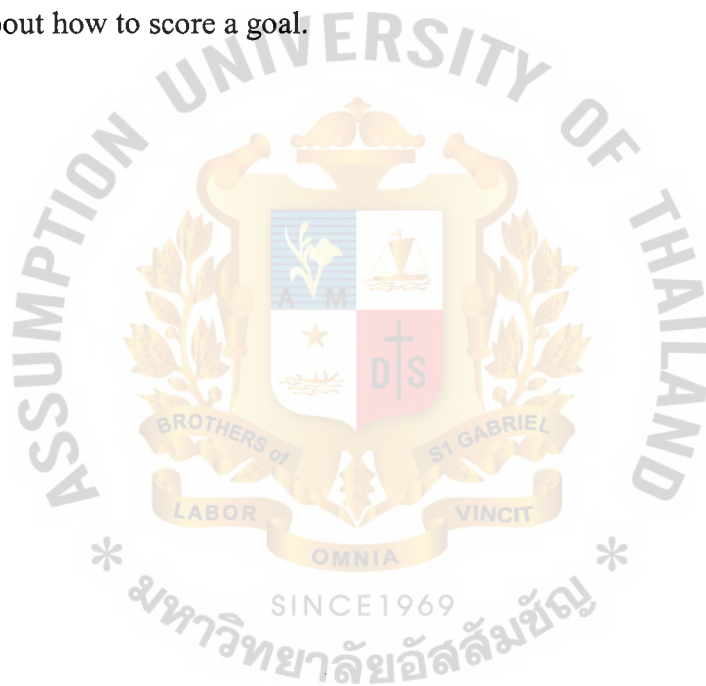
The agent learns from trial and error and delay rewards. When the state space is large the agent has to trial many times until reach the goal-state, so if the agent cannot reach the goal-state, it will not learn anything or it may perform a lot of poor action before it reaches goal-state.

3.1 Approaches to speed up the Q-learning: In general, there are four approaches to enhance Q-learning techniques in multi-agent system.

Transfer Knowledge: Tan [14] showed some knowledge that has been transferred between agents, can reduce training step by providing three kinds of share knowledge, which are share sensation, share policies or episodes and passive share sensation. All techniques showed that they can reduce number of trials compared to no sharing information.

Applying function approximator: Noda, Matsubara, Hiraki, Frank [9] proposed the way to deal with the large state space by using function approximator, for instance, neural network and decision tree (C4.5) to generate rule for mapping which situations belong to which actions. M. Wiering, R. Salustowicz, J. Schmidhuber [11, 16] propose Probabilistic Incremental Program Evolution (PIPE) by combining Q-learning with linear neural network, but they have a problem of good policies that do not stabilize but tend to get destroyed by subsequent “unlucky” experience.

Decomposition of goal techniques: This technique helps RL to converge faster by dividing global goal to sub goal [3, 4, 10]. Dayan and Hinton [2] have proposed Feudal RL that divides single task up into subtasks at multiple level. Each level has a manager and local goal. The manager sets the goal of sub-manager then sub-manager applies Q-learning to find the best way to perform task. Kostiadis and Hu [7], introduced immediate goal that is dividing global goal (win the game) to local depending on the agent's role. So the agent only needs to achieve a local goal such as, a goalkeeper does not need to worry about how to score a goal.



CHAPTER 4

The Proposed Technique

This thesis proposes the technique that combines between transfer knowledge, function approximator and decomposition of goals together to minimize the learning step of Q-learning.

4.1 Environment

Soccer game is a good test based for multi-agent system, cooperate and artificial intelligence [13]. This thesis uses Soccer Manager game to simulate the proposed technique. The size of the soccer field is scaled from 110m \times 75m. The environment in Soccer Manager game is represented similar to the real world by applying physics' law such as gravity, force, friction, momentum, and projectile movement. All soccer robots have to learn from their environment and adapt themselves.



Figure 4-1: Soccer Field

According to Figure 2, the following numbers explain the soccer field's area

Zone Number	Description
1, 3, 7, 9	Centering zone
2, 8	Penalty box zone
2, 4, 8, 10	Long shot zone
5, 15, 18	Attacking zone
11, 17, 19	Defensive zone
12	Left side zone
13	Center zone
14.	Right side zone
20, 21	Goal zone

Table 4-1: Soccer field area description

Each soccer team contains 11 soccer players; each player in the simulator has different skills and roles. Each soccer player has individual 23 different skills as shown in Table 2:

Skill Number	Skill Name	Description	Attribute Value
1	Accept Ball	Accept ball ability	1-20
2	Aggressive	How hard he plays when he tries to win the ball	1-20
3	Attacking	How well he can perform when he plays attack	1-20
4	Balance	Body balancing	1-20
5	Crossing	Crossing ball accuracy	1-20
6	Curve	Free kick and shooting ability	1-20
7	Defensive	How well he can perform when he plays defense	1-20
8	Dribble	Dribble ability	1-20
9	Heading	Heading accuracy	1-20
10	Long Shot	Long shot Ability	1-20
11	Marking	Man to man defense ability	1-20
12	Jumping	How good he is in the air	1-20
13	Pace	Running speed	1-20
14	Passing	Passing accuracy	1-20
15	Respond	How quick he can react to the current situation (making decision)	1-20
16	Shoot Accuracy	Shooting accuracy	1-20
17	Shoot Power	Shooting power	1-20

18	Sliding	Sliding ability	1-20
19	Strength	Body strength (Pushing ability)	1-20
20	Tackle	Tackle ability	1-20
21	Technique	Technique	1-20
22	Flair	Vision of the player	1-20
23	Consistence	Consistency of the player	1-20

Table 4-2: Soccer player's skills

The roles of each player depends on player position. The position can be classified into four types, which are:

- ❖ Attacker
- ❖ Midfielder
- ❖ Defender
- ❖ Goalkeeper

The soccer player has to select the action to perform from five types of actions, which are passing, shooting, moving, intercepting and combination actions.

Table 2 shows the list of player's action ID.

Action ID	Action Type	Description	With ball
1	Defensive Move	Turn to ball	No
2	Defensive Move	Blocking shoot	No
3	Defensive Move	Play zone defense	No
4	Defensive Move	Play man to man defense	No
5	Defensive Move	Marking opponent player	No
6	Defensive Move	Play defense type 1	No
7	Defensive Move	Play defense type 2	No
8	Defensive Move	Play defense type 3	No
9	Attacking Move	Chase ball	Yes/No
10	Attacking Move	Finding space (Overlap run)	Yes/No
11	Attacking Move	Play attack type 1	Yes/No
12	Attacking Move	Play attack type 2	Yes/No
13	Attacking Move	Play attack type 3	Yes/No
14	Attacking Move	Dribble	Yes
15	Passing	Long pass	Yes
16	Passing	Short pass	Yes

17	Passing	Through pass	Yes
18	Passing	Centering	Yes
19	Passing	Heading	Yes/No
20	Shooting	Normal shoot	Yes
21	Shooting	Long shoot	Yes
22	Shooting	Loop shoot	Yes
23	Shooting	Volley shoot	Yes/No
24	Shooting	Heading	Yes/No
25	Shooting	Over head kick	Yes/No
26	Interception	Tackle	No
27	Interception	Sliding	No
28	Combination	Turning	Yes/No

Table 4-3: Soccer player’s action ID

Each soccer player has his own sensor to gather the picture of the environment such as soccer field, teammates’ position, opponents’ position, ball’s position, and this sensor can get only a partial of the environment. Soccer player has own thinking engine to make decision, then select the action from Table3 to react to the change of the environment.

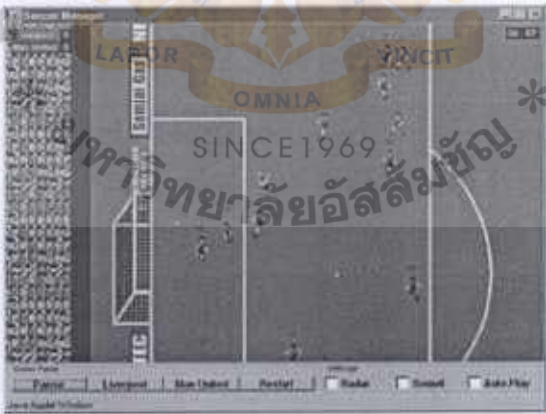


Figure 4-2: The screen shot of Soccer Manager game

4.2 Training Process

In training process, first the agent in the same team must share the global goal, which wins the game. Then, decomposes global goal to local goal based on the role of each player. The example of the local goal is, the forward tries to score and the midfielder tries to pass the ball to the forward when their team has the ball and the defender tries to clear the ball.

Next, let the soccer player play with another team for collecting the training data. The soccer player gets training data by using Q-learning. A training data records the states and situation of soccer player at the moment when he gets reward. A training data composes of the following attributes:

- ❖ Input Nodes: The attributes that describe the state and situation of the soccer player including positions, teammates, and opponent status.
- ❖ Action ID: The ID of action that soccer player chooses to perform
- ❖ Frequency: The number of duplicate training data ID.
- ❖ Average Reward: The average score is used for justify that result of each action

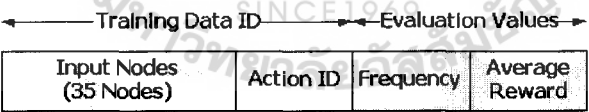


Figure 4-3: A data collected from reinforcement learning

After time out, the agents share their own knowledge with each other. The policy in shearing information the agent can share with the agent that has the same role.

Finally, soccer player uses his knowledge to construct training set and trains neural networks. Before training neural network, soccer player has to build the

training set by filtering all the training data that was collected by using Q-learning.

Table 4-4 shows the meaning of input node.

Node	Type	Description of each soccer player's input data
1	Team	Team1 has ball?
2	Team	Team2 has ball?
3	Player	I have ball?
4	Player	Close to ball?
5	Player	Is ball around?
6	Player	Is there any obstruction?
7	Ball	Ball in sensor 1?
8	Ball	Ball in sensor 2?
9	Ball	Ball in sensor 3?
10	Player	Opponent in sensor1?
11	Player	Opponent in sensor2?
12	Player	Opponent in sensor3?
13	Field	I'm in team1 penalty area?
14	Field	I'm in team2 penalty area?
15	Field	I'm in team1 long-shoot area?
16	Field	I'm in team2 long-shoot area?
17	Field	I'm in left/right wing area?
18	Field	I'm in centering area?
19	Field	I'm In center area?
20	Field	I'm in defensive area?
21	Field	I'm in midfield area?
22	Field	I'm in attacking area?
23	Ball	Is ball in defensive area?
24	Ball	Is ball in midfield area?
25	Ball	Is ball in attacking area?
26	Player	Can pass?
27	Player	Through pass?
28	Player	Can shoot?
29	Player	Is in shooting range?
30	Player	Clear for shoot?
31	Team	My teammate in opponent area?
32	Player	I'm a goalkeeper?
33	Player	I'm a defender?
34	Player	I'm a midfield?
35	Player	I'm an attacker?

Table 4-4: Input Node

To construct the training set, the training data that has highest average reward will be selected. This is the filter process:

1. Merge all training data for the player that has the same role.

- 2. Sort all training data
- 3. Search for all training data that have duplicate, then merge them
- 4. Select all training data that have unique value in input nodes field. In the case where there are many training data that have the same value in input node field, then select the training data that has highest average reward. If there are still in the same values in input nodes, and average reward, then select the training data that has highest frequency

35 Input Nodes	Action ID
35 Input Nodes	Action ID
...	
35 Input Nodes	Action ID

Figure 4-4: A training data

- 5. Building the training set from all training data that have been selected in step3

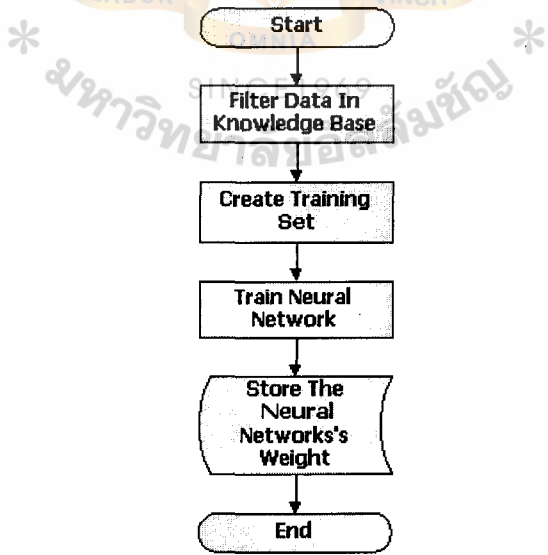


Figure 4-5: Training neural network process

After that soccer player use that training set as input to train neural networks, then calculate the neural networks' weights and use this weight to choose action.



CHAPTER 5

Experiment and Evaluation

This thesis uses soccer game to evaluate the learning algorithm. In learning process, first the agent in team has no knowledge. They choose every action at random. After that, if the agent can reach the goal state, it will receive a reward and then collect the state, action and reward in memory as knowledge.

5.1 Rewarding Mechanism

This subsection explains how award is given to each action occurred in the game. The reward signal is ranged in 0 to 100. The reward is just a scalar number and it can change to other reward such as ranged in 0 to 1000 or -1000 to 100. The effect in changing range of reward is the response time of the learning system. For example, the agent performs bad action, it will get the punishment of -1000, this action may be a good action in normal situation. So that, next time when the agent wants to select action in this state, perhaps it chooses another action (which may not be the best action). The strong punishment reduces the average reward for this action. If the punishment is too strong, the agent has to explore the state more for choosing that action again. Because the environment is uncertainty, the reward is collected in average reward. Therefore, this thesis selects to use the range of reward from 0 to 100. The goal state is set by depending on the player's role. The following describes algorithm used for assigning rewards.

If team wins, then the agent will receive a reward of 100 {

If team has ball and the agent is a forward {

If the agent scores ,then the agent will receive a reward of 100.

If the agent can get the ball into the penalty area of opponent, then the agent will receive a reward of 90.

If the agent passes the ball successfully, then the agent will receive a reward of 80.

If the agent scores own goal or lose the ball, then the agent will receive a reward of 0.

}

If team has ball and the agent is a midfielder {

If the agent scores ,then the agent will receive a reward of 100

If the agent can get the ball into the penalty area of opponent, then the agent will receive a reward of 80

If the agent passes the ball successfully, then the agent will receive a reward of 90.

If the agent scores own goal or lose the ball, then the agent will receive a reward of 0.

}

If team has ball and the agent is a defender {

If the agent scores ,then the agent will receive a reward of 90

If the agent passes the ball successfully, then the agent will receive a reward of 90.

If the agent scores own goal or lose the ball, then the agent will receive a reward of 0.

}

If team does not have ball and the agent is a forward {

If the agent can get the ball , then the agent will receive a reward of 100.

If the agent goes into the right zone, then the agent will receive a reward of 90.

}

If team does not have ball and the agent is a midfielder {

If the agent can get the ball, then the agent will receive a reward of 100.

If the agent goes into the right zone, then the agent will receive a reward of 90.

```

    }

    If team does not have ball and the agent is
    a defender {

        If the agent can get the ball, then the
        agent will receive a reward of 100.

        If the agent goes into the right zone,
        then the agent will receive a reward of
        90.

    }

}

```

Figure 5-1: Assigning rewards algorithm

5.2 Transferring Knowledge Mechanism

Transferring Knowledge means the using knowledge of the other agent experience. The transferring of this thesis is the transferring between the agents who have the same role. For example, the group of midfielder will transfer knowledge among them which occurs after the game.



Figure 5-2: Transfer knowledge between agent

5.3 Training a neural network

This thesis uses the neural network as a tool for selecting action, it does not concern the way to optimize the neural network.

The knowledge from reinforcement learning is the training set of neural network. The knowledge is filtered to the training data of neural network. Finally, the training data is used for training neural network.

Each agent uses neural network as a thinking engine to map which situation belongs to which action. In training neural network, this thesis uses standard back-propagation algorithm, which composes of 35 input nodes, 35 hidden nodes and 28 output nodes. The meaning of input and output node is shown in figure 4-3 and 4-4.

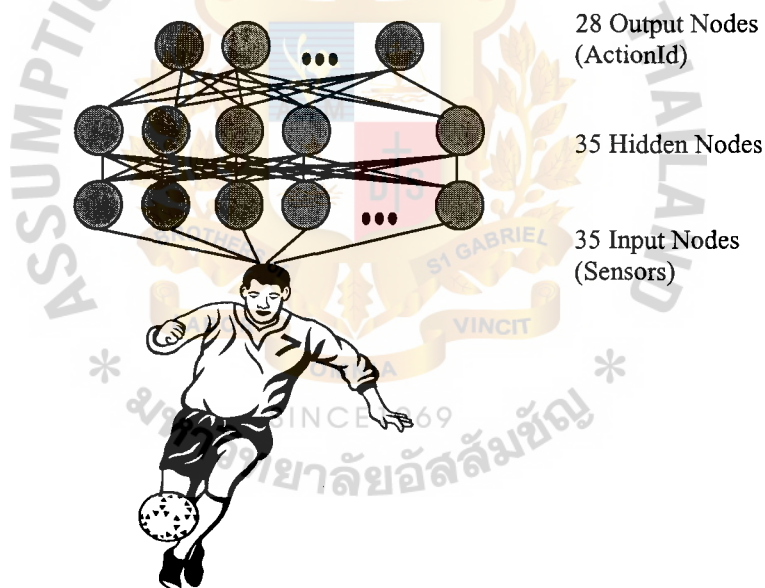


Figure 5-3: Training neural network process

The training set of neural network is the data that is learnt from Reinforcement learning. Table 4-4 describes the definition of each input node.

To estimate the performance of learning technique in this thesis, the proposed technique is used in the implementation to compare with: decomposition of goals

technique and transfer knowledge technique. The three approaches are compared with rule-based system. Note that, all rules are used by the rule-based team which are constructed using domain experts, and the system is not adaptable.

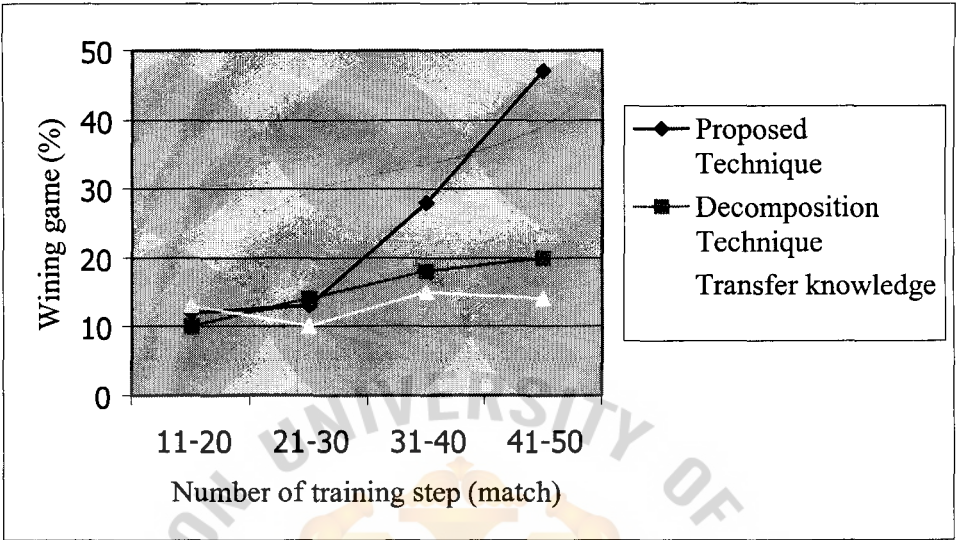


Figure 5-4: Training neural network process

From figure 5-1 the graph of proposed technique shows that the proposed technique can learn to win the rule-based system faster than other two systems. Each ten matches the agent are trained and the result are collected from playing 100 matches.

The next figure shows, the number of the training data, which the agent gets from apply Q-learning. The result shows that when the agent uses the proposed technique, it can reach more goals and know more states than the other 2 techniques.

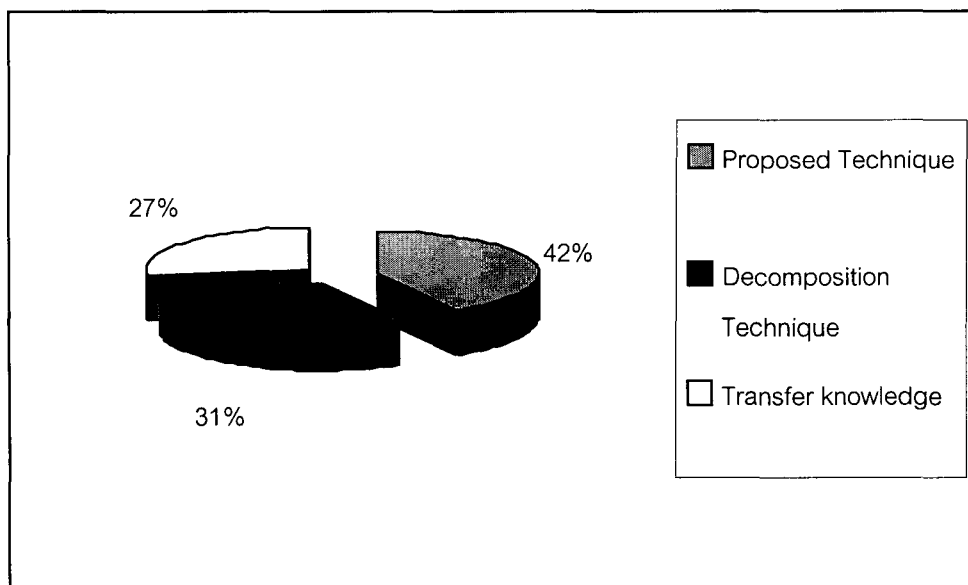


Figure 5-5: The number of the knowledge which the agent derives from Q-learning

When the agent has more knowledge, it means in the same period of time the agent can learn much more than other two approaches.

Note that, after the neural network is trained by the set of input which is generated from reinforcement learning. After the weight was set, the agent will select action by feeding input to the neural network. Characteristics in making decision of neural network is a black box, you know what is the result of neural network's decision but no one knows why neural network made decision like that. The result of making decision of neural network is based on the training set but differs from C4.5 which generates rules explicitly so everyone knows why it made decision like that.

CHAPTER 6

Drawback and Future Research

There are some drawbacks in this proposed technique.

- ❖ Normally, one special characteristics of reinforcement learning is online training, that means the agent learns from interacting with the environment immediately. But when neural network was applied to the system, the agent cannot learn and perform action in real-time.
- ❖ In decomposing process, the domain expert is required to set the sub-goal to reduce the sequence of blind search. This technique cannot guarantee sub-goal that was divided is the sub-optimal goal or not.
- ❖ Time consumed in training neural network, if the system has more training data, the training time will also increase.
- ❖ The system can react to the change of the environment slower than the pure reinforcement learning. The major adaptation of this framework is the training process of neural network.

The experiment in this thesis based on the situation of the agent, has to learn from large and uncertain state space. This thesis proposes in the framework to reduce the searching step. The proposed framework can be further researched on changing function approximator, applying other decomposition techniques or using other transfer knowledge techniques to yield the minimum searching steps.



CHAPTER 7

Conclusion

Reinforcement Learning is a kind of machine learning, which learns from interaction between agent and environment. When the agent reaches the goal state, the agent will get the reward back to each sequence of actions. Q-learning is a kind of reinforcement learning which can guarantee the optimal solution. Reinforcement learning can learn in real-time by using searching and memory management techniques.

To cooperate between multi-agent systems, the agent in the same team must share global goal, which is win the game [12]. In the real-world problem, state space is very large and some problems need to be solved in real-time. This is the major problem in reinforcement learning. Suppose the agent has to search into large state space,

The first problem, the agent must use a large memory to store the last sequence of action (which action is good or bad). Next problem, sometimes in the agent's lifetime, he has never reached the goal state, especially when the goal is a long-term goal, that means the agent cannot learn anything. Another problem is indexing problem, when the data is huge, they must use special data-structure or special technique, if the agent has to perform action in real time.

This thesis proposes in a framework to deal with the problem above by combining three approaches which are transferring knowledge, applying function approximator and decomposition of goal techniques. The Transfer knowledge is used to reduce the repeat search. Neural network is used in another reason, which reduces

the searching time when the agent makes decision. Because, after the neural network was trained, the neural's weight is set. Then, when the agent feeds input to the neural network, the time that is consumed in processing is $O(1)$. Decomposition technique is used to split long-term goal into intermediate goal for helping the agent to reach the goal more frequently.



Bibliography

- 1 Boutilier C., Y. Shoham and M. Wellman, Economic principles of multi-agent systems, Artificial Intelligence, 1997.
- 2 Dayan P., G. E. Hinton, Feudal Reinforcement Learning, Proceedings of the IEEE Conference in Denver, 1993.
- 3 Dean T., and S. H. Lin, Decomposition Techniques for Planning in Stochastic Domains, In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, pages 1121--1127, Montreal, 1995.
- 4 Dietterich T., The MAXQ Method for Hierarchical Reinforcement Learning, In Proceedings of the Fifteenth International Conference on Machine Learning. Morgan Kaufmann, 1998.
- 5 Kaelbling L.P., A W. Moore, Reinforcement Learning: A Survey, Journal of Artificial Intelligence Research, 1996.
- 6 Kostiadis K., and H. Hu, Reinforcement Learning and Co-operation in a Simulated Multi-agent System, IROS'99, 1999.
- 7 Mitchell T.M., Machine Learning, McGraw-Hill, 1997.
- 8 Noda I., H. Matsubara, K. Hiraki, I. Frank, Soccer Server a tool for research on multi-agent systems, Applied Artificial Intelligence, 1997.
- 9 Parr R., and S. Russell, Reinforcement Learning with Hierarchies of Machines, In Proceedings of Advances in Neural Information Processing Systems 10, 1998.

- 10 Salustowicz R., M. Wiering, J. Schmidhuber, Learning team strategies: Soccer case studies, Machine Learning, 1998.
- 11 Sen S., M. Sekaran and J. Hale, S. Sen, M. Sekaran, and J. Hale. Learning to coordinate without sharing information. In Proc. AAAI, pages 426—431, 1994.
- 12 Sutton R.S., and A. G. Barto, Reinforcement Learning: An Introduction, MIT PRESS, 1998.
- 13 Tan M., Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents, In Proceedings of the Tenth International Conference on Machine Learning, 1993.
- 14 Fausett L., Fundamentals of Neural Networks Architectures, Algorithms and Applications, Prentice Hall, 1994.
- 15 Watkins C., P. Dayan, Q-learning, Machine Learning 8, 279-292, 1992.
- 16 Wiering M., R. Salustowicz, J. Schmidhuber, Reinforcement Learning Soccer Teams with Incomplete World Models, Journal of Autonomous Robots, 1999.

St. Gabriel's Library. An

