Particle Swarm Optimization in Multi-Agents Cooperation Applications

XU LIANG

NATIONAL UNIVERSITY OF SINGAPORE

2003

PARTICLE SWARM OPTIMIZATION IN MULTI-AGENTS COOPERATION APPLICATIONS

XU LIANG, B.ENG.

NANJING UNIVERSITY OF AERONAUTICS AND ASTRONAUTICS

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2003

Acknowledgements

To my supervisors，Dr. Tan Kay Chen and Dr. Vadakkepat Prahlad. Their patient and instructive guidance has shown me that for every challenge on one side, there is solution on the other side.

To my friends and fellows in the Control & Simulation Lab. I have benefited so much from those valuable assistance and discussions. It is really lucky to have so many sincere friends here.

Special thanks to the National University of Singapore for research scholarship, library facilities, research equipments, and an enthusiastic research atmosphere. This is an ideal campus for study, research and life.

Finally, my gratitude goes to my family for their firm support and unreserved love, which have made my life abroad such an enjoyable experience.

Summary

In the past decades, rapid progress has been made in the development of individual intelligence. This progress has consequently made group intelligence, which is based on individual intelligence, applicable and, therefore, more attractive. Concerning current research focus, most of research works on group intelligence are concentrated on external-driven group intelligence, whereas, inner-motivated group intelligence is yet rather a research direction than a research topic. However, as in many circumstances, especially in an isolated environment, since external-driven cooperation is not applicable, inner-motivated group intelligence is necessary. FAMAC (Fully Automatic Multi-Agents Cooperation), to be presented in this thesis, is the very one designed to explore inner-motivated group intelligence so as to offer multi-agents the ability to perform autonomic cooperation independently of external instructions.

In the first part of this thesis, the origination, principles, and structure of FAMAC are described in detail. Human cooperation in soccer game is studied and the principles of human cooperation are replanted into FAMAC. For this reason, FAMAC strategy adopts a structure which combines distributed control with global coordination and comprises of three functional units: the Intelligent Learning and Reasoning Unit (ILRU), the Intelligent Analyzing Unit (IAU) and Central Controlling Unit (CCU).

Equipped with ILRU and IAU, intelligent individuals are supposed to be capable of thinking, analyzing and reasoning. The CCU, however, helps to coordinate the group behavior.

In the second part, two main components, ILRU and IAU, of FAMAC are detailed. Additional knowledge of Neural Network and Fuzzy logic as well as their functions and applications in IAU and ILRU are covered in this part.

A series of simulations are conducted and analyzed in the third part. These simulations are designed to validate the feasibility of FAMAC and compare the effectiveness of M^2PSO network with other computational algorithms regarding their performance in the training of FAMAC. Through simulations, significant advance has been achieved with the multi-agents system that adopts the FAMAC strategy. Further advance has also been achieved after the introduction of M^2PSO-NETWORK into FAMAC. These experimental results have proved that the inner-motivated group intelligence, may or may not be in the format of FAMAC, is realizable and is efficient in prompting the capacity of multi-agents as a united team.

Contents

List of Figures

List of Tables

List of Abbreviations

BP	error Back-Propagation
CCU	Central Control Unit
FAMAC	Fully Automatic Multi-Agents Cooperation
FL	Fuzzy Logic
GA	Genetic Algorithm
IAU	Intelligent Analyzing Unit
ILRU	Intelligent Learning and Reasoning Unit
MA	Multi-Agents
MAC	Multi-Agents Cooperation
MAS	Multi-Agents System
MPSO	Multi-level Particle Swarm Optimization
M^2PSO	Multi-level--Multi-step Particle Swarm Optimization
NN	Neural Network
PSO	Particle Swarm Optimization

Chapter 1

Introduction

1.1 Overview: The main tasks

Intelligent individuals, such as robots and flying vehicles, have become such an important part of modern life that more and more interest, both in research and industry, has arisen in this area. In the meantime, rapid advances in science and technology have promoted the development of such intelligent individuals. As a result, these developments have set up substantial foundation for, and given rise to, the research and technology of group intelligence, which is a kind of intelligence on top of individual intelligence that harmonize group behavior.

Being a most popular existence of group intelligence in nature, group cooperation, has attracted most of the interest in this field. For instance, Robocup has aimed at developing a team of fully autonomous humanoid robots that can cooperate to beat the human world soccer champion team through the utilization of group intelligence.

To archive this goal, for a team of robots, being intelligent and independent is not enough, they must also be capable of working as an integrated team for a common goal based on some strategies, which can assign each robot appropriate thing to do according to its temporary existence. This assignment is not supposed to be done by external force. Instead, as that in human soccer, this assignment is actually realized through the inner negotiation, coordination, and even, in some situations, competition.

Much research work has been previously conducted in artificial cooperation. And there are a huge number of publications in this area each year. However, most of those research works are focused on external-driven cooperation and depend heavily on human researchers. For this reason, much work needs to be done by researchers before cooperation can really come true. Moreover, in such circumstance, artificial cooperation, to some extent, will lack of freedom and flexibility.

This thesis focuses on a cooperation strategy, which we give the name---Fully Automatic Multi-Agents Cooperation (FAMAC), which requires no external interference since intelligent individuals themselves will manage to adjust their behavior to fulfill their task against their opponents' competition and pullback.

In addition, a fresh new training algorithm for FAMAC is brought up for the sake of an even more reasonable cooperation result. This algorithm, which is named M^2PSO-Network, is a combination and improvement from the prototype of PSO (Particle Swarm Optimization) and Neural Network. It is tested and compared with other training algorithms: a traditional algorithm BP (error Back-Propagation), a relatively mature algorithm GA (genetic algorithm), and an orginal PSO algorithm.

1.2 Outline of Thesis

To begin with, some fundamental concepts and background knowledge is presented in chapter 2. A review of previous research sharing the same focus and detailed knowledge about tools and methodologies to be utilized in this research can be found in this chapter.

In chapter 3, Fully Automatic Multi-Agents Cooperation (FAMAC) is put forward. Its original idea, system structure and functions are detailed this chapter. Central Control Unit (CCU), a simple one of the three main components of FAMAC, is also covered in this chapter. The middle part of this chapter is focused on a major component of FAMAC, Intelligent Analyzing Unit (IAU). Functions of IAU and the application of fuzzy logic in IAU will be detailed in this chapter. The concluding part of this chapter, on the other hand, focuses on the other major component of FAMAC, Intelligent Learning and Reasoning Unit (ILRU). Readers are expected to get a clear understand of the principles of FAMAC as a result of a thorough study and decomposition of FAMAC in this and the forgoing chapter.

After that, in chapter 4, simulation is designed to test the proposed idea of FAMAC. A simulated platform is set up to provide agents a game environment. Agent' actions, their corresponding effects, are also defined.

When it comes to chapter 5, a series of simulations are done to simulate games between two teams of agents, one adopting FAMAC and the other not. Performance of the two teams is evaluated, assessed, and compared. Through the comparison, the validity of the idea and structure of proposed FAMAC is confirmed. After that, FAMAC is further improved with a new computation algorithm M^2PSO.

Further discussions and conclusions of the results from chapter 5 are given in chapter 6. Both advantages and defects of FAMAC are referred in this chapter. Following that, a retrospection the research work done in this thesis is conducted.

Chapter 2

Background Knowledge

2.1 Agents, Multi-Agents System, and Multi-Agents Cooperation

Agent*,** referred to as a kind of intelligent individual, is a widely quoted concept in both academic research and technical applications. Since different definition may be given when different character of agent is in the focus, there is still no universal definition of it. In this thesis, a generally accepted definition of agent is sited. ***Agent, which can be either physical or software entity, is self contained and autonomous in certain degree and is capable of perceiving and affecting its working environment either directly by itself or together with other agents. As this definition indicates, an agent is an intelligent individual capable of perceiving, thinking, interacting and working. And it can either have a real material body, such as biologic agent and robot agent, or have an imaginary dummy body, such as software agent.

Multi-Agents System (MAS) is a systematic integration of agents. The purpose of this integration is to make each agent informatively accessible to each other and, thereby, be capable of sharing individual knowledge, as well as temporary information, among all agents to overcome the inherent limitation of individual agents in identifying and solving complicated problem. In a word, agents in this system are required to communicate, negotiate, and coordinate each other. In this manner, agents may be expected to work both independently and interactively. A typical example can be found in the decision-making processes of a robot soccer team. In a team simply made up of a number of agents without adopting the structure of MAS, each agent will make an *optimal* decision solely meeting its own situation, intention, and desire, regardless of the existence and influence of other agents. However, due to random chaos, it is most likely that, though each agent is doing the job that it thinks to be most contributing, none of them can actually carry out its action towards its desired outcome smoothly and all their efforts may be easily counteracted. In the worst situation, they can even crush into each other and totally spoiled the work of the whole team. On the other hand, in a multi-Agents System, each agent will try to exchange information and share its individual knowledge among other agents. By sharing information, they could discuss and negotiate with each other, and then work out a group-wide optimal decision. Based on the above discussion, MAS has led agents evolve from the initial nature individual to social cell and therefore made ***Multi-Agents Cooperation (MAC)*** possible.

Multi-Agents Cooperation (MAC) is targeted at letting agents work together to achieve a common goal, minimizing their counterwork while maximizing their mutual support. The cooperation ranges from competitive cooperation, to antagonistic conflict resolution, to neutral information sharing, and, finally, to supportive task scheduling.

In competitive cooperation, if there are several agents pursuing one certain role in a same team, agents will have to compete for the role and only the fittest agent will be selected to perform this role. During the course of selection, each agent's fitness to perform a certain role is evaluated, by itself and possibly by others as well. The winner, whose fitness value is the highest, is offered the right to perform the target role while the others have to take their less desired roles, which may also be assigned through competition if the number of agents is larger than the number of the roles. This process cycles until every agents has been assigned a role or all the available roles have been taken up. Here is a typical example in robot soccer. When two robots both are very near to the football, which happen to be at the neighborhood of opponent's goal and both of them, according to their own analysis, want to perform an action of shooting. In such a circumstance, if no strategy is taken to handle this hostile competence, it is most likely that neither of them can successfully perform this action due to and conflict and coincidence. Competitive cooperation can handle this problem easily. Under competitive cooperation, these two robots will exchange information and figure out a fair judgment on each agent's fitness value. Then the fitter one will shoot while the other will perform other action to help his team member.

In friendly cooperation, the tem work is more likely to be a series of jobs in time or spatial sequence. Each agent has already been assigned a single and fixed role. Agents are expected to perform their roles in sequence to fulfill the task in the shortest time or with the best quality. In such situation, there is solely cooperation among all agents. This cooperation is mainly concerning with job arrangement and scheduling. Taking multi-agents to make a simple table for an example, if provided all necessary wood components for a table and tools such as hammer and nails, robots are to pin up these wood components into a stable table. One robot is assigned the role to assemble these wood blocks with another robot is to pin up them. Neither can any single robot make a table by itself, nor are they supposed to compete against each other. So in this case, there is only friendly cooperation between the two robots.

2.2 A review of MAC

In the previous section, concerning the amity among agents in MAS, we have classified MAC into several general categories. In this section, a review of MAC is conducted and focused on the degrees of intelligence and automation in MAC. Generally, in this thesis, MAC is classified into three different ranks according to its intelligence and automation. These three ranks of MAC are: passive cooperation, semi-autonomous cooperation, and autonomous cooperation.

As shown below, the first rank of MAC, passive cooperation is a sort of fixed cooperation Strategies:

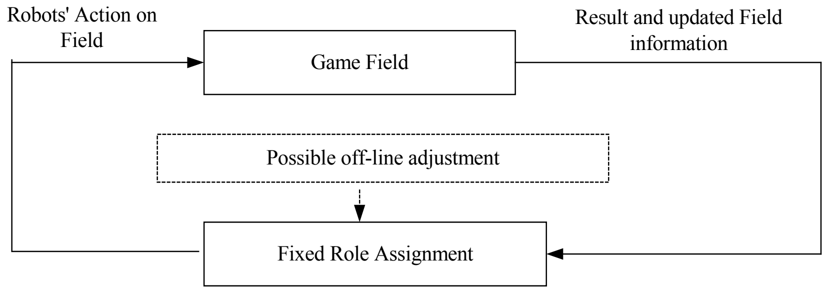

Fig.1: First rank of MAC: Passive cooperation

In this kind of cooperation, agents are individuals that are capable of doing something rather than thinking about something and do not have any idea about cooperation. Therefore, to design cooperation for such agents, human designer needs to arrange everything about cooperation by telling what they should and should not do. For this reason, this cooperation is critical upon the environment as well as analytical ability of human designer.

Examples of passive cooperation can be easily found in early robot soccer teams in which the roles and actions of robots are determined before the match starts and, in any circumstance, cannot be changed during the course of match. The below are some examples of this kind of cooperation:

A method for Conflict detection and entire information exchange which eventually leading to an acceptable decision is presented in [1].

A task-oriented approach and a motion-oriented approach is used for multi-robots cooperation in the space [2].

On the other hand, in other kind of fixed cooperation strategies, the roles of agents are not that absolutely fixed, instead, they can demonstrate some property of variability when agents are working in the environment. As in [3], a fixed role assignment is put introduced for agents according to their positions. However, this change only occurs at a designed location spot and at a certain moment that is pre-determined by the designer. This cooperation seems more flexible. However, it is still a fixed operation since each agent role at every moment is under the control of the designer. The agents have to *obey* the will of human designers.

The second and higher rank of MAC, semi-autonomous cooperation, is a rank of cooperation strategies that support agents' intelligent learning following supervision of humankind. Rather that tell agents what to and not to do, human designers find it more helpful to teach agents to think about what they should do. Fig.2 illustrates a typical semi-autonomous cooperation:

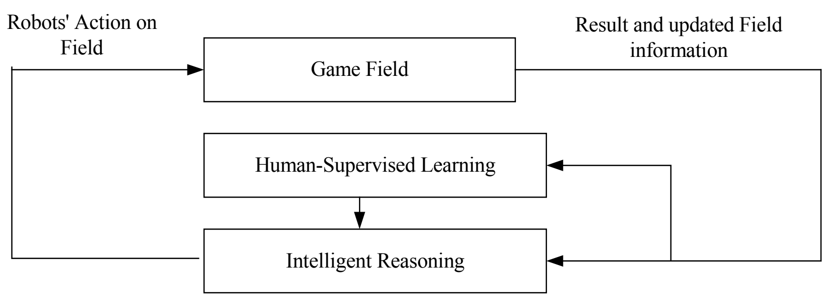

Fig.2: Second rank of MAC: Semi-autonomous cooperation

Instead of just do as being told, agents try to learn to behave properly by themselves. The character of this kind of cooperation is that agents can learn to adjust their behaviors towards what are expected but, as they are still not autonomous enough, they do not know the reasons of their doings. And, before they can learn, they need instructions and sufficient information about how and what to learn. A series of rules will be set up by the human designer to supervise the learning process of agents. Since human designers need to be involved in this cooperation before agents are set out to work, this cooperation also requires information and analysis about the environment. But since human supervisors need not to arrange every detail about the cooperation, their workload has been significantly cut down.

According to the classification, research on semi-autonomous includes:

Multiple objective decisions making based on behavior coordination and conflict resolution using fuzzy logic in [4].

In [5], the authors report a fuzzy reinforcement learning and experience sharing method in dealing with multi-agent learning in dynamic, complex and uncertain environments.

Fuzzy behavior coordination using a decision-theoretic approach is implemented in [6] to instruct multi-robots to perform a serial of actions in consequence.

Li Shi et al combined Neural Networks with fuzzy logic and put forward a supervised learning to map the competition among the robots. [7]

Jeong and Lee used genetic algorithm trained fuzzy logic to instruct their agents to capture quarry. [8]

As new requirements arise for agents to commit complicated tasks automatically in an unknown and complex environment which may be beyond the reach of humankinds. Cooperation of even higher intelligence is required for agents to acclimatize themselves to their working environment. This rank of cooperation need to be more advanced than semi-autonomous cooperation as agents should be independent enough to supervise their learning themselves. To behave such cooperation, agents are expected to be capable of identifying, analyzing, and affecting the environment through their own efforts. Moreover, their learning performance is not, or at least not mainly, evaluated by how they react to a certain situation but is evaluated by agents' overall performance towards committing a complete mission smoothly. If this cooperation strategy is realized and adopted, ideally, the human manipulator only needs to do the least work: tell the agents what they are expected to achieve but not what to do. And after that, the agents will try to fulfill the mission all by own. That is, they evaluate their work, resolve their problems, and learn to improve their performance automatically. This cooperation hardly needs any prerequisite information about the environment. No intervene from outside is needed during the learning process.

By now, most of the research on multi-agents cooperation is concentrated on the second rank. In order to explore the validity of the autonomous cooperation, we carried out this research on multi-agents autonomous cooperation that aims at enabling agents to learn to cooperate independently of human instruction and be capable of adapting to dynamic environment.

A fully autonomous multi-agents cooperation strategy namely FAMAC is proposed in this thesis. Agents adopting FAMAC strategy are expected to behave like social beings as a result of introduction of the three intelligent components, Intelligent Learning and Reasoning Unit (ILRU), Intelligent Analyzing Unit (IAU) and Central Control Unit (CCU). ILRU is a unit for agents to remember what happened before, both the experience of success and lessons of failure, and, thus, when requirements rise to make a decision, to perform associative thinking upon what has been experienced and remembered. IAU is a unit designed to provide agents the ability to analyze information, evaluate results, and correct errors. So after decisions are made through the ILRU and then corresponding actions have be exerted upon the environment, agents are able to tell whether these decisions are reasonable through an examination of their effects upon the environment. The result of analysis is feedback to ILRU for its future evolvement. The CCU, however, will see to the problems of global coordination for cooperation. Based on some simple rules, it tries to solve any potential conflict and harmonize the behavior of agents.

2.3 Intelligent Computation Algorithms in this Thesis

2.3.1 Fuzzy Logic

The term "fuzzy logic" emerged in the development of the theory of fuzzy sets by Lotfi Zadeh [Zadeh (1965)]. A fuzzy subset A of a (crisp) set X is characterized by assigning to each element x of X the *degree of membership* of x in A (e.g. X is a group of people, A the fuzzy set of *old* people in X). Now if X is a set of propositions then its elements may be assigned their *degree of truth*, which may be "absolutely true," "absolutely false" or some *intermediate* truth degree: a proposition may be more true than another proposition. This is obvious in the case of vague (imprecise) propositions like "this person is old" (beautiful, rich, etc.). In the analogy to various definitions of operations on fuzzy sets (intersection, union, complement, …) one may ask how propositions can be combined by *connectives* (conjunction, disjunction, negation, …) and if the truth degree of a composed proposition is determined by the truth degrees of its components, i.e. if the connectives have their corresponding *truth functions* (liketruth tables of classical logic). Saying "yes" (which is the mainstream of fuzzy logic) one accepts the truth-functional approach; this makes fuzzy logic to something distinctly *different from probability theory* since the latter is not truth-functional (the probability of conjunction of two propositions is *not determined* by the probabilities of those propositions).

The basic structure of an example, which is two-input, one-output, three-rule tipping problem, is shown in the figure below.

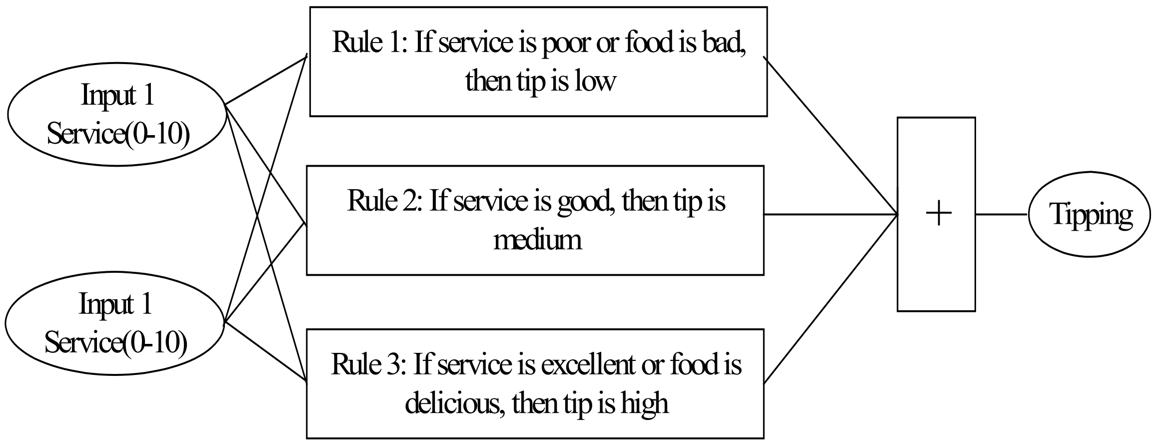

Fig.3: Application of Fuzzy Logic into *Tipping problems*

Information flows from left to right, from two inputs to a single output. The parallel nature of the rules is one of the more important aspects of fuzzy logic systems. Instead of sharp switching between modes based on breakpoints, we will glide smoothly from regions where the system's behavior is dominated by either one rule or another.

2.3.2 Neural Network

Neural network has been proved to be effective and powerful in prediction, system modeling, data filtering and data conceptualization etc [9]. Especially, in the case of supervised learning, if the learning objective is rational with explicit record of input and output data, neural networks can track this object and construct a model for it with very high accuracy. Whereas, in our case of cooperation strategy learning, as the environment is supposed to be a black-box to outside word, the information is far from being sufficient or explicit. What's more, this information cannot be used directly as sample data for neural networks training. Therefore, for the purpose of data processing and analyzing, fuzzy logic is implemented in our method for the purpose of analyzing the data and results. The fuzzy logic unit is expected to furnish neural network with advisory instruction on *how to study* as well as *what to stud*y. Neural network will refer to such instructions and learn to evaluate the status and performance of agents.

Multi-layer feed-forward neural network trained with BP algorithm is widely used today. Its convergence to a local optimal has already been mathematically proven. However, as a result of its benefit of fast gradient convergence, it is very easily stuck to a local optimal. For this reason, it is very difficult and sometimes impossible to use this training algorithm solely to find the global optimum for neural networks. And, thus, alternative methods, Genetic Algorithm (GA) and Particle Swarm Optimization (PSO), are introduced in this thesis to train neural networks. These methods both are simpler than BP algorithm in mathematical computation and thereby can be expected drastically reduce the computing time through the entire solution space.

2.3.3 Genetic Algorithm

A genetic or evolutionary algorithm applies the principles of evolution found in nature to the problem of finding an optimal solution to a Solver problem. The principle of evolution of human gene is quoted in GA. In a "genetic algorithm," the problem is encoded in a series of bit strings (gene) that are manipulated by the algorithm; in an "evolutionary algorithm," the decision variables and problem functions are used directly. After a population of genes are selected and evaluated, they may undergo a s election, mutation, or crossover process. Optimization is realized in this manner.

GA can solve problems that do not have a precisely defined solving method, or if they do, when following the exact solving method would take far too much time. There are many such problems; actually, all still-open, interesting problems are like that. Such problems are often characterized by multiple and complex, sometimes even contradictory constraints, that must be all satisfied at the same time. Examples are crew and team planning, delivery itineraries, finding the most beneficial locations for stores or warehouses, building statistical models, etc.

GA works by creating many random "solutions" to the problem at hand. Being random, these starting "solutions" are not very good: schedules overlap and itineraries do not traverse every necessary location. This "population" of many solutions will then be subjected to an imitation of the evolution of species. All of these solutions are coded the only way computers know: as a series of zeroes and ones. The evolution-like process consists in considering these 0s and 1s as genetic "chromosomes" that, like their real-life, biological equivalents, will be made to "mate" by hybridization, also throwing in the occasional spontaneous mutation. The "offspring" generated will include some solutions that are better than the original, purely random ones. The best offspring are added to the population while inferior ones are eliminated. By repeating this process among the better elements, repeated improvements will occur in the population, survive and generate their offspring.

2.3.4 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is also a fruit of careful and minded observance of natural existence and was developed by James Kennedy and Russell Eberhartm [10]. This algorithm simulates social behavior of particles such as a bird flock and fish school searching through a target space, each particle representing a single intersection in the space. The particles evaluate their positions with respect to a goal at each iteration, and particles within a local neighborhood share memories of their best positions, and then use those memories to adjust their own velocities, and thus subsequent positions. In this way, the entire search space may be searched and examined thoroughly. Extended PSO technique is an extension in the structure of

PSO, which aims at improving the searching accuracy as well as search speed of the PSO algorithm.

Unlike BP, PSO is a global optimization algorithm. PSO, with its simple concept and inexpensiveness in computation, can comprise a large number of particles and thus could possibly search through the whole subspace for a global optimal solution. Especially for some 2-dimensional or 3-dimensional problem, enough particles can be chosen randomly to cover every point and every corner of the entire solution space.

Though goes without mathematical support, due to its simple concept and convenience in application, PSO has been used in several areas. Two kinds of typically usages of PSO are:

Power system control using PSO [11] and Neural networks training using PSO [12] [13].

Particle Swarm Optimization (PSO) is a global optimization algorithm. PSO, with its simple concept and inexpensiveness in computation, can comprise a large number of particles and thus could possibly search through the whole subspace for a global optimal solution. Especially for some 2-dimensional or 3-dimensional problem, enough particles can be chosen randomly to cover every point and every corner of the entire solution space.

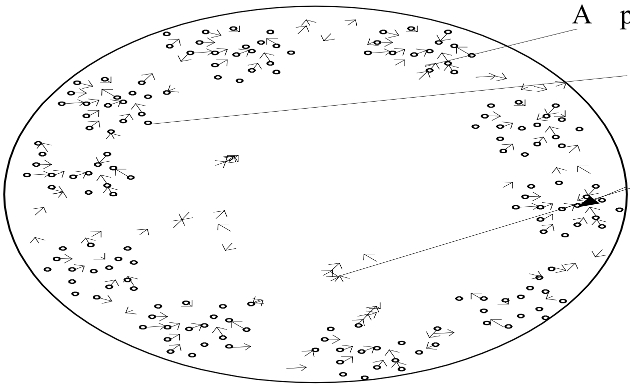

Fig.4: Particle Swarm Optimization

Because of its advantage in searching through a large solution space quickly and thoroughly, PSO is applied to find a global optimal solution for neural network. In this application, each particle is a vector standing for a whole set of weights. Each particle is evaluated and compared with its previous best value (this is pbest) and the global best among all the particles (that is gbest) and is adjusted using the following equations （1），（2）:

$$v_{id} = v_{id} + c_1 \times rand() \times (p_{id} - x_{id}) + c_2 \times rand() \times (P_{gd} - x_{id}) \quad (1)$$

$$x_{id} = x_{id} + v_{id} \quad (2)$$

Where, d is the dimension of the solution space.

x_{id} is the value of i-th particle. i=1…n $rand()$ is a random number

v_{id} is the varying speed of x_{id} c_1, c_2 are constant parameters

p_{id} is the pbest of i-th particle i=1…n p_{gd} is the gbest of all particles

With the dimension of the solution space increasing, the number of particles required by PSO method also increases drastically in geometrical order. High computational speed, one of numerous advantages of PSO, may be cancelled if the dimension of the solution space reaches a certain value.

Chapter 3

Fully Automatic Multi-Agents Cooperation (FAMAC)

3.1 The proposed FAMAC

Fully Autonomous Multi-Agents Cooperation (FAMAC) is the very MAC designed to strengthen the ability and intelligence of agents to cooperate without online supervision from external forces. Ideally, fully autonomous cooperation means that once agents are set to work they will be absolutely independent and free and are supposed to behave like a responsible adult in society. The significant point of this kind of cooperation strategy is that, from the initial state of be absolutely ignorant of the surrounding environment, through their inner-driven study and analysis, theoretically, agents can finally explore all the information about environment and learn to commit any mission. However we won't go so far in this research as it is not practical under present scientific and technical conditions and actually, in most applications of FAMAC, some fundamental information about the environment is available beforehand. Such fundamental information may include invariable environmental features, i.e., the boundaries of environment that confine the active region of agents, restrictions or regulations that regulate agents' behavior, and so on.

Once such basic information is set known to agents, they will try to explore more information, which is important yet still unknown, and bring their missions to success. During this course, they assess their work, resolve their problem, and improve their performance automatically. No intervene from outside is needed before and during the learning process. To behave such high intelligence, agents should possess of the skills of thinking, analyzing and remembering. This, obviously, need a combination of technologies in Artificial Intelligence. Thus, several intelligent algorithms are utilized in this research. For instance, Neural Network is introduced to play the function of information storing as well as associative thinking of agent's brain whereas fuzzy logic is implemented as the analytical part of agent's brain.

3.1.1 Origination of Idea of FAMAC

For the purpose of reproducing human intelligence in an artificial world, it is always worthwhile to take a first look into human behavior in similar circumstance. This time, again, we come to human soccer game for inspirations. As a common sense, for a soccer team, apart form the individual competence of players, team cooperation is also of very great importance and can significantly affect the performance of each team, especially when competence gap between two teams is not too large. No players are born good cooperators. Most of them do not know how to cooperate when they first take part in such sports. Though they may be told some experiential knowledge before, in reality, things will be somewhat different from knowledge. To go mature, they need to practice, practice again, and practice again and again. Therefore, every team would spend much of the daily training time on cooperative drilling. During the drilling, various cooperation strategies are put forward, tested and improved in the field. After trying different strategies, analyzing corresponding results, updating old strategies and re-trying updated strategy; those strategies that are more likely to produce a positive result is chosen as a reasonable cooperation strategy for later reference and improvement. The training process is shown in Figure below:

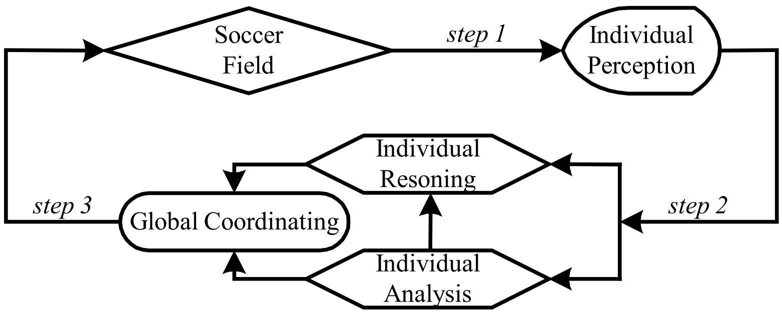

Fig.5: Illustration of a typical training cooperation strategy learning through daily training in real soccer sports

At the first step of this process, each individual player tries to explore the working environment by itself. Here, the working environment is not merely a working field; it also includes all individuals working in the field and all other relative factors. Considering that in the soccer game, environment includes the football fields, all the players in the field, the coaches, the referees, the fans and other external factors such as the weather and the light. At the second step, the information is analyzed and effect of individual player on the environment is assessed. Thereafter, each player may adjust their further action and improve their skill in order to perform better later. These actions, which are based on individual's judgment, will be examined through global coordinating. The global coordinating in reality often reflects an individual's self-identification of its role and its liability in achieving the group task. Once all the individuals have worked their actions out, they will act upon the environment in the third step to drive the environment towards their target: offend for a goal or defend against opponent's goal.

During this process,

1. If the result is positive (the host team wins), the cooperation associated with such circumstance is deemed as a suitable one and will match similar circumstance and is worth to be recorded as a successful cooperation sample for future reference.

2. Otherwise, if the result is negative (the host team loses), lessons are learned and suggestions, on improving or replacing this cooperation, may be brought forward. If the result turns out to be very extremely negative, this cooperation will be considered to be totally a failure and is unreasonable and players should try different ways later in similar circumstances later. On the other hand, if the result is not that disappointing, after some improvements, it can still be utilized, however, as candidate cooperation and tested again in later rounds of similar circumstances.

3.1.2 System Structure of FAMAC

The idea of FAMAC is simply a reproduction of idea of human cooperation. Just as what players behave in soccer field, agents in the simulation also evolve in a way that practice makes perfect. The structure of FAMAC and its function in the multi-agents system is illustrated in figure below:

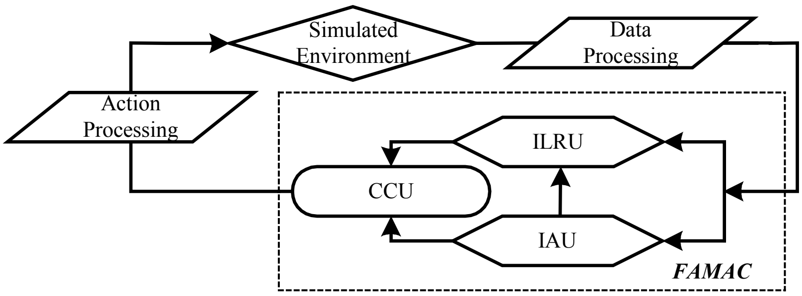

Fig.6: Idea representation FAMAC and its structure

As in the figure, ILRU (Intelligent Learning and Reasoning Unit) and IAU (Intelligent Analyzing Unit) correspond to human reasoning and human analyzing respectively. CCU (Central Control Unit) will perform the function of global coordinator. However, unlike Global Coordinator, which is inseparable from human brain, the CCU is a separated part independent of individual agent. This slight difference has greatly enhanced the cooperation by minimizing chance of a conflict caused by failure of exchanging information among agents.

Once the information of environment is available, it will be transmitted to FAMAC after necessary procession. In the framework of FAMAC, the 3 units, each one will perform its distinguishable function.

1. The agent analyzes the effectiveness of its previous action, assesses the performance of ILRU and generates advice on the improvement of ILRU.
2. The ILRU receives data form the environment and IAU and outputs the fitness value of each agent.
3. The CCU will deal with the coordination of the roles of all agents.

Here, since the research is designed to explore cooperation in an environment that is partially unknown, the available information is limited and only those of agents as well as the effects of agents' action upon the environment can be obtained. Such information is observable and thus is not critical upon particle environment. This will ensure the method feasible in almost any environment though it works as black box to us and, in most occasions, sufficient information about environment cannot be easily obtained. What's more, many environments are dynamic and might change in every moment. As a result, against the human-dependent intelligent learning in MAC in figure 2, self-supervised learning has taken the place of human-supervised learning in FAMAC.

As complementary research, in this thesis, in order to enhance the learning ability of agents and consequently improve the overall performance of FAMAC strategy, a new algorithm, Multi-level-Multi-Step Particle Swarm Optimization Network (M^2PSO-Network), is proposed to replace Neural Network in ILRU. This M^2PSO Network algorithm is a revised and asynchronous format of Neural Network, in which the weights relating to nodes in hidden layer are updated asynchronously. That is the successive node in hidden layer will not be trained until, the forgoing one has been totally trained after enough training steps.

3.2 The Intelligent Analyzing Unit (IAU)

3.2.1 Functions of IAU

The powerful function of Neural Network in data mapping makes it an ideal tool for information storage as well as associative thinking and, consequently, the core functional composition of Intelligent Learning and Reasoning Unit. However, as mentioned before, neural network works best when the object in study is rational with sufficient and explicit information. However, in this research, the information about the environment is neither sufficient nor explicit enough to be used as direct sample data for neural networks training. Hence, these data must be processed and translated into a form that is more recognizable to neural network. To solve this problem, we introduced an Intelligent Analyzing Unit (IAU).

Concerning the functional consequence, there is a delay of one step between the ILRU and IAU. In the system of FAMAC, ILRU functions first and IAU functions a step later since it is designed to deal with the data flow succeeding that of ILRU. Once the feedback from environment is available, IAU analyzes the information collected from the environment as well as knowledge in the database. According to the result of analysis, an evaluation about the agents' performance, as a consequence of their actions upon environment, is brought forward. Since the actions of agents are controlled by ILRU, the performance of agents will indicate how well the ILRU works. Thus by analyzing the performance of agents, IAU is actually analyzing the performance of ILRU. After that, a set of data containing instructions for improving ILRU as well revised form of data from environment is transferred to ILRU.

Due to its advantages in handle imprecise and nonlinear information, Fuzzy logic is utilized in IAU and plays an important part in FAMAC as the analysis part of agent's intelligence. Since it's impossible to use the direct information from the simulation environment to adjust the weights of neural networks, we apply fuzzy logic to analyze the match result. The results of this analysis are stored into the database, which serves as a collection learning samples for neural Networks.

3.2.2 Fuzzification

Lotfi Zadeh pioneered a method of modeling human imprecise reasoning using fuzzy sets. Using this technique, the concept 'tall' is related to the underlying objective term, which it is attempting to describe; namely the actual height in centimeters. The transformation of an objective term into a fuzzy concept is called fuzzification. As an example, the term 'tall' can be represented in this graph:

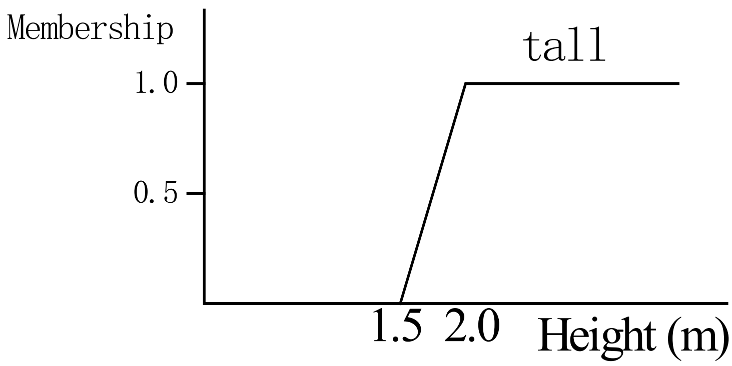

Fig.7: An example of Fuzzification

It shows the degree of membership with which a person belongs to the category (set) 'tall'. Full membership of the class 'tall' is represented by a value of 1, while no membership is represented by a value of 0. At 1.5 m and below, a person does not belong to the class 'tall'. At 2.0 m and above, a person fully belongs to the class 'tall'. Between 1.5 m and 2.0 m the membership increases linearly between 0 and 1. The degree of belonging to the set 'tall' is called the confidence factor or the membership value. The shape of the membership function curve can be non-linear. The purpose of the fuzzification process is to allow a fuzzy condition in a rule to be interpreted. For example the condition 'person = tall' in a rule can be true for all values of 'height', however, the confidence factor or membership value of this condition can be derived from the above graph. A person who is 1.75 m in height is 'tall' with a confidence factor of 0.5 (membership value of the club 'tall'). It is the gradual change of the membership value of the condition 'tall' with height that gives fuzzy logic its strength.

In this thesis, similar process is carried out to fuzzify the inputs.

Denote:

T1: The shortest time taken by agents of team A to reach the destination

T2: The shortest time taken by agents of team B to reach the destination

maxT1(maxT2): The maximum of K values of T1(T2)

minT1 (minT2): The minimum of K values of T1(T2)

$$\max T1 = \max imum(T1)\Big|_{1}^{K} \tag{3}$$

$$\max T2 = \max imum(T2)\Big|_{1}^{K} \tag{4}$$

$$\min T1 = \min imum(T1)\Big|_{1}^{K} \tag{5}$$

$$\min T1 = \min imum(T2)\Big|_{1}^{K} \tag{6}$$

$\mu_{1,good}$ and $\mu_{2,good}$ represent how well a team performs in a match.

$$\mu_{1,good} = \begin{cases} 0 & 0 < T1 < \min\ T1 \\ \dfrac{(T1 - \min\ T1)^2}{(\max\ T1 - \min\ T1)^2} & \min\ T1 < T1 < \max\ T1 \\ 1 & \max\ T1 < T1 \end{cases} \tag{7}$$

$$\mu_{2,good} = \begin{cases} 0 & 0 < T2 < \min T2 \\ \dfrac{(T2 - \min T2)^2}{(\max T2 - \min T2)^2} & \min T2 < T2 < \max T2 \\ 1 & \max T2 < T2 \end{cases} \tag{8}$$

μ_{Offend} , μ_{Defend} and μ_{ward} indicate how good the Offend/defend/ward action are.

$$\mu_{Offend} = \sqrt{\frac{(\mu_{1,good})^2 + [1 + (\mu_{2,good} - 0.5)^2]^{-1}}{2}} \tag{9}$$

$$\mu_{Defend} = \sqrt{\frac{(1 - \mu_{2,good})^2 + [1 + (\mu_{1,good} - 0.5)^2]^{-1}}{2}} \tag{10}$$

$$\mu_{ward} = \sqrt{\frac{(\mu_{2,good})^2 / 4 + (\mu_{1,good})^2 / 2}{0.75}} \tag{11}$$

$$E_{middle} = \sqrt{\frac{(\mu_{2,good})^2 / 4 + (\mu_{1,good})^2 / 2}{0.75}} \tag{12}$$

Since the parameters in the fuzzy logic rules are related to the environment and tend to vary with simulation going on, fuzzy logic in this form can adapt itself to the dynamic environment.

Here, membership functions are of the triangle format.

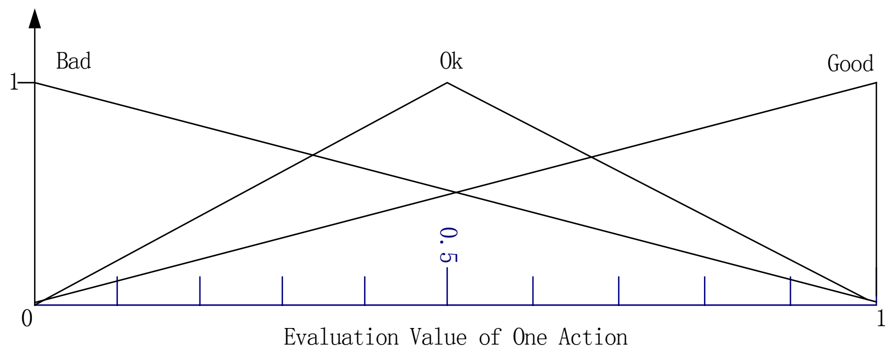

Fig.8: IAU: Membership functions

In succession, the evaluation value is adjusted by some if-then rules and the adjusted value is forwarded to ILRU for its training.

3.2. 3 Fuzzy Rules

Human beings make decisions based on rules. Although, we may not, at all time, be aware of it, all the decisions we make are all based on computerlike if-then statements. If the weather is fine, then we may decide to go out. If the forecast says the weather will be bad today, but fine tomorrow, then we make a decision not to go today, and postpone it till tomorrow. Rules associate ideas and relate one event to another.

Fuzzy machines, which always tend to mimic the behavior of man, work the same way. However, the decision and the means of choosing that decision are replaced by fuzzy sets and the rules are replaced by fuzzy rules. Fuzzy rules also operate using a series of if-then statements. For instance, if X then A, if y then b, where A and B are all sets of X and Y. Fuzzy rules define fuzzy *patches*, which is the key idea in fuzzy logic.

In the simulation, to evaluate the fitness for an agent to perform a certain task, performances of both the agents and the team it belonging to must be considered. Thus in the composition of the fuzzy rules, both factors are involved as input vectors.

1, If (Team performance is good) and (Agent's Offend action is successful) then (Agent's fitness to offend is high)

2, If (Team performance is good) and (Agent's Offend action is Okay) then (Agent's fitness to offend is medium)

3, If (Team performance is good) and (Agent's Offend action is not successful) then (Agent's fitness to offend is Low)

4, If (Team performance is good) and (Agent's Defend action is successful) then (Agent's fitness to Defend is high)

5, If (Team performance is good) and (Agent's Defend action is Okay) then (Agent's fitness to Defend is medium)

6, If (Team performance is good) and (Agent's Defend action is not successful) then (Agent's fitness to Defend is Low)

7, If (Team performance is good) and (Agent's Ward action is successful) then (Agent's fitness to Ward is high)

8, If (Team performance is good) and (Agent's Ward action is Okay) then (Agent's fitness to Ward is medium)

9, If (Team performance is good) and (Agent's Offend action is not successful) then (Agent's fitness to Ward is Low)

10, If (Team performance is not good) and (Agent's Offend action is successful) then (Agent's fitness to offend is medium)

11, If (Team performance is not good) and (Agent's Offend action is Okay) then (Agent's fitness to offend is high)

12, If (Team performance is not good) and (Agent's Offend action is not successful) then (Agent's fitness to offend is Low)

13, If (Team performance is not good) and (Agent's Defend action is successful) then (Agent's fitness to Defend is medium)

14, If (Team performance is not good) and (Agent's Defend action is Okay) then (Agent's fitness to Defend is high)

15, If (Team performance is not good) and (Agent's Defend action is not successful) then (Agent's fitness to Defend is Low)

16, If (Team performance is not good) and (Agent’s Ward action is successful) then (Agent’s fitness to Ward is medium)

17, If (Team performance is not good) and (Agent’s Ward action is Okay) then (Agent’s fitness to Ward is high)

18, If (Team performance is not good) and (Agent’s Offend action is not successful) then (Agent’s fitness to Ward is Low)

3.2.4 Aggregation of outputs and defuzzification

The probor (probabilistic or) method is employed to aggregate the outputs. The logic description of probor is: $\mathrm{probor}(a, b) = a + b - ab$. The most popular defuzzification method, namely centroid calculation, is used to calculate the final output value, which is a single number.

If the conclusion of the fuzzy rule set involves fuzzy concepts, then these concepts will have to be translated back into objective terms before they can be used in practice. For a rules set including the credit limit rule described in the previous section, fuzzy inference will result in the terms 'credit limit is low', 'credit limit is medium' and 'credit limit is high' being assigned membership values. However, in practice, to use the conclusions from such a rule base we need to defuzzify the conclusions into a crisp credit limit figure. To do this we need to define the membership functions for the credit limit outcomes.

In the previous sections, we have made evaluations on agents' actions of offending, warding, and defending as terms low, medium, and high. Once these evaluations have been successfully done, the rest job is only to defuzzify these terms into a number that can be viewed and employed directly.

3.3 Intelligent Learning and Reasoning Unit (ILRU)

3.3.1 Functions of ILRU

ILRU is actually not a single unit but a pool of individual learning and reasoning units ----the memorial and reasoning unit of agent's intelligence. Each individual unit receives information from the environment and making evaluation on agent's positions and motions. However, these individual units are not isolated from each other. In stead, they have connections. Besides evaluates its host agent, each unit will also evaluate the positions and motions of other agents. Since they are connected, they will exchange their evaluations about each other. These connections among individual units have insured that the final results of evaluation are objective rather than subjective. Therefore, each agent will not be that ego-focused as others may influence it and it can also influence others. In this way, they are themselves and they are elements of a team.

The functions of ILRU are realized through neural networks. Three-layer neural networks are adopted in the system. Since two teams each comprises of 5 members and each member has its velocity and position in the field, the 2-dimensional environment information contains 20 sets of data. These 20 sets of data form the input vector of Neural Network. The outputs of Neural Network represent the evaluation of the agent's position and motion and will be passed to the Central Control Unit (CCU) for the purpose of global role assignment.

The function of ILRU is illustrated below:

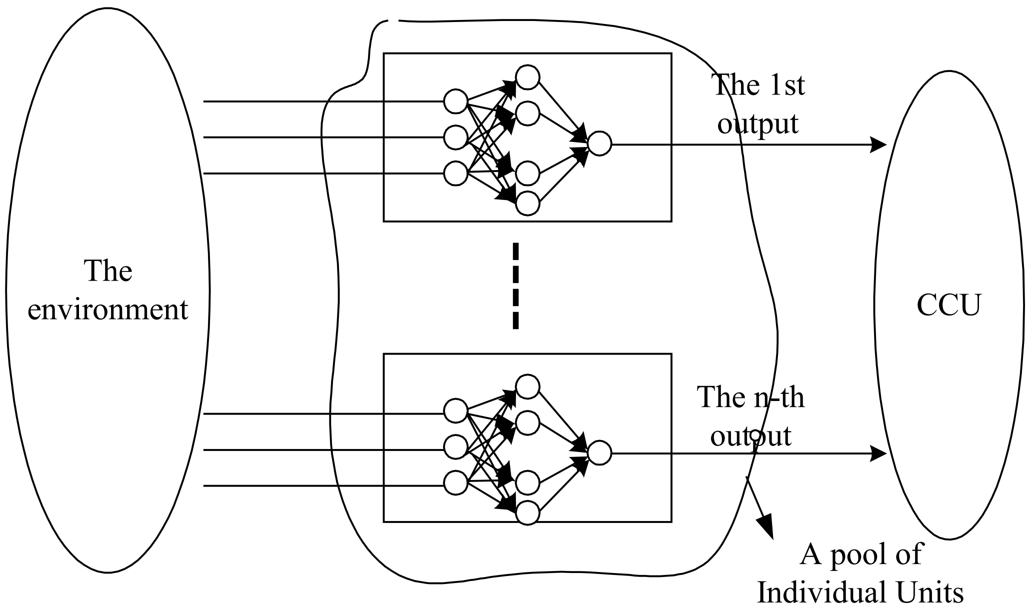

Fig.9: Illustration of function of ILRU

As soon as the role assignment has been successfully carried out, agents will behave properly and influence the environment by their actions. The effects of agents on the environment are fed back to IAU for further evaluation. The results of evaluation will be used to improve the training of ILRU. In the off-line training process, neural networks are not trained immediately after a new round of test is carried out. Actually due to the inner inertia of neural network, which is caused by its large scale of database containing match results over a long period, the neural networks cannot response to a new sample. Therefore, in this training, neural networks will be trained after a certain number of tests have been done.

3.3.2 Optimization for Neural Network

A most popular Neural Network training algorithm, Error Back-Propagation (BP) algorithm is applied to train the neural network. Error Back-Propagation is a gradient descent algorithm that adjusts the weights of neural networks little by little to reduce the error at each step. At each step, after input information is fed to neural network, the output of neural network is produced and compared to the desired output. The error between the output of neural network and the desired output is then fed back to the neural network for the purpose of weights adjustment. A gradient descent that is used to adjust the weights has insured the decease of the error. After weights adjustment, the new output of the adjusted neural network is computed. This process repeats step after step until a minimal error is obtained. Detailed discussion about neural network training is provided later.

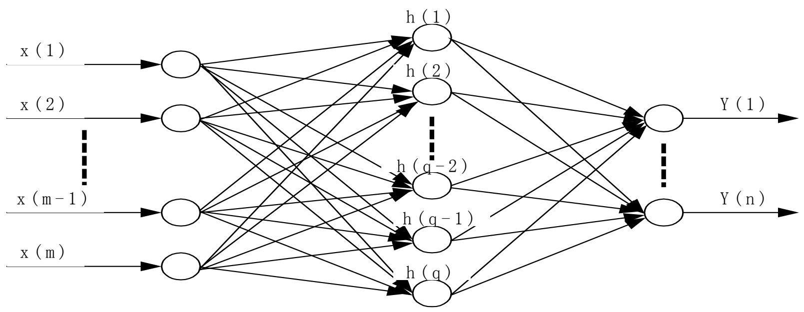

Fig.10: Structure of neural network

The neural network is of the size (m, q, n): m input nodes, q-hidden nodes and n output nodes. Node functions at the 3 layers are purelin, tansig, and purelin respectively. Here purelin is a linear function and tansig is an S-shaped function:

$$\tan sig(x) = \frac{2}{1+e^{-2x}} - 1 \tag{13}$$

Let the weight between the input layer and the hidden layer be $w1 \in R^{q\times m}$ and the weights between hidden layer and output layer be $w2 \in R^{n\times q}$. The threshold value of nodes in hidden layer and output layer are $b1 \in R^{q}$ and $b2 \in R^{n}$ respectively. $Y_D \in R^{n}$ stands for the desired output of neural network

Thus the real output of neural network is:

$$Y = purelin(w2 \times \tan sig(w1 \times x + b1) + b2) \tag{14}$$

Let the neural network's error function be:

$$E(w1, b1, w2, b2) = \frac{1}{2}\sum \| Y_D - Y \|^2 \tag{15}$$

Where $\| Y_D - Y \| = \sqrt{(Y_D - Y)^T (Y_D - Y)}$

According to the error function, in order to globally optimize the neural network, the weights $(w1, b1, w2, b2)$ minimizing $E(w1, b1, w2, b2)$ throughout the entire solution space, instead of a local space, is to be discovered.

Two assumptions:

In later discussion, we assume that:

1° The solution space is not indefinite. Or equivalently, the weights of neural network are bounded. This assumption is based on the two practical considerations.

First, in computer computation, the value that a computer can express is definite. Thus the value of weight must be definite and the solution space should be definite too.

Secondly, consider $Y = w2 * \tan sig(w1 * x + b1) + b2$

If $w2/b2$ goes indefinite, Y will also goes indefinite.

If $w1/b1$ goes indefinite, as $\tan sig(\infty) = 1$ and $\tan sig(-\infty) = -1$, the outputs will not respond to the variance of input.

2° Neural network do not have redundant hidden nodes.

It means that any of 2 hidden nodes are not all equally (evaluated by the weights) connected with the input nodes. For any 2 hidden layer node, kth node and lth node, at least one of the following (m+1) inequations is valid.

$$w1_{ki} \neq w1_{li} \quad \text{, i=1,...m} \tag{16}$$

Where $b1_k \neq b1_l$

If not, the neural network is said to have redundant nodes since these two nodes can be combined into one node.

Due to the property of symmetry of nodes in neural network hidden layer, any two nodes can interchange without infecting the value of output. So given one global optimum, by interchanging nodes yield many new global optima can be produced. Let the number of the optima after a thorough interchange be T.

$$T = C_q^1 C_{q-1}^1 C_{q-2}^1 \cdots C_2^1 C_1^1 = q! \tag{17}$$

The whole solution space can be equally divided into $q!$ subspaces size and there is at least one global optimum in each subspace. So, if the hidden nodes are sorted according to a certain rule, in order to find the global optimum, only one subspace instead of the whole solution space is to be searched. Take a 3-dimensional solution space for example. As illustrated in Fig.8, axis X, Y and Z are assumed to be mutually symmetric. It can be proved that the subspace X>Y>Z enveloped by two shadowed plane in fig.11 is a subspace that contains all the global solutions. In fact, let X, Y and Z stands for 3 hidden nodes of a neural network. If (X1, Y1, Z1) is a global optimal solution and the following holds that Y1>X1>Z1, (X1, Y1, Z1) is certainly outside the subspace X>Y>Z. But (Y1, X1, Z1), which is a symmetric to (X1, Y1, Z1), is also a global optimum and is in the subspace Y1>X1>Z1. Thus it is proved that in a 3-dimnesion space a $\frac{1}{3!}$ portion of the entire solution contains all the global solutions.

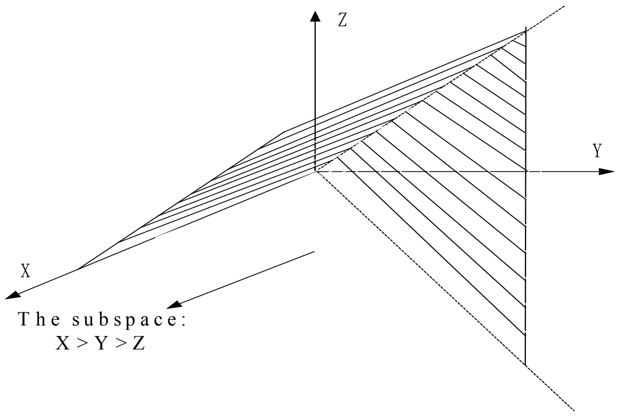

Fig.11: One of the 6 (3!) subspaces in a 3-dimension solution space

As a gradient descent algorithm, BP can hardly train a neural network against the trap of local optimum [9]. Some research on improving BP algorithm to globally optimize neural network has been reported. The shared focus of these research is to find a region in the solution space where there happens to exists a global optimum of neural network.

In this Thesis, a method named GOT proposed by Chen L.H. is referred [14]. The main point of this method is that a number of different sets of initially weights are randomly set in order that some of them may lie in the region that comprise a global optimum. Then these sets of weights are adjusted using BP algorithm. More sets of weights should be introduced until certain criterion is satisfied. This whole process is represented below:

Step 1: Randomly select a number of initial sets of weights $\{W_i^0\}$ (i=1…n) and use a local optimization tool (i.e. BP algorithm) to find corresponding local optimal solutions $\{W_i^*\}$ (i=1…n). Discard any W_i^* that is the same as .any previous solution.

Step 2: Check if there exists a subset $\{W_{nj}^*\} \subset \{W_i^*\}$ (nj=n1…na) such that $E(W_{n1}^*) \approx E(W_{n2}^*) \approx .... \approx E(W_{na}^*)$ and $E(W_{n1}^*)$=min $\{E(W_i^*)\}$ if not go to step 3 else go to step 4.

Step 3: Randomly select one more initial set of weights, i.e. W_k^0, k=n+1, go to step 2.

Step 4: End the search and $\{W_{nj}^*\}$ are regarded as the global optima.

Different from a local-optimization algorithm, GA and PSO both belong to global-optimization algorithms. Both algorithms choose a large number of points in the solutions space and these points will evolve in a certain manner in order to search through the entire solution space. In GA, these points are called individuals, each of which stands for a set of weights of Neural Network. After being evaluated by a fitness function, some elite individuals may stay unchanged while other individuals will mutate itself or crossover with others. In PSO, these points are named particles and each particle is a set of weights. There is also a fitness function to evaluate each particle. All particles will swarm towards its local best particle and global best particle. This is the way that the entire solution space is searched. Since training processes of these two algorithms are similar, here only PSO algorithm will be detailed.

Instead of using the prototype of PSO, a revised and improved version of PSO, namely Multi-level PSO (MPSO) is described. Unlike particles in PSO, particles in MPSO are extended into two different levels, the one is a level of real particles and the other is a level of imaginary particles. The swarming process of MPSO is illustrated in the following figure:

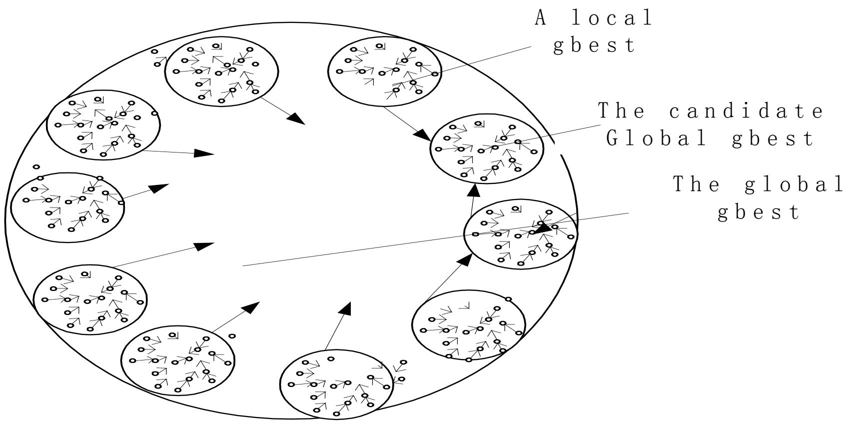

Fig.12: Multi-level Particle Swarm Optimization

In MPSO, the real particles are divided into M groups according to their location in the problem space. And each group is made up of N adjacent particles. The first level of particles is made up of the real particles in these M groups. The second level of particles, which have only M imaginary particles, are an aggregation selection of local gbests of m groups of first level of particles. Particles of this level are imaginary since the local gbest particle of a group is not a determinate particle. In the training process, different level of particles will swarm it own level.

In the first level of MPSO, standard PSO training is carried out in individual group of particles independently. Each group has its own local gbest. And there is no pbests. And particles of a group will swarm towards the group's local gbest only. This level of particles swarming is represented in the equations:

$$v_{id} = v_{id} + c_1 \times rand() \times (p_{gd} - x_{id}) \tag{18}$$

$$x_{id} = x_{id} + v_{id}\text{, i=1…n} \tag{19}$$

In the second level of MPSO, particles are made up of the local gbests of all the groups. Among all these local gbests, there certainly exists a best one of the bests, which we would call global gbest. And the local gbests, forming individual particles of the second level, will swarm towards the global gbest.

$$v_{jd} = v_{jd} + c_2 \times rand() \times (g_{gd} - x_{jd}) \tag{20}$$

$$x_{jd} = x_{jd} + v_{jd}\text{, j=1…m} \tag{21}$$

Where, g_{gd} is the global gbest.

Thus when the first-level particles in each group are swarming towards their local gbest, they are also swarming towards the global gbest relatively at the mean while. We can find out the actual adjusting functions of particle are:

$$v_{kd} = v_{kd} + c_1 \times rand() \times (p_{gd} - x_{kd}) + c_2 \times rand() \times (g_{gd} - x_{kd}) \tag{22}$$

$$x_{kd} = x_{kd} + v_{kd}\text{, } k = 1, \ldots\ldots, m \times n \tag{23}$$

But what really need to be computed are equations (7) (8) (9) and (10). Thus the total computation time is reduced a lot. What's more, in the training we do not let all particles in the first level reach the global gbest. We set the speed that a normal particle swarms to local gbest much higher than the speed of swarm of the particles in the second-level. So the first-level PSO will end much earlier than the second-level PSO. Only the local gbests will swarm all the way until they reach the global gbest and the other particles will die when they reach their local gbests.

In the above equations, if each particle stands for an entire collection of weights of Neural Network, MPSO can be used to train the Neural Network.

3.3.3 Structure of M^2PSO Network

When the number of nodes of Neural Network rises, the solution space of Neural Network expands in algebraic order, but the number of particles will rise in geometric order. Thus when the number of nodes rises to a certain number, the computation of PSO may become excessively expensive.

What's more, after deeper study of the structure of 3-layer feed-forward Neural Network in Fig.13, it can be noticed that the transfer functions between the nodes of two layers were of one same type, i.e. tansig between the input and hidden layer. This is necessary to meet the requirement of the traditional Neural Network training algorithm. However, since PSO is an algorithm that totally different from such algorithm, there is no such requirement on the transfer function. Therefore, different nodes can adopt different transfer functions in order to obtain a better overall performance.

Upon the above two considerations, a Multi-level--Multi-step PSO-Network (M^2PSO-Network) is proposed. The structure of M^2PSO-Network is shown in Fig.13:

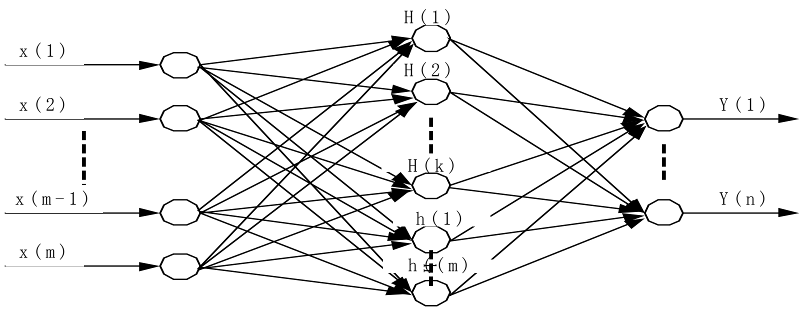

Fig.13: M^2PSO-Network

The difference between fig.7 and fig.10 is that the nodes h(m+1) to h(n) of hidden layer in fig.7 have been deleted and other hidden nodes, H(1) to H(k) have been added into fig.10. The number m is an empirical value that is obtained through a series attempt ion. Of course m can be set to no less than n, however there may be some redundant nodes, which will rapidly lower the computation speed. The number k is the number of basic mathematic functions and therefore can be a fixed number.

The transfer functions of the added-in hidden nodes are generally basic mathematic functions. These functions include power function, exponential function, logarithm function and trigonometric/anti-trigonometric functions. In this paper high-order power functions and anti-trigonometric are not adopted for the purpose of a tradeoff between the precise and the efficiency.

3.3.4 Training process of M^2 PSO-Network

The training process of M^2 PSO-Network is also different from that of neural network. In the training of neural networks all nodes in the hidden layer are symmetric and thus the weights are adjusted in the same training step. But in the M^2 PSO-Network the nodes in hidden layer are no longer symmetric since the adoption of different transfer functions. A comparison between the standard transfer functions in neural network, i.e. the tansig function, and the basic mathematic functions can lead to a conclusion that the standard neural network transfer functions are finer than the basic mathematic functions. In this sense, the basic mathematic functions of the hidden nodes aims at catch the coarse but more general part of the desire output while the standard neural network transfer function are used to catch the fine part the desired output. Therefore, the hidden-layer nodes with the standard neural network transfer function are given the name fine nodes while the other nodes in hidden layer are called coarse nodes.

Based on the above discussion, the nodes of M^2 PSO-Network are trained one after the other in a number of steps. The number of steps equals the number of nodes in hidden layer. In the first step, the first node in hidden layer is trained. The sampling outputs of the first nodes are just the desired output of the whole network. And after this step of training, the output of the first step of training obtained and its error against the desired output is calculated. This error will be used as the sampling output for the second node. Then the second step is carried out aiming at eliminating the error between the graph of real output and the output of the first step training.

For the target output $Y = f(x)$, where $f(\cdot)$ is a complicated function, however that's for sure $f(\cdot)$ can e divided into two parts, one is the simple part that can be model using basic mathematic function and the other part is the real complicated part that is supposed to be modeled using the more complex functions such as the standard neural network transfer functions. Then $f(x)$ is rewritten as $f(x) = p(x) + q(x)$, where $p(x)$ is the simple part while $q(x)$ stands for the complicated part. Therefore the complicated training process has been divided into two steps and been done more efficiently. The decomposition of function is shown in fig.14:

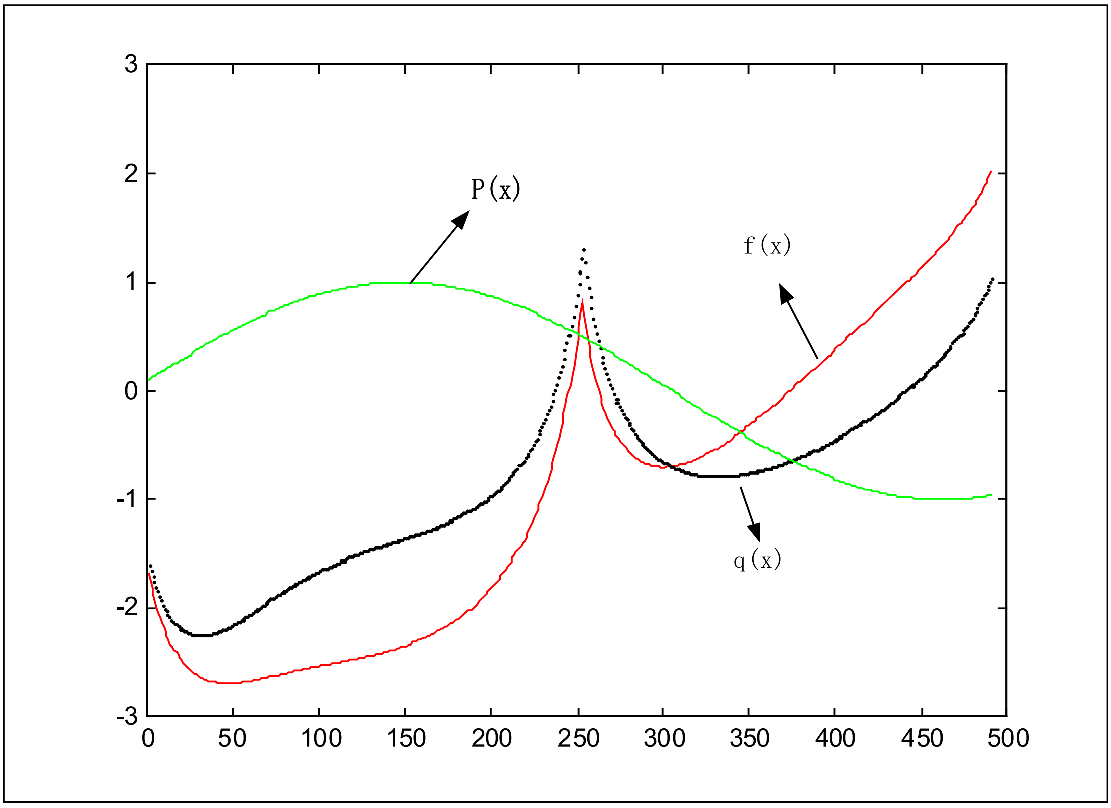

Fig.14: Functional decomposition of M^2 PSO-Network

Chapter 4

Simulations

In this chapter, we will set up a new software platform and design a series of simulations to validate the feasibility of FAMAC and make attempts to further better its performance.

4.1 Simulation Facilities

All simulations in this thesis were done with the MATLAB software. The simulation programs were run on a computer of Pentium III 1.0 GHz processor with 512 MB RAM. The computer operation system is Microsoft Windows 2000 Professional. All the simulations are programmed using Matlab software (Version 6.1).

4.2 The Simulation platform for FAMAC

4.2.1 General description of Platform

The environment of soccer game in reality is very complicated as there are a large number of factors, such as field, human and weather factors that may affect a match. For the convenience of research, it is not practical or necessary to count all these factors in the simulation environment. Instead, considering the focus of current research, the simulation environment only needs to represent some principles of a real working environment or game field. However, for the sake of further research, the simulation environment should be flexible enough to be extended to a complicated level, as the future research may require.

According to the above concerns, we set up a new simulation platform, namely Counterwork-Platform. This platform involves 3 parts: environment, agents, and game coordinator. The environment part contains the information about spatial dimension of the environment, reactions of environment on objects actions on it, and other all fixed stuffs within the environment. The part of agents' part includes the dimensions of agents, individual's behavior, and mutual affectations. The third part, game coordinator, however, provides definitions, facilities, and regulations about all sorts of games.

As it is designed to test of agent' ability cooperation, all individual agents are treated coequally in all aspects. This does not necessarily mean that all agents are the same everywhere at all time. In fact, just like playing lottery, some agents are lucky at some times with others being lucky later. But all agents hold an equal chance to be lucky.

A kind of such platforms, called Flag-Game, which is utilized to apply FAMAC, is shown in Fig 15:

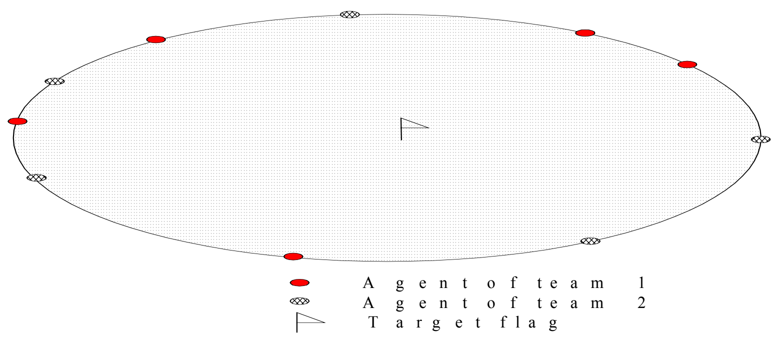

Fig.15: Simulation platform

In Flag-Game, there is a flag at the center of round planar field. Two teams of agents are placed at the edge circle of the plane. If any member of a team reaches the target flag first, that team wins the game. As it is an equal-chance platform, each agent is equally potent and the rules are fair to each other, each team holds an equal chance to reach the flag in the first place. But if cooperation is involved and two teams adopt different cooperation strategies, tings will be different.

Form the above discussions, we will find this platform a perfect one to evaluate and compare cooperation strategies. Generally, advantages of the Flag-Game platform include:

(1) Simplicity and standardization

The inputs and outputs are simple. The interface is simple and can be easily integrated with different kinds of cooperation algorithms. This means that little work needs to be done if new algorithm is to be tested on this model. So the researchers can concentrate on the design of learning algorithm.

(2) Flexibility

It provides a flexible game environment for multi-agents. Both simple and very complicated rules can be tested on this platform. And the rules can be changed independently of the cooperation algorithm that is evaluated. If researchers want to test a new algorithm in a complicated environment, they can extend this platform as complicated as they wish by counting in more environment factors.

4.2.2 Agents' Roles in Flag-Game and Their Cooperation

In a soccer game, it is not the offender's privilege to commit a goal. Some times, a warder or even a defender can do that. Of course, they are doing that randomly and they have their reason. This reason is: the right man does the right thing. In

Flag-Game, each agent also plays a relatively variable role. They do the right thing they are up to and, thus, behave in a way different form role assignment in [2].

Though agents in Flag-Game are not as potent as human players in soccer game, they have can perform some actions too. These actions may affect the environment and other agents in the environment as well.

● Definition of Actions (Roles):

In the simulation, an agent can perform any one action from the three actions below:

1, Offence, agent which takes this action will take the responsibility to reach the target as soon as possible.

2, Defense, agents taking this action is to hinder agents of the other team form reaching the target.

3, Ward, this is to help offender in the same team to resist the opponent's defense.

● Role Assignment:

1. Determination of fittest role of each agent through self-analysis through ILRU. The information of the environment and agents entered ILRU and ILRU outputs advisory information on the suitability of agents to perform any one of the 3 actions in this situation.

2. Competition in role (action) assignment.

   In the second step, the fitness values of agents are compared and role assignment is done in CCU based on some rules. The aim of this step is to let agents do the right jobs that contribute most to the team's goal.

As mentioned before, in Flag-Game, antagonistic teams are treated as equally capable without any discrimination on either team. And, to purely examine the cooperation strategy, it is assumed that there will not be any change or improvement on the ability of any individual agent. This assumption will ensure that only cooperation but not any other factor will influence the group performance of a team.

Then what, in such simple simulation environment, has made the cooperation possible and important? The answer is that each robot has its positional dominance. As can be observed in Fig.16, the agent that is near to the offender of opponent's team would prefer to acting as a defender because of its positional advantage over the antagonistic offender. On the contrary, the agent near to the antagonistic defender might be willing to be a defender for the similar consideration. But tings are not as simple as that, an obvious dilemma is that when an agent is close to both an antagonistic offender and an antagonistic defender that what should it do, offend or defend? They need reasonable cooperation to solve such problems.

According to the above discussion, agents must be able to recognize the roles of its antagonists and assign roles appropriately among themselves. In our research, this process is realized through communication, and competitive cooperation. By communication agents can exchange and collect information of nearby opponents and make judgment on the roles of opponent agents using such collected information. Then, in competitive cooperation, each agent will evaluate its own positional dominance and put forward its desired role as well as an evaluation grade of its suitability to take such a role. If the number of agents pursuing one role exceeds the number limit, evaluation grades of those agents are compared and agents with high evaluation grade will be considered to be fitter for the role.

Chapter 5

Results and Discussions

5.1 Test of PSO in Global Optimization for NN

Firstly, we carried out a set of simulations on optimization of Neural Network. These simulations are conducted to train Neural Network to track 4 functions which are actually mathematical forms of 4 neural networks. So, in extreme, neural networks in study can track these functions with zero tracking error when their weights match the parameters of those functions perfectly. And this perfect match will result in global optima for neural networks.

Thos four objective functions are:

$$Y_j = A2_j * \tan sig(A1_j * x + B1_j) + B2_j \qquad \text{j=1,2,3,4} \tag{24}$$

Where:

$$A1_j \in R^{q\times m}, B1_j \in R^q, A2_j \in R^{n\times q}, B2_j \in R^n, \ (A1_j, B1_j, A2_j, B2_j) \in Subspace$$

The radius of solution space in our research is set to be 200 and the parameters of the 4 objective sets of weights $(A1_j, B1_j, A2_j, B2_j)$, j=1, 2, 3, 4 to make them at the out, inner and middle layer of the subspace. And $(A1_4, B1_4, A2_4, B2_4)$ is a set of parameters chosen randomly in the subspace. Thus, 4 sets of neural network weights were generated:

Table 1 Four sets of NN weights chosen as benchmark NN weights

	A1(1,1)	A1(2,1)	A1(3,1)	B1(1,1)	B1(2,1)	B1(3,1)	A2(1,1)	A2(1,2)	A2(1,3)	B2(1,1)
NN1	1.7	-12.5	4.3	-0.73	5.34	-9.7	7.7	-0.4	-11.3	10.2
NN2	-27.7	-42.5	2.3	11.4	-21.7	-1.34	30.5	-14.5	7.8	5.4
NN3	21.4	-59.8	33.5	-120.6	-67.9	11.7	-19.1	42.7	18.5	67.4
NN4	52.7	-7.5	29.7	34.8	4.2	-7.3	-12.4	5.7	-0.35	5.7

In the simulation, Neural Network is trained with 3 methods: BP (GOT), PSO, and MPSO. All 3 trainings are allowed to run for a long time enough to get their best performance. The training result is shown in table 2:

Table 2The result of neural network training using 3 different methods

Training Algorithm	Training Error				Distance to Global optimum				Type of Optimum
	NN1	NN2	NN3	NN4	NN1	NN2	NN3	NN4	
BP(GOT)	0.26	5.43	0.13	0.86	24.23	87.63	191.2	65.73	Local
PSO	0.11	0.85	0.32	0.013	1.62	2.04	0.87	1.14	Local
PSO	0.0002	0.007	0.012	0.004	0.02	0.01	0.03	0.009	Global

The results are also represented in box plot format (Chambers et al., 1983) to visualize the distribution of different simulations. The 3 box-plots represent the 3 methods respectively:

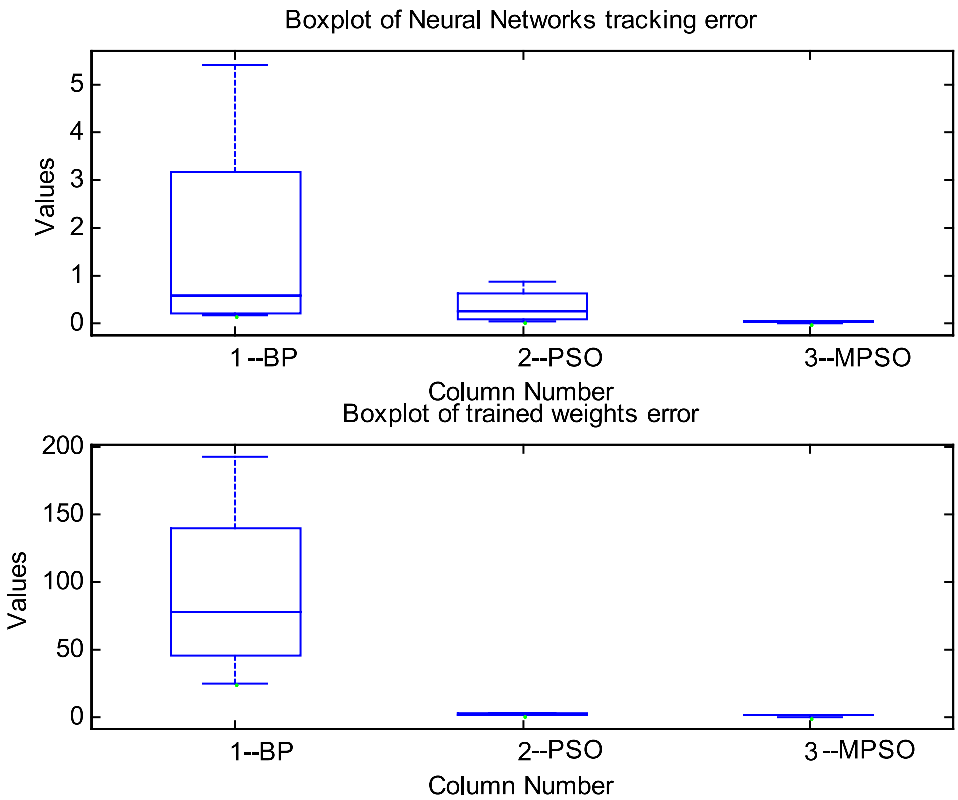

Fig.16: Box plot of training results

More detailed results of the first benchmark training are shown in fig.17 to fig.18:

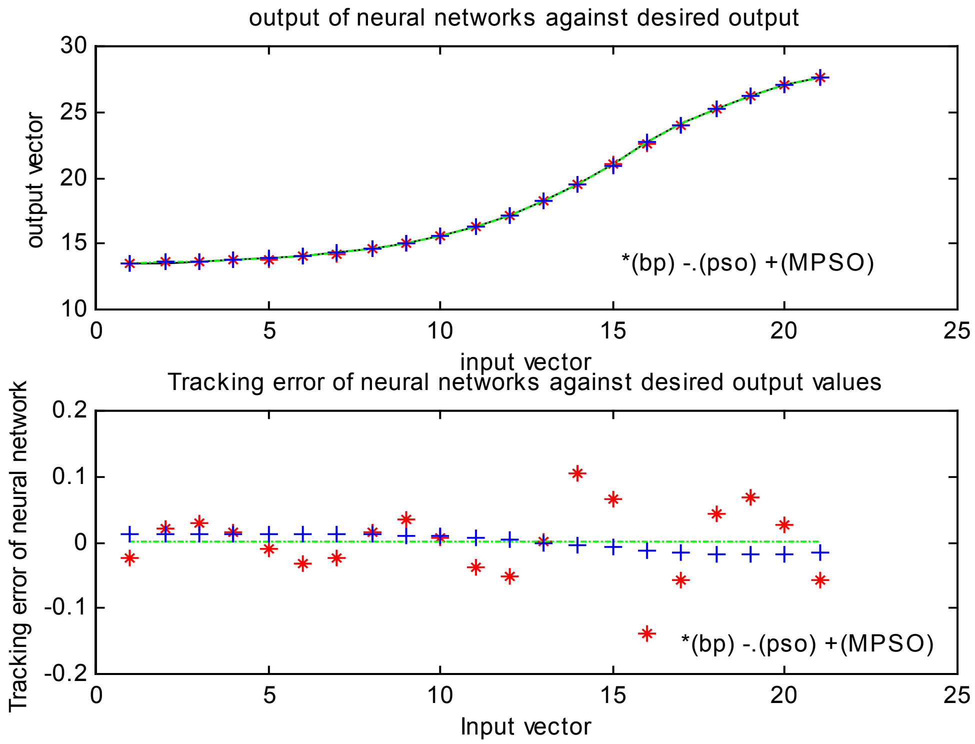

Fig.17: Outputs of trained Neural Networks and the tracking error

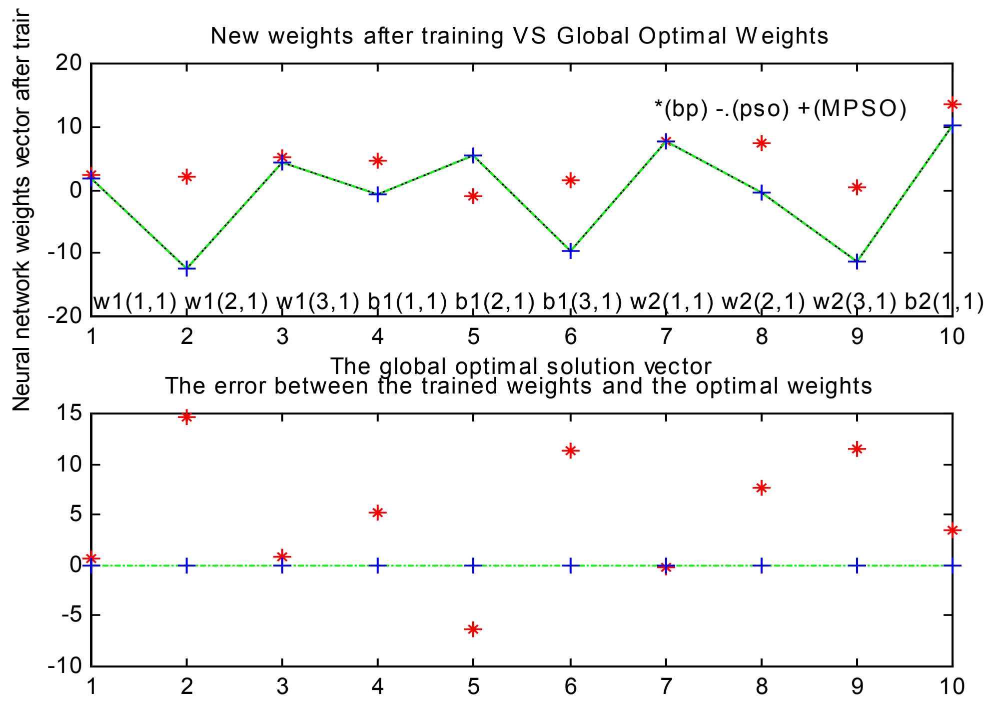

Fig.18: Weights of trained Neural Networks and the error against benchmark weights

From table 2 and fig.16, it can be observed that the tracking error of Neural Networks trained by PSO method is significantly smaller than that by BP. Moreover, when extended to MPSO, the tracking error has been further cut down. As can be seen in fig.18 and fig.19, difference between the final global best and the global optimum is minor and can be neglected in most circumstance.

Though BP is faster than PSO and thus can start for many times, it still cannot escape the local trap. The PSO, on the contrary, is slower but is capable of searching through the whole solution space and thereby can escape the local optima. And strange enough, even if BP are restarted the same number of times as the number of particles in PSO, it still cannot reach the global optimal. How to explain this?

In this section, we will take a brief and inner look at the process of neural network optimization. Fig.19 shows the tracking error of Neural Network in a one-dimensional solution space:

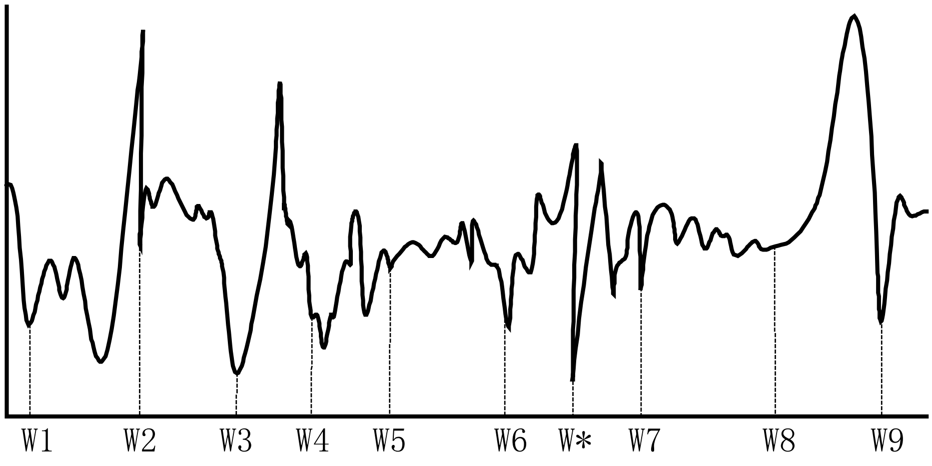

Fig.19: Tracking error of Neural Network in the solution space

Assume this curve for a one-dimension solutions space. If a gradient algorithm, for instance, BP algorithm, is adopted to explore the globally nethermost point, W*, due to its intense to continuously *go down*, this algorithm will tend to get stuck at the nearest concave point, which is a local nethermost. There are so many racks along this curve, such as w1, w2, that even we try for tens of times, we may still not be able to reach the target point. But two particles---the end points of this curve---are chosen and a global searching algorithm is adopted to search the globally nethermost point. Though every single particle also tends to rest at a local nethermost point, the one in a position higher than the other will have to leave its rest and search ahead towards the other particle and cannot get rest until it reached a position that is still lower. In this manner, every point in the curve will be searched. This is the difference between a global optimization algorithm and local optimization algorithm.

5.2 Performance of FAMAC in static Cooperation

After each round of game, the simulation result is analyzed using fuzzy logic rules. The result of analysis is fed-back to neural networks for neural networks' training. Fig.20 shows how a membership function will adjust itself to fit the dynamic environment:

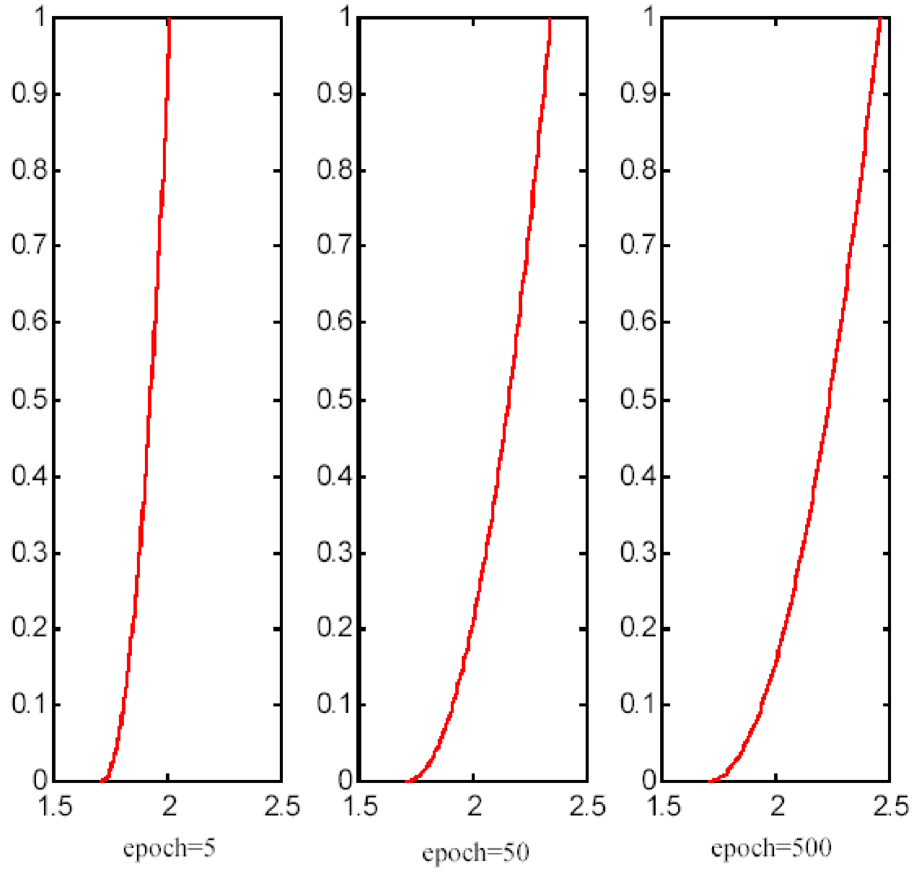

Fig.20: The membership function adjusting itself to the environment during simulation

To verify the effects of FAMAC on multi-agents system, in this part, static cooperation is applied to the multi-agents system. By the term static cooperation, we mean that the multi-agents system is required to cooperate at the starting point and after that there will be no cooperation any longer until a new round of game starts. Oppositely, in dynamic cooperation, agents will continuously react to the dynamic environment throughout the game.

Numbers of bouts has been simulated. Here, Neural Networks in ILRU are trained by BP algorithm. At the beginning, 100 bouts of simulation were carried out. Results of simulation were saved into the database of IAU and were analyzed. In succession, ILRU is trained using this database. Once the training of ILRU is successfully done, FAMAC is upgraded with the new IAU and ILRU and another 100 bouts of simulation were made. This process cycled and the performance of FAMAC in every 100 bouts of simulation is compared. A full record of this training in totally 2000 bouts is shown in Fig.21:

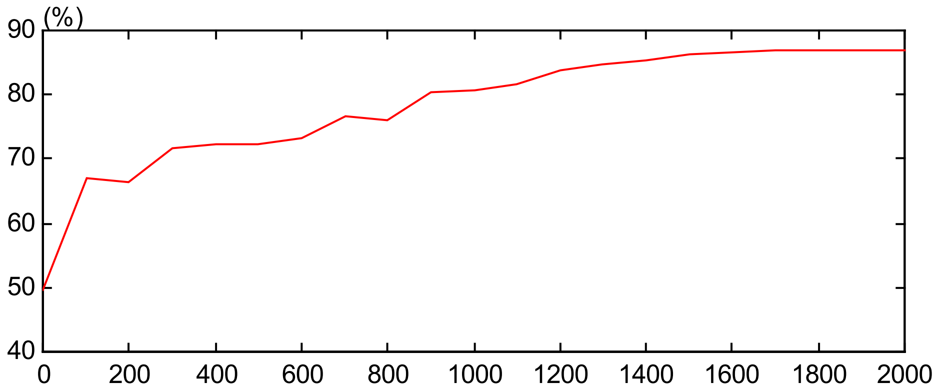

Fig.21: Performance of FAMAC with respect to training

At the beginning, as both teams choose to cooperate randomly, two teams got tied; each has a 50% chance to win a round of match. However, with training going on, significant progress in the performance of the team facilitated with FAMAC has been observed. A highest rate of success of 86.75% appeared in the end of 2000 bouts.

5.3 Comparison of M^2PSO- Network and Neural Network in FAMAC

The simulations in this section are targeted at enhancing the advantages and significances of FAMAC on current base of BP-Neural Networks.

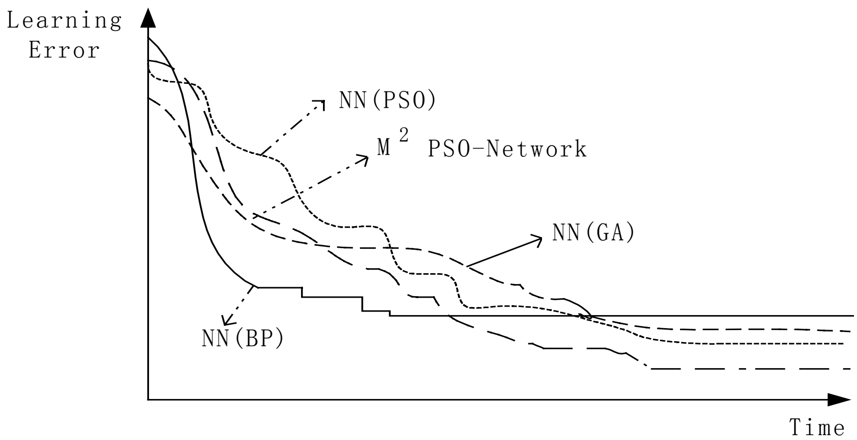

Fig.22: Comparison of learning performance between NN (BP/GA/PSO) and M^2PSO

As shown in Fig.22, because of the property of gradient decent, the tracking error of BP training drops much faster than any other methods. With training process going on, the decrease of BP tracking error slowed down quickly and finally no decrease can be observed after it has reached a local optimum. GA, due to the large number of individuals, presented a smallest value of tracking error at the beginning among all methods. However after that, the decrease of its tracking error is neither rapid nor lasting. While in PSO training for Neural Network, the tracking error drops much more slowly than BP method. However, this drop process lasted for a much longer time than BP. So though make little improvement in tracking performance, long time accumulative reduction leads to lower tracking error than BP. Considering the speed, M^2PSO-Network is fasted than GA and PSO and is slower than BP. The decrease in its tracking error is much more lasting than any other method.

Five Simulations, each of which comprising 1000 bouts of game, are carried out to evaluate the agents' ability to think while working and their adaptability to the dynamic environment that changes continuously all the time. In the 1st simulation there are no intelligent cooperation in both teams. In the following 4 simulations, FAMAC realized by BP-trained Neural Networks, GA-trained Neural Networks, PSO-trained Neural Networks and M^2PSO-Network are implemented respectively.

Table 3 Results of 1000 matches before and after training

Training Method	Goals of our team	Goals of opponent	Rate of Win/Lose
Untrained	487	513	0.95
Neural Network (BP)	874	126	6.94
Neural Network (GA)	891	109	8.17
Neural Network (PSO)	907	93	9.75
M^2PSO-Network	924	76	12.16

This comparison is not a straight one since each method only competes against a same third-part random cooperation strategy. Table 4 show the result of a straight comparison of FAMAC using M^2PSO-Network and FAMC using Neural Network:

Table 4 Direct comparisons between M^2PSO and PSO/BP

Simulation		Bouts of win (of 600 bouts)	Win ratio	Value of Win/Lose
1	Team A (M^2PSO-Network)	386	64.33%	1.459
	Team B (BP)	214	35.67%	
2	Team A (M^2PSO-Network)	352	58.67%	1.419
	Team B (GA)	238	41.33%	
3	Team A (M^2PSO-Network)	332	55.33%	1.239
	Team B (PSO)	268	44.67%	

5.4 Dynamic Cooperation of FAMAC with M^2PSO-Network

Further simulations were carried out on the dynamic cooperation of multi-agents system. In the dynamic cooperation, agents were required to cooperate continuously from the beginning to the end of one bout of game. 6 matches were simulated between two teams. In these simulations, in each step, the agents can obtain and analyze their new situations in the environment and exchange their roles for better performance. To keep the size of the database so as not to slow down the learning process, based on a

First-In-First-Out (FIFO) rule, old data in the database is regarded to be obsolete and be deleted from the database once the database is full.

Figures below illustrates an example of continuous steps cooperation process of the agents in a round of match.

Illustrations:

(1) In Fig.23, at the beginning, roles are intelligently assigned to the agents of our team according to the agents' initial states.

(2) In Fig.24, since a step of actions has been carried out, the states of agents have been changed and thus roles may need to be reassigned.

(3) In Fig.25, finally, agent of team A reached the target flag ahead of its opponent and won a goal in a round of match.

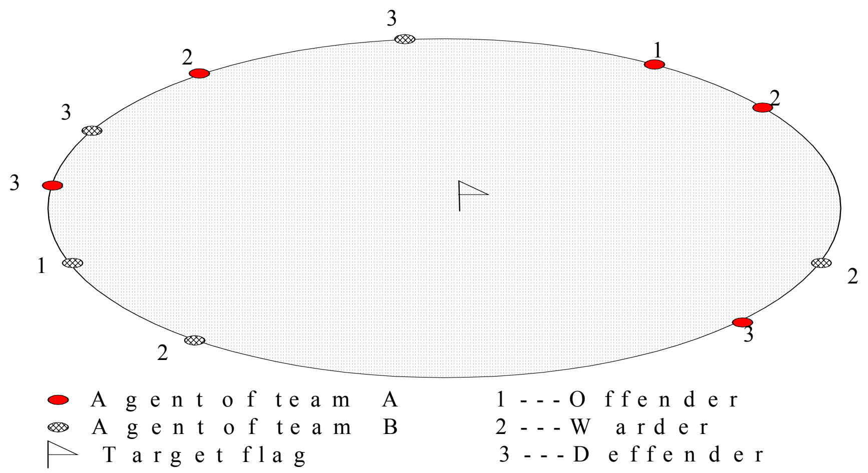

Fig.23: Step 1: Roles assignment according to initial status

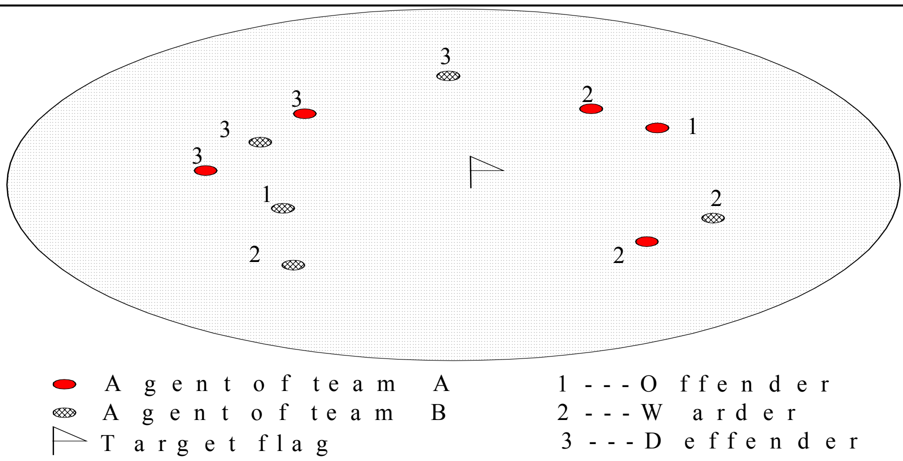

Fig.24: Step 2: Roles reassignment according to new situation

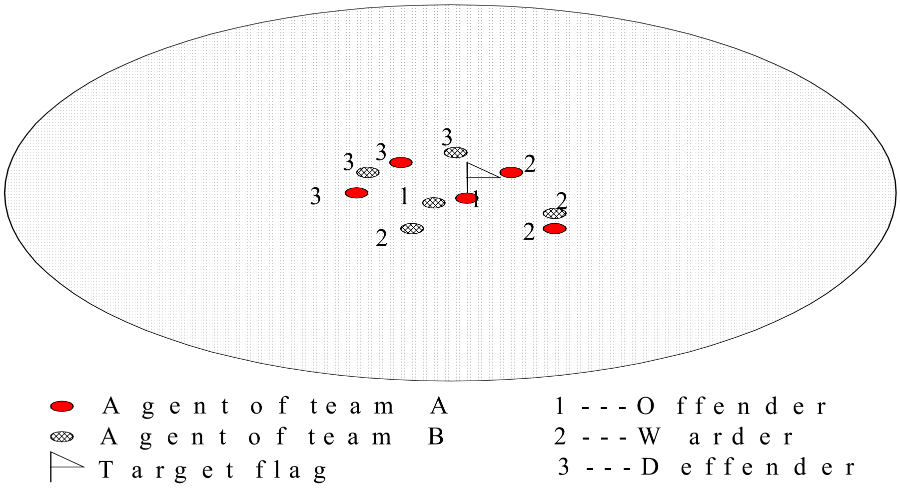

Fig.25: Final result: Team A reached the flag in the first place

The results of overall tests of dynamic cooperation are presented in table 5. With more steps of cooperation, rate of success of team A has increased to 98.33%. The reason is that with the number of steps of cooperation increased, the chance that team will run into proper cooperation strategies in all these steps has been greatly cut down. No one can flip a coin into face for a consecutive 100 times. Neither can agents choose to do right by chance all the way.

Table 5 Results of six rounds of matches after training

Round No.	Goals of team A	Goals of opponent B	Match Result
1	10	0	Win
2	10	0	Win
3	10	0	Win
4	9	1	Win
5	10	0	Win
6	10	0	Win
Overall	59	1	Win

Chapter 6

Conclusions

We have proposed a new cooperation strategy namely Fully Automatic Multi-Agents Cooperation (FAMAC) for Multi-Agents System. The FAMAC is made up of three units: IAU, ILRU and CCU. These 3 units correspond to the 3 functional units of human intelligent respectively: human analysis, human reasoning and global coordinator. Three different training methods, BP, GA and PSO, were applied to train the Neural Network in ILRU. And a Multi-level-Multi-step Network (M^2PSO-Network) is also put forward to further improve the performance of ILRU as well as that of FAMAC.

A number of important contributions have resulted from these works. First of all, the combination of ILRU and IAU has enabled agents to think, remember and analyze what happened, happening and to happen. All these abilities of agents have made MAC achievable. Secondly, M^2PSO-Network is introduced to take the place of traditional neural network for the sake of a better tracking performance. Comparison between them has proved such improvement.

Chapter 6 Conclusions

Through research in this thesis, some conclusions can be drawn:

(1) **It is effective and applicable to decompose and reproduce team intelligence using three intelligent units: ILRU, IAU and CCU**

ILRU is intrinsically a learning machine. With Neural Networks, ILRU does well in tracking objects whose information is explicit and rational. However, in a material world, not all information is so direct and explicit enough to be easily numerated. Such information need to be fuzzified and then transformed into numerical format. IAU is ace in dealing with this. CCU, as an irreplaceable unit, will solve the conflicts among agents and harmonize agents' behavior.

(2) **More training leads to better performance until it reaches its climax**

In real soccer game, a team is more likely to succeed with more extensive training. It's the same in the system of FAMAC. As we can see in the simulation results, the success rate rises continuously as the training time increases. However there is a threshold for this success rate. After this threshold point the success tare increases very slowly with respect to the increasing training time. This is caused by many factors. Future work is expected to increase the value of this threshold.

In general, a progressional method is developed in this thesis to generate a suitable cooperation for a team pursuing a common goal. And since this method is not critical about the abundance of information, it will have a wide range of usage.

As further research, both ILRU with M^2PSO-Network and IAU unit with fuzzy logic need improvements to fit for a much more complicated environment. And if we want to implement this method into practice, we need to speed the algorithm up to handle with the fast change of robots' and ball's positions and velocities.

References

[1] Dan L. Grec, David C. Brown, 'Learning by design agents during negotiation', 3rd International Conference on artificial Intelligence in design-------Workshop on Machine Learning in design, Lausanne, Switzerland, 1994

[2] Ferreira, J. R. da S., Cavalcanti, J. H. F. & Alsina, P. J., 'Intelligent Tasks Scheduler', Proceedings of the International Joint Conference on Neural Networks - IJCNN-99, Washington - DC, USA July1999

[3] Young D. Kwon, Dong Min Shin, Jin M. Won, et al., 'Multi agents cooperation strategy for soccer robots', Fira'98, pp 1-6, July 1998

[4] J.E. Doran, S. Franklin, N.R. Jennings and T.J. Norman, 'On Cooperation in Multi-Agents Systems', The Knowledge Engineering Review, 12(3), pp.309-314, 1997

[5] Hamid R. Berenji, David Vengerov, 'Learning, cooperation, and coordination in multi-agent systems', Intelligent Inference systems Corp. technical report IIS-00-10, 2000

[6] Paolo Pirjanian and Maja Mataric, ‘A decision-theoretic approach to fuzzy behavior coordination’, IEEE International S ymposium on Computational Intelligence in Robotics and Automation. November 1999

[7] Li Shi, Chen Jiang, Ye Zhen et al(2001), ‘Learning Competition in Robot Soccer Game based on an adapted Neuro-Fuzzy Inference System’, Proceedings of the 2001 IEEE International symposium on Intelligent control, pp.195-199, 2001

[8] Il-Kwon Jeong and Ju-Jang Lee (1999), ‘Evolving fuzzy logic controllers for multiple mobile robots solving a continuous pursuit problem’, Fuzzy Systems Conference Proceedings, pp.685 -690 vol.2, 1999

[9] Xu, L., Huang, J.Q., ‘Dynamic identification with neural networks for aircraft engines in the full envelope’, Journal of NUAA, Vol.33, No.4, pp. 334-338

[10] Kennedy, J., Eberhart, R., ‘Particle swarm optimization’, Neural Networks, 1995. Proceedings., IEEE International Conference on , vol.4, 1995 pp.1942 -1948

[11] Yoshida, H., Kawata, K., Fukuyama, Y., Takayama, S. and Nakanishi, Y., ‘A particle swarm optimization for reactive power and voltage control considering voltage security assessment’, IEEE Transactions on Power Systems, Volume: 15 Issue: 4, Nov. 2000 Page(s): 1232 –1239

[12] Salerno, J., ‘Using the particle swarm optimization technique to train a recurrent neural model’, Tools with Artificial Intelligence, 1997. Proceedings., Ninth IEEE International Conference on , 1997 Page(s): 45 –49

[13] Suzuki, K. Kitagawa, S. and Ohutchi, A., ‘Incremental evolution of weight modifiers for Neural Networks in cooperative behavior generation’, Systems, Man, and Cybernetics, 1999. IEEE SMC '99 Conference Proceedings, pp.121 -126 vol.6

[14] Chen,L.H.,’ A Global Optimization Algorithm For Neural Network Training’, Neural Networks, 1993. IJCNN '93-Nagoya. Proceedings of 1993 International Joint Conference on Neural Networks, Volume: 1 Page(s): 443 –446

[15] Berenji, H.R.and Vengerov. D, ‘Cooperation and coordination between fuzzy reinforcement learning agents in continuous state partially observable Markov decision processes’, Fuzzy Systems Conference Proceedings, 1999, pp.621 -627 vol.2

[16] João Sequeira, Pedro Lima, M. Isabel ribeiro, et al., ‘Behavior-based cooperation with application to space robots’, Proceedings of 6th ESA Workshop on ASTRA 2000,Noordjwikerhout, The Netherlands, 2000

[17] Shang,Y., Wah, B.W., ‘Global optimization for neural network training’, Computer , Volume: 29 Issue: 3 , March 1996 Page(s): 45 –54

[18] FUKUDA, 'Coordinative Behavior by Genetic Algorithm and Fuzzy in Evolutionary Multi-Agent System', Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on, 1993 760 -765, vol.1

[19] Xu,L., Tan, K.C., Vadakkepat, P., Lee, T.H.,"Multi-Agents Competition and Cooperation Using Fuzzy Neural Systems", ASCC2002, pp. 1326-1331

[20] Man-Wook Han and Kopacek, P., 'Neural networks for the control of soccer robots', Industrial Electronics, 2000. ISIE 2000. Proceedings of the 2000 IEEE, pp.571 -575 vol.2

[21] K. S, Narrendra, K, P, Parthasarathy, "Identification and Control of Dynamical Systems Using Neural Networks", IEEE Trans. on Neural Networks,vol.1,no.1,pp.4-27,March 1990

[22] Huang, J.Q., Xu, L., Lewis,F.L., "Neural Network Smith Predictive Control for Telerobots with Time Delay", Transactions of NUAA , 2001 Vol.18 No.1, pp. 35-40

Author’s Publications

The author has contributed to the following publications:

1. Xu,L., Tan, K.C., Vadakkepat, P., Lee, T.H.,“Multi-Agents Competition and Cooperation Using Fuzzy Neural Systems”, ASCC2002, pp. 1326-1331

2. Xu, L., Tan, K.C., and Vadakkepat, P., “A Fully Automatic Multi-agents Cooperation Strategy using M^2PSO-Network”, Submitted