Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Improving Coordination via Emergent Communication in Cooperative Multiagent Systems: A Genetic Network Programming Approach

Armin Tavakoli Naeini
Department of Electrical and Computer Engineering
Isfahan University of Technology, Isfahan, Iran
E-mail: a.tavakoli@ec.iut.ac.ir

Mehran Ghaziasgar
Department of Computer Science
Islamic Azad University of Majlesi, Isfahan, Iran
E-mail: gghmrm@yahoo.com

***Abstract*— The design and development of strategies to coordinate the actions of multiple agents is a central research issue in the field of multiagent systems. To address this issue, in our previously reported research we proposed a novel methodology based on Genetic Network Programming (GNP) to allow agents in the pursuit domain to autonomously learn an effective coordination strategy in order to achieve group behavior. GNP is a newly developed Evolutionary Computation (EC) technique whose genome is network structure.**

In this paper, we extend our methodology by allowing agents in the pursuit domain to autonomously learn communication. We consider autonomous and independently learning agents, and we seek to obtain an optimal solution for the team as a whole. We design a novel methodology for the emergence of communication between agents in cooperative multiagent systems based on GNP and we seek to obtain more coordination. Through simulations, we demonstrate that the proposed approach is effective in evolving communicating agents and furthermore a comparison is made between agents with and without communication in order to show that the emergent communication among agents is beneficial i.e., improves their coordination. In addition, we show the robustness of generated programs which is achieved as a side-effect of the capability of communication.

***Keywords*— Evolutionary Computation, Communication, Coordination, Genetic Network Programming, Multiagent Systems.**

I. INTRODUCTION

Multiagent Systems (MAS) are the systems where two or more agents carry out works autonomously depending on mutual interaction [1]. Recently many researchers have investigated the automatic design of intelligent multiagent systems using Evolutionary Computation (EC) techniques. Genetic Network Programming (GNP) is a newly developed EC technique, inspired from Genetic Programming (GP) [2]. While GP uses a tree structure as the gene of an individual, GNP uses a directed graph type (network) structure which has some distinguished abilities inherently. For example, GNP can make compact structures and furthermore overcome the low searching efficiency of GP [3]. Until now, GNP has been applied to various multiagent evolutions and its effectiveness was clarified by the comparison with GP and so on [4, 5]. In our previous paper [6], we proposed a new methodology based on GNP to evolve an effective multiagent coordination strategy among the four predator agents in the pursuit domain to capture the prey agent as soon as possible.

Pursuit domain is known as an easy-to-describe but difficult-to-solve domain in Distributed Artificial Intelligence (DAI). In this domain cooperation of agents is highly required, and design of an effective coordination strategy is challenging.

In the following, we are especially interested in using emergent agent communication, in order to obtain more coordination. It has been shown that emergence of communication is an important factor in coordinating the agents having a common goal in multiagent systems and various approaches have been proposed in the literature to model it [7-9].

In this paper, we extend our pervious methodology by allowing agents to autonomously learn if communication is useful and we seek to improve their coordination strategy. We propose a new approach to the emergence of effective communication between predator agents in the pursuit domain based on GNP.

This approach does not evolve a communication language between agents but instead it evolves the *use* of communication between agents which means that the agents are given sensory capabilities and hard wired messages and can learn if *using* communication is useful for them.

The contribution of this paper is threefold. First it demonstrates the capability of GNP architecture in evolving communicating agents in multiagent systems. Second, it presents a new approach for evolving communicating agents in the pursuit domain using GNP architecture. Third, it shows that the proposed methodology is effective and robust.

The rest of the paper is organized as follows. In the next section, the details of Genetic Network Programming such as the basic architecture, chromosome representation and GNP algorithm are described. Section III describes the pursuit domain which is the simulation environment. In section IV the proposed methodology and its setup is described. Section V presents the empirical results. Finally conclusions and future works are given in section VI.

II. GENETIC NETWORK PROGRAMMING (GNP)

In this section, Genetic Network Programming (GNP) is explained briefly. Basically, GNP is an extension of GP in terms of gene structures. The original idea is based on the more general representation ability of directed graphs than that of trees. Compared to GP, GNP can find solutions for

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

problems without bloat because of fixed number of nodes in GNP [10]. In addition GNP has built-in memory functions which implicitly and naturally preserve agent's actions as a chain of events [10].

A. Basic Structure of GNP

The basic structure of GNP is shown in Fig. 1. As shown in Fig. 1, an individual of GNP has a directed graph structure which is consisted of three kinds of nodes: "Initial Boot node", "Judgment nodes" and "Processing Nodes". In Fig. 1, the initial boot node, the judgment nodes and the processing nodes are denoted by square, diamonds and circles respectively. The initial boot node is the node which is executed only when GNP starts. Initial boot node is an origin for node transitions and has no functionality. The processing node is the smallest unit describing agent's actions to the environment whereas the judgment node is the smallest unit describing how to judge the environment the agent senses. Each judgment node has several judgment results, which corresponds to the number of its outgoing branches. Each processing node has only one outgoing branch. Judgment nodes and processing nodes have their "Function Label" in a library which is prepared by the system designer in advance. If we assume there are P types of processing nodes, their function label varies from 1 to P and when there are J types of judgment nodes, their function label varies from 1 to J.

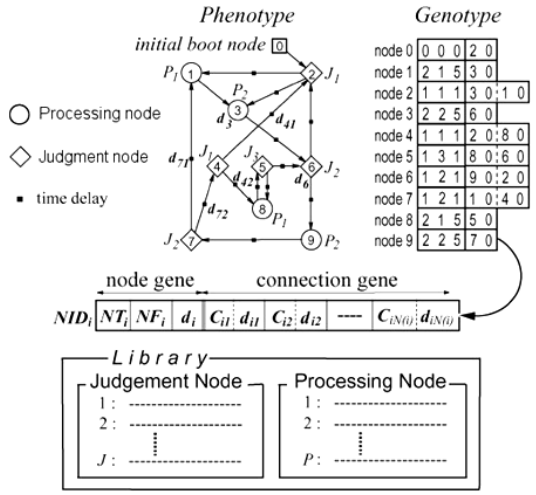

Figure 1. Basic structure of GNP [10].

B. Chromosome Representation

As shown in Fig. 1, GNP can be illustrated by its "Phenotype" and "Genotype". Phenotype shows the directed graph structure and genotype provides the chromosome of a GNP individual which is a set of the gene of nodes. In Fig.1, NID_i shows the gene of node i in which all variables are integers.

NT_i describes the node type, $NT_i = 0$ when the node i is initial boot node, $NT_i = 1$ when the node i is a judgment node and $NT_i = 2$ denotes a processing node. NF_i describes the function label (Judgment node: {1,2,…,J}, Processing node: {1,2,...,P}). C_{ik} indicates the node number to which node i's kth branch is connected. d_i means the delay time needed for processing and judgment at node i and d_{ij} means the delay time for moving from node i to j. These delay times have been introduced to model GNP like human brain which needs the time for thinking. When the accumulated time delay exceeds the "Time Delay Threshold Value" which is a parameter determined in advance, GNP fails to operat in the desired time [10].

C. Genetic Operators of GNP

Selection, mutation and crossover are carried out as genetic operators in GNP.

1) Selection: Selection is the operation of selecting an individual which serves as parents of the next generation, according to the degree of fitness. The fitness shows the quality of how each individual adapt itself to environment. In GNP, the commonly used selection operators in evolutionary computations can be used and their algorithms are the same as conventional evolutionary computations. In this paper, *tournament selection* and *elite preservation* are used.

2) Mutation: Mutation operates on only one network and new offspring is generated as follows.

- A network is selected using selection method.
- Some branches are selected randomly for mutation with the probability P_m.
- The selected branches are changed randomly and new offspring is generated.

Fig. 2, shows an example of mutation in GNP.

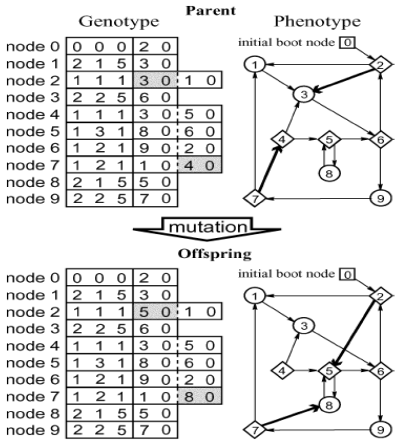

Figure 2. Mutation of GNP [10].

3) Crossover: Crossover is executed between two networks called parents and generates two offspring as follows.

- Two parent individuals are selected using selection method.
- Corresponding nodes with the same node number are selected as crossover nodes with probability P_c.
- Two parents exchange the selected corresponding nodes having the same node number and two new offspring are generated.

This type of crossover is called *uniform crossover*. Fig. 3, shows an example of uniform crossover in GNP.

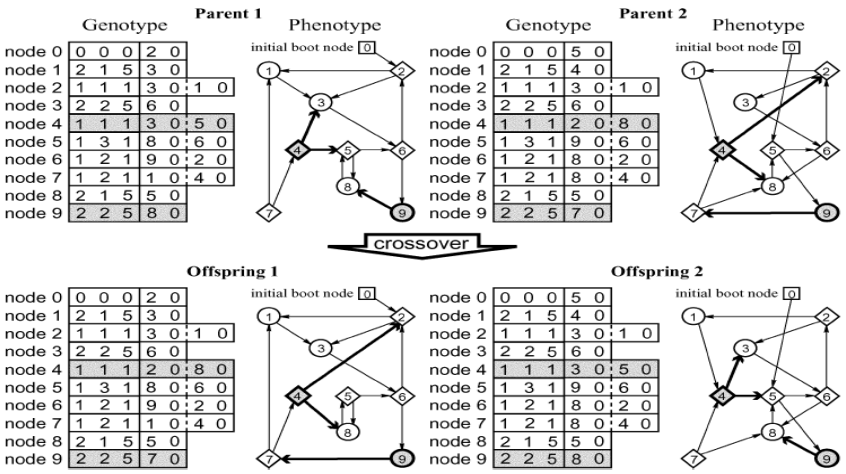

Figure 3. Crossover of GNP [10].

D. *Algorithm of GNP*

In this subsection, the algorithm of GNP is explained.

1) **[Initialization]**: Initialize the population with N_p randomly generated individuals (networks).
2) **[Fitness evaluation]**: Calculate the fitness value of each individual and find an elitist.
3) **[Genetic operations]**:
   3-1) Selection: Select parents for crossover by the tournament selection.
   3-2) Crossover: Apply the crossover operator to the selected parents.
   3-3) Mutation: Apply the mutation operator to the connection gene.
   3-4) Elite strategy: Preserve the elitist individual found in Step 2 or Step 5.
4) **[Replacement]**: Replace the newly generated population with the new generated one.
5) **[Fitness evaluation]**: Calculate the fitness value of each individual and find an elitist individual.
6) **[Termination condition]**: Terminate the algorithm if the specified condition is satisfied. Otherwise return to step 3.

E. *Execution of GNP*

GNP program starts from the initial boot node (IBN) at the beginning of its execution, and the node transition immediately moves to the node *i* which is the only connected node to IBN. If node *i* is a judgment node, its judgment function is carried out and the next node is determined depending on the judgment result and the node transition continues with this new node. If node *i* is a processing node, its processing function is done and the next node is *uniquely* determined because each processing node has only one outgoing branch in the genotype, and the transition stops at this node. Note that In the case of exceeding the "time delay threshold value", GNP stops its node transition, even during the turn. GNP resumes from the node where it stopped in the previous turn.

III. PURSUIT DOMAIN

The simulation environment is the pursuit domain. It is a common domain used in Distributed Artificial Intelligence (DAI) research to evaluate techniques for developing cooperation strategies [11]. Many variations to the original definition of this domain have been devised by changing key domain parameters [12]. The parameters which are used in our experiments are described briefly here. Our pursuit domain consists of a torroidal grid environment with one prey agent and four predator agents with a unique ID namely 0, 1, 2 and 3, as shown in Fig. 4. The initial positions of agents are randomly generated. Agent movements in this world, are possible in orthogonal directions only, and take the agents from one cell into another. The goal for the predators is to move so as to *capture* the prey.

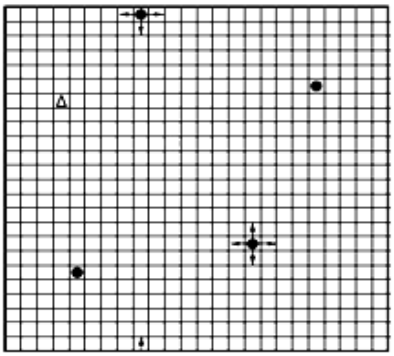

Figure 4. The torroidal pursuit domain. Triangle is the prey agent and solid circles are predator agents.

To capture the prey, the four predators must occupy all cells immediately adjacent to the prey which are called *capture positions* (Fig. 5). The prey moves randomly at 90% of the speed of the predators. The prey and predators agents move simultaneously and because only one agent can occupy a cell at a time, conflicts occur if one or more agents try to occupy the same cell (Fig. 6).

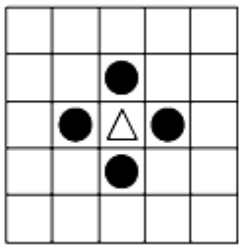

Figure 5. Capture configuration.

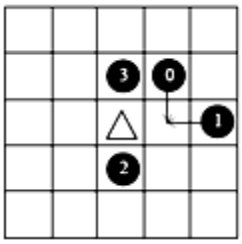

Figure 6. Conflict between agent 0 & 1.

Agents that try to move to the same cell are *bumped back* to their original positions. A predator may push another predator out of the cell it occupies if it decides not to move but it can not push the still prey. These conflict resolution mechanisms are performed by the environment rather than the agents.

IV. METHODOLOGY

The This section describes how we used GNP to evolve communicating agents in the pursuit domain especially paying attention to various nodes designed for our GNP methodology, fitness function and simulation parameters.

A. Processing and Judgment Nodes

Here is a summary of processing and judgment nodes shown in Table I. To evolve communicating agents, four judgment nodes namely {Pred0, Pred1, Pred2, Pred3} are added to the previously designed methodology in [6]. These nodes are introduced to allow the predator agents to communicate. The processing nodes are exactly the same as what are proposed in [6].

1) Processing Nodes

The processing nodes are *MN, MS, ME, MW* and *SH*. These nodes determine actions which predator agents can execute in the environment. When *MN* (Move North) is executed, the agent moves north, when *MS* (Move South) is executed, it goes south. *ME* (Move East) is for moving east and *MW* (Move West) is for going west. In addition when *SH* (Stay Here) is carried out, the agent stays still and does nothing.

2) Judgment Nodes

The judgment nodes are *CNC, CSC, CEC, CWC*, *FP*, Pred0, Pred1, Pred2, and Pred3.

These nodes have the decision functions dealing with the specific inputs from the environment. *CNC* (Check North Cell) checks the north cell of the predator agent and has three judgment results (Floor, Predator, Prey). *CSC*, *CEC*, *CWE* are designed to check south, east and west cell of the predator agent respectively and they have three judgment results as well. FP (Find Prey) finds the rough direction of the prey with respect to the predator and has four judgment results (North, South, East, West).

The Pred# judgment nodes in which # can be 0, 1, 2 or 3, request data from the predator agent whose ID is # if it can see the prey agent, i.e. it is within the range of its vision, then the Pred# node returns the rough direction from the calling predator to the prey which can be North, South, East or West and if the prey is not in its scope returns Null. So the Pred# nodes have five judgment results and thus they have five outgoing branches as it is shown in Table I.

Table I. The set of proposed GNP nodes.

Node Type	Node Function	Branch
Judgment Node	CNC, CSC, CEC, CWC	3
	FP	4
	Pred0,Pred1,Pred2,Pred3	5
Processing Node	MN, MS, ME MW, SH	1
Initial Boot Node	IBN	1

B. Fitness Function

One of the most important design decisions in developing an effective GNP solution is choosing a suitable fitness function. We used the same fitness function as in [6]. Table II, shows the pseudo code of this fitness function. For training purposes we evaluate each strategy (GNP program) by executing it on 30 randomly generated predator placements (training cases). The fitness which is assigned to a strategy becomes an average of the 30 training cases. In all of the training cases the prey is placed in the centre of the environment and moves randomly. In order to generate general solutions, i.e., solutions that are not dependent on initial predators' positions, the same 30 training cases were run for each member of the population in all generations. In each case, the strategy was run for 100 cycles (MaxCycles). A cycle is defined as a move made by each of the agents simultaneously.

In our experiments, the distance between agents is measured by *Manhattan distance* (sum of x and y offsets) between their locations. After each cycle of a training case, we measure the Manhattan distance between each of the agents and the prey, the sum of which is added to the current fitness value. After each simulation, we give the strategy further rewards for each agent that is occupying a capture position, and add yet a further bonus if the prey is captured. This is a highly effective fitness function because it rewards movement close to the prey, with bonuses for occupying capture positions and a further bonus for capturing the prey.

Table II. Fitness function pseudo code.

```
Initialize the positions of the agents according to the training case
fitness = 0
for cycle = 0 to MaxCycles
   move the predators according to the evolved code
   if the prey is not stationary move the prey (only 90% of the time)
   check to see if the moves are valid
   undo invalid moves
   commit valid moves
   foreach predator p, fitness += GridWidth/ManhattanDistance (p, prey)
endfor
fitness+=MaxCycles*GridWidth * NumberOfCapturePositionsOccupied
if the prey is captured  fitness += MaxCycles * GridWidth * 4
```

C. *Parameters Specification*

Table III shows the parameters we used to implement the GNP algorithm which was described in section II.D. These are the same parameters as our previous research in [6] because we want to compare the two performances.

Table III. Parameters specification

Parameter	Value
Population size	200
Tournament size	5
Elite size	1
Number of generations	50
Crossover probability (P_c)	0.9
Mutation probability (P_m)	0.01
Number of nodes per kind	1
d_i and d_{ij}	0

V. SIMULATION RESULTS

In order to demonstrate the ability of the proposed methodology in evolving communicating agents in the pursuit domain, we carried out ten runs, and also the simulation results are the total sum of fitness calculated on ten different environments. Note that we evolve one separate GNP strategy (program) for each predator.

For confirming the effectiveness of communication, the following pairs of cases are compared:

1) Communicating agents with finite visibility (i.e., the communicating agents that can perceive a square area around themselves which its side is a given value V and its center is the agent) are compared to non-communicating agents with finite visibility.

2) Communicating agents which can sense the entire environment are compared to non-communicating agents which can sense the entire environment.

Fig. 7 shows the average of maximum fitness value for ten independent runs when a predator agent's visibility is 4 (i.e., V=4) while Fig. 8 illustrates the average of maximum fitness value for ten independent runs when the predator agents can sense the environment completely.

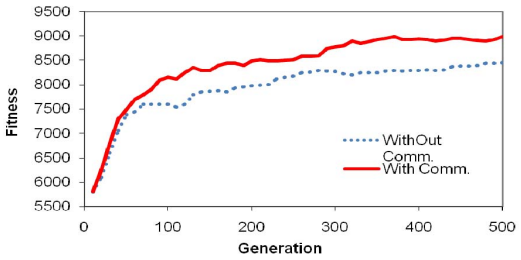

Figure 7. Average of maximum fitness value over 10 runs (V=4).

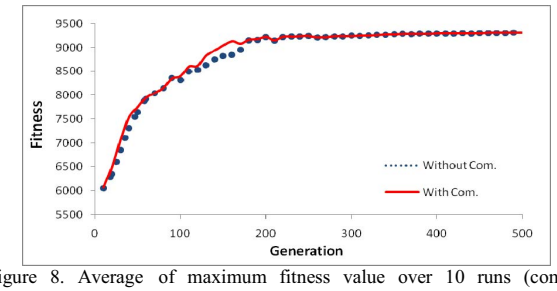

Figure 8. Average of maximum fitness value over 10 runs (complete perception of the environment).

Fig. 7 illustrates the superiority of communicating agents in terms of the fitness transition which means that when agent's view is limited the communication nodes are very useful to solve the pursuit domain efficiently by evolution.

The way that communication helps the predator agents to improve their coordination is that when the prey agent comes to the vision scope of a predator agent, other predators ask it the direction of the prey and with this information they can approach and enclose it quickly.

Fig. 8 shows that when the predator agents can sense the entire environment the results between the communicating and non-communicating agents are nearly similar, because all the predators can see the prey agent at all the time they don't need others' help and can make their decisions without need to communication with other agents.

In Table IV, MAX, MIN, AVG, SDEV show the maximum, minimum, average and standard deviation of the best fitness value of ten different runs, respectively.

Table IV. Best fitness value

	MAX	MIN	AVG	SDEV
V= 4 With Com.	9012	5810	8250	96.7
V=4 Without Com.	8503	5801	7805	453.0
Complete view With Com.	9179	6015	8745	9.238
Complete view Without Com.	9179	6015	8705	9.539

According to Table IV, the results illustrate that when agents' visibility is 4, GNP with communication produced good results with fairly low standard deviation meaning that the results are statistically significant. But GNP without communication produced results with high standard deviation that means the results are good only in some cases.

Table V, shows the maximum, minimum, average and standard deviation of success (capture) rate (%) of the pursuit task to 1000 randomly generated environments with the 10 evolved GNPs.

Table V. Success rate of GNP

	MAX	MIN	AVG	SDEV
V= 4 With Com.	61	41	52	6
V=4 Without Com.	53	15	41	15
Complete view With Com.	100	92	97	2.9
Complete view Without Com.	100	90	94	4.2

According to the results in Table V, when visibility value is 4, GNP with communication obtains higher success rate than GNP without communication. And also when the agents can sense the entire environment the results of communicating agents are better. As the standard deviation of communicating agents show, the results are significant. But when the predator agents can perceive the entire environment the results are fairly similar.

The robustness is an important feature of a program evolved by an EC technique. It is defined as the ability to cope with noisy or unknown situations. Here we tested the evolved GNP programs with communicating nodes, on 1000 *randomly* generated environments, different from the training cases, and they were successful. Thus we can say that our evolved communicating GNP strategies are robust.

VI. CONCLUSIONS AND FUTURE WORKS

In this paper a novel methodology for evolving communicating agents based on GNP was proposed and its effectiveness in improving coordination and its robustness was clarified in the pursuit domain. In addition it was demonstrated that performance of the communicating agents is superior to non-communicating agents in this domain. The most significant feature of this method is its ease for using in other cooperative multiagent domains. By assigning each agent in the domain a unique ID and a corresponding judgment node for communication in the GNP genotype, this methodology can be well extended for them.

In future we are going to enhance this approach by eliminating unnecessary communications between predator agents that take place when they get close to the prey.

The conflict resolution mechanism used in this paper was built in the environment. It would be interesting to design a GNP solution that enables the agents to resolve the conflicts themselves.

ACKNOWLEDGMENT

Our thanks to Professor Kotaro Hirasawa for giving the permission of using some of the figures of his papers.

REFERENCES

[1] G. Weiss, *Multiagent Systems*. Cambridge, MA: MIT Press, 2000.

[2] N. Vlassis, A Concise Introduction to Multiagent Systems and Distributed AI. Informatics Institute, University of Amesterdam, 2003.

[3] L. Panait, S. Luke, "Cooperative Multi-Agent Learning: The State of the Art", *Autonomous Agents and Multi-Agent Systems*, Springer-Verlag, Volume 11, pp. 387-434, 2005.

[4] J. R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA, USA, 1992.

[5] T. Murata, and T. Nakamura, "Genetic network programming with automatically defined groups for assigning proper roles to multiple agents", in *Proc. 2005 Genetic and evolutionary computation Conf.,* volume 2, Washington DC, USA, pp. 1705-1712, 2005.

[6] A. T. Naeini and M. Palhang "Evolving a multiagent coordination strategy using genetic network programming for pursuit domain", in *IEEE Congr. Evolutionary Computation, Hong Kong, 2008.*

[7] K. Hirasawa, M. Okubo, J. Hu, and J. Murata, "Comparison between genetic network programming (GNP) and genetic programming (GP)", in *Proc. IEEE Congr. Evolutionary Computation*, Seoul, South Korea, pp.1276–1282, 2001.

[8] T. Murata, and T. Nakamura, "Developing Cooperation of Multiple Agents Using Genetic Network Programming with Automatically Defined Groups", *Proc. of Late Breaking Papers in GECCO*, 2004.

[9] T. Murata, T. Nakamura., and S. Nag amine, "Performance of genetic network programming for learning agents on perceptual aliasing problem", *IEEE International Conf. on Systems, Man and Cybernetics* (IEEE SMC05), Hawaii, USA, 2005.

[10] T. Eguchi, K. Hirasawa, J. Hu, , and N. Ota, "A study of evolutionary multiagent models based on symbiosis", *IEEE Trans. on Systems, Man, and Cybernetics*, pp. 179-193, 2006.

[11] L. Panait, S. Luke, "Cooperative Multi-Agent Learning: The State of the Art", *Autonomous Agents and Multi-Agent Systems*, Springer-Verlag, Volume 11, pp. 387-434, 2005.

[12] P. Stone, and M. Veloso,. "Multiagent systems: A survey from a machine learning perspective", In *Autonomous Robotics,* volume 8, number 3, 2000.