Modeling Agent Communication in a Complex System as a Neural Net

Paul Darbyshire, *Member, ACM*

***Abstract*—Communication between cooperating agents has been shown in research to enhance the learning process of agents and their ability to complete tasks successfully in a group environment. However, an environment where we have many independent interacting components (agents), with each affecting the decision processes of the others, is essentially a complex system. Studying the behavior of the inter-agent communication in a complex system is difficult, particularly as the interaction between the agents may also affect the communication type and level. However, modeling the agent communication as a Neural Net may provide a convenient way to view such communication and further research it's effect on the learning process.**

***Index Terms*—Agent Communication, Reinforcement Learning, Multi-Agent Systems, Complex System, Neural Network**

I. Introduction

It's difficult to define a complex system, as many of the definitions concentrate on different facets that make a complex system what it is. Numerous definitions vaguely describe complexity as being midway between order and chaos, but a common thread running through many definitions is that a complex system is comprised of a large number of interacting components that interact in a nonlinear fashion. A combat environment aptly suits this definition as it comprises many components, all adapting to the environment and interacting in a nonlinear manner. Although combat systems can be modeled using linear simulations, nonlinear simulations employing multi-agent systems produce more realistic, if not quantifiable results. In constructing such a simulation, one of the more important facets is that of the agent communication during the simulation progression.

Agent communication is an important element in a multi-agent system, particularly when the communication is required in order for the success of the group. In a hostile environment, communication may also be vital for ensuring the agents survival, or the survival of the group. In order to understand and study the effects of communication on such a system, we need an effective way to model it. Communication in multi-agent systems has been studied in the literature for specific purposes such as achieving efficient cooperation, and modeling the semantics of communication languages. However, modeling the actual communication between agents in a complex adaptive system, is difficult, as the elements of the communication system could further be considered as a complex system itself.

Reinforcement Learning is often utilized to study agent behavior in a multi-agent system. It addresses the problem of how an autonomous agent that senses and acts in its environment learns by interaction with that environment [1]. While there is some question of the validity of using reinforcement learning to study multi-agent systems due to the underlying Markov assumption, nonetheless reinforcement learning algorithms perform well. However, while reinforcement learning allows us to model emergent behavior via simple agent interactions, it does not in itself provide us with a way of modeling the communication between the agents as they share learning experiences. We can utilize a Neural Network to implement a reinforcement learning simulating of a multi-agent system. Such an implementation has many advantages, particularly in allowing us to model the communication as a sum of weighted inputs to the agents, modeled as neurons in the network. This provides a convenient way to view such communication and further research it's effect on the learning process.

II. Background

The term complex system comes from the relatively new field of Complexity Science. Complexity science is an intersection of many diverse fields, including mathematics, AI, computing, engineering, sociology, biology, and many others. The emergence of complexity science has led us to seek different methods to simulate systems that are too complex for deterministic mathematical solutions. The best known definition of complexity is the KCS (Kolmogorov-Chaitin-Solomonoff) definition [2], which places complexity somewhere between order and randomness. However, a less rigorous definition defines a complex system as one in which an algorithm could describe the behavior, but where mathematical models do not provide efficient solutions to understand and then predict underlying phenomena [3]. Put very simply, complexity theory deals with systems that have many parts that interact in non-linear ways.

Many traditional approaches to simulation and modeling have been linear in nature, that is, processes and actions are directly proportional to, or related to input. As in Newtonian science, in such systems, cause and effect are usually separate [4]. In many simulations this is a valid model, but in complex systems that have many components that interact, cause and effect cannot be separated, and

Manuscript received July 24, 2010.

P. Darbyshire is with the School of Information Systems, Victoria University, Melbourne, Australia. (phone: 61 03 9919-4393; fax: 61 03 9919-4272; e-mail: Paul.Darbyshire@vu.edu.au).

positive non-linear feedback results. In these situations, a non-linear approach to simulation may prove more desirable. Traditionally, reinforcement learning is a technique particularly suited to, and often utilized, when we have autonomous agents that sense and act in their environment, and learn by interaction with the environment [1]. However, as indicated, the formal theory of Reinforcement Learning is based around a state-space conforming to a Markov assumption, and thus limited to a single agent. The practical applications of reinforcement learning utilizing a single agent are generally specialized and limited in nature, particularly with recent trends in AI research towards Agent Based Systems. An emerging area of research is Multi-agent Reinforcement Learning which is concerned with how an agent can learn to act optimally in an unknown environment through trial and error interaction, and in the presence of other adaptable agents [5].

Interaction between agents is central to the design of a Multi-agent System [6], and is a consequence of their plural nature. Interaction occurs when two or more agents interact through a series of events during which the agents are in contact. We normally think of interaction in terms of human communication (as in speech), which is usually modeled as message passing between the agents. In multi-agent reinforcement learning, the advantages of communicating agents has been documented by a number of research efforts and in all cases, by allowing agents to cooperate via communication, accelerated learning rates and a greater chance of group success is achieved. This cause-effect relationship has emerged in many research simulations [7], [8], [9], [10], [11]. The communication can be either implicit or explicit, however reinforcement learning does not specifically allow us to model the communication as an activity in itself.

Within the literature, most references for modeling communication are mainly concerned with the structure of the communication language, the semantics of the messages passed or the communication structures [12], [13], [14]. Yet given the importance communication can play in the success of a multi-agent system, it would seem important to be able to model the communication in order to study it's effect on the learning process. Artificial Neural Networks provide a convenient tool to describe the communication taking place within a multi-agent system in a more rigorous fashion.

Neural networks have been studied for some time in computer science, but work well when a non-linear algorithm best suits the problem. Such an algorithm essentially consists of a set of processing units (neurons) with a set of weighted connections between them with a set of inputs and a set of outputs. In a backpropogated neural network, feedback loops from the output back to the process units which allow the weights to be adjusted and the system learns [15]. We can model reinforcement learning utilizing neural networks [16], which in turn allows us to model the communication between the agents as the weighted connections between the neurons.

III. Motivation

The motivation for this research came from the desire to model the communication occurring in multi-agent system representing a military distillation. Military distillations represent a move away from the traditional constructive combat simulations, which are usually based on linear models of attrition. Such simulations often must embody the details of what they are trying to show, whereas the agent-based approaches attempt to produce the macro-level behavior by micro-level interactions [17]. At the conceptual level, these simulations provide insights into land combat by allowing the strategist to specify the local interaction between the combatants and then observe the emergent behavior of the system.

The distillation consisted of a simulation world represented by W, a continuous world where agent movement is determined by a vector consisting of angle and distance (based on a speed variable). The simulation is modeling a specific problem and utilizes a finite state and action space. Time is discrete, and for each time step $t \in \mathbb{N}$, each agent observes the world, determines its current state, and chooses an appropriate action to perform. All actions are performed at the end of each time step, with the sequenced randomized before execution to prevent bias.

There are two groups of agents operating within W. Y and X. $Y=\{y_1, y_2, \dots y_n\}$ represents the control group (Red Team), with the behavior of each $y_i \in Y$ governed by a rigid control paradigm.

The group of agents $X=\{x_1, x_2, \dots x_n\}$ is the focus of this research (Blue Team), and each $x_i \in X$ is controlled by a modified multi-agent reinforcement learning algorithm (utilizing Q-learning) given in (1).

$$\left\{\begin{array}{l} Q(s_t,a_t) \leftarrow Q(s_t,a_t)+\alpha\left[r_{t+1}+\gamma \max_a Q(s_{t+1},a)-Q(s_t,a_t)\right] \\ \forall\ m_{i,j}(t) \in M_j(t): \\ \quad Q(s,a) \leftarrow Q(s,a)+\alpha\left[r+\gamma \max_a Q(s',a)-Q(s,a)\right] \end{array}\right\} \quad (1)$$

In Eqn. (1), $Q(S_t, a_t)$ represents the current estimate of the action-value pair, action *a* in state *s* at time *t*. This is updated from the current value of $Q(S_t, a_t)$, the reward r_{t+1} received from taking action *a* while in state s at time *t*, (received at time *t+1*), and the maximum action-value (over all possible actions) of the resultant state s_{t+1} at time *t+1*. The values α and γ represents the values of the step-size, and discount factors respectively. Communication is a function of time, so we denote a message between two agents at time *t* as $m_{i,j}(t): x_i \rightarrow x_j$ then the total communication for the MAS at time *t* is then given by $M(t)=\{m_{i,j}(t)\}$. The set of messages received by x_j at time *t* is denoted by $M_j(t)=\{m_{i,j}(t)\}_{i=1\dots n, i\neq j} \subseteq M(t)$. The structure of each message $m_{i,j}(t) \in M_j(t)$ is of the form $\langle s_{t-2}, a_{t-2}, r_{t-1}, s_{t-1}\rangle$. Note that there is a small time delay between when an agent experiences a learning event and is able to communicate that to other agents in the network.

TABLE I: Attributes used to determine state space S. The first five attributes have two possible values, while the last two have three possible values

attribute description	possible values
enemy in visual range enemy wounded in visual range enemy in firing range wounded enemy in firing range	0 – no 1 - yes
agents health	0 – bad 1 - good
ratio of enemy to friendly in visual range ratio of enemy to friendly in firing range	0 – more friendly 1 – equal 2 – more enemy

The finite state space S is determined by seven discrete variables. Each of the variables targets an attribute that will be of interest to the agent as it pursues its goal. These state variables are described in Table I.

The number of possible states generated by the state variables is 288. Each state $s_i \in S$ is represented by a tuple such as <1,0,1,0,0,0,2> with each number being one of the possible values for the different attributes represented in the tuple. However, not all states are possible as they are not orthogonal. If the visual range is greater than the firing range, then if we have no enemy in visual range we cannot have any in firing range, and the two ratio attributes are meaningless. Eliminating impossible states and inconsistencies leaves a possible 68 states.

The reward function develops the agent behavior, but is dependent on the state space and available actions. The state space must allow the agent to discern or partition itself into states that are interesting in the pursuit of its reward. The available actions must allow the agent the possibility of attaining its reward. Given the combative nature of the simulation, the objective of each team is to engage in combat with the opposing team until one team is eliminated. The reward function for the Blue team must be designed to elicit this behavior. The simple reward structure used in this simulation to develop this behavior is illustrated in Table II.

TABLE II: Simple reward function. The reward at time t for action a_i in A chosen when in state s_i in S at time t-1

	reward value	action
$r_t\ (s_i,a_i) =$	0.5	damage to enemy
	-0.5	receives damage
	1.0	kills enemy
	0	otherwise

While the motivation stems from a desire to model the communication the specific simulation just described, the techniques described to model communication will essentially apply to any multi-agent system.

IV. MODELING COMMUNICATION

In trying to model the communication within the simulation described above, we take a *communication systems* view [6] of the communication component. That is, the communications, not the environment or agents becomes the focus of attention of the model. This is depicted in Figure 1.

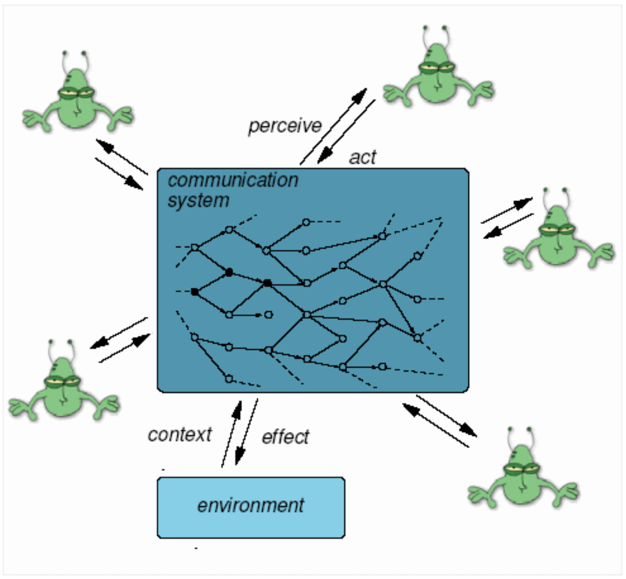

Figure 1: Communications systems view (adapted from [12])

The relatively simple setup of the communication system described in the previous section resembles that of a communication broadcast system. Thus all *n* agents are effectively connected to all other agents within the same group, forming in effect, a fully connected communication network. The agents themselves are situated in their environment and can move around, thus effectively changing the shape of the graph, however, this can be ignored as we are only interested in modeling the communication of the agents, thus the connectivity here is important, not the physical position of the vertices's. Such a static communication system of *n* agents could be modeled with a static graph $G = (V, E)$ where $|V(G)| = n$ and the set of edges, $E(G)$, would represent the communication channels [18]. In fact, the set-up as described could be represented by the *complete graph* K_n. Although the results of the communication in the experimental design indicated success, the current design itself is clearly not scalable for large values of *n* if the simulation is to represent a realistic combat model.

In a complete graph on *n vertices* (k_n), there are *n(n-1)/2 edges*, thus each agent could potentially be processing *(n-1)* communication messages at each distinct time period *t* in the simulation. Clearly if we wish the simulation to remain realistic, such a system is not scalable when modeling for large values of *n*. To avoid this, the agents could use directed messaging to only smaller lists of specific agents, or more commonly use a signal prorogation system similar to a broadcast but where the strength of the signal decreases over distance [12]. In the latter case, as one agent moves too far from another, the broadcast signal would get weaker, eventually fading altogether. Thus between two agents, a_1 and a_2, the strength of the message signal received by a_1, $V(a_1)$, is usually given by an equation of the form

$$V(a_1) = \frac{V(a_2)}{dist(a_1, a_2)}$$

or utilizing the square of the distance [12]. Implementing directed messages would be easier, but given the type of learning paradigm the agents are utilizing, the signal propagation method could be more suitable as the learning events taking place in an agents immediate neighborhood will be more relevant.

With this type of communication taking place, we can no longer model the communication as a static graph, since edges will appear and disappear as agents move in and out of range. We could however utilize a dynamic graph model. In this case we model the communication channels with a dynamic graph $G = (V, E)$ where $|V(G)| = n$ and where at time t $E_t(G) \subseteq E(K_n)$. Thus the communication channels are then represented by a series of timed edge changes $E_{t_0}, E_{t_0}, E_{t_0}, \dots$ [19]. The dynamic graph model is useful for studying the connectivity of the agents, but does not aid in the understanding of the communication that takes place between the agents. A more useful model to promote this understanding is that of an Artificial Neural Network (or simply a Neural Network).

If we adopt a neural network approach to modeling, then each of the agents $a_1 \dots a_n$ become nodes in the neural network. Each node has a number of weighted input vectors, corresponding to outputs from each of the other agents, as well a feedback loop to itself. This in effect describes a recurrent neural network model, or more specifically a locally recurrent neural net. Figure 2 Shows a high-level overview of such a neural network model of a system with only 5 agents.

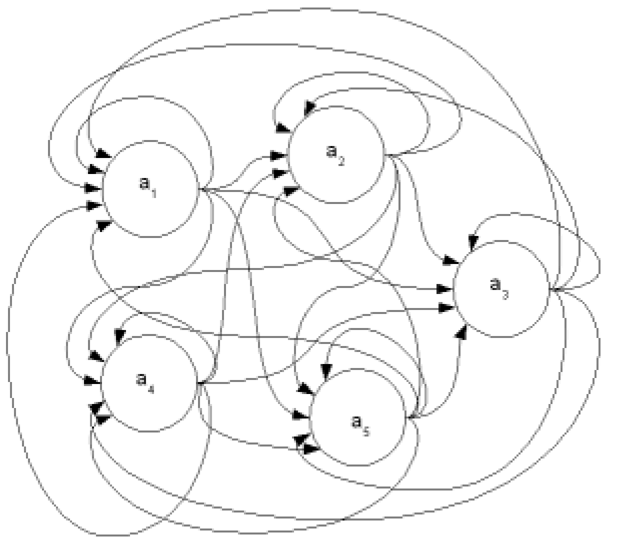

Figure 2: Overview of neural network model with 5 nodes

A more detailed diagram showing the detail of the feedback loop and weighted vectors for each node in the network is depicted in Figure 3. In order for the agent to learn, it perceives the state of its environment and initiates an action at time t. At time $t+1$, the agent then perceives its state again and possibly receives a reward for the action taken at time t. The learning matrix is then updated based on the original state at time t (S_t), the action taken at time t (a_t), the reward received at time $t+1$ (r_{t+1}) and the new state S'_{t+1} at time $t+1$. This is represented by the 4-tuple $<S_t, a_t, r_{t+1}, S'_{t+1}>$ on the feedback loop of the agent shown in Figure 3. This is enough information for the agent a_n to update it's learning matrix at time $t+1$. The feedback loop is normally represented internally in reinforcement learning, but can be represented as an external input vector in our model for completeness.

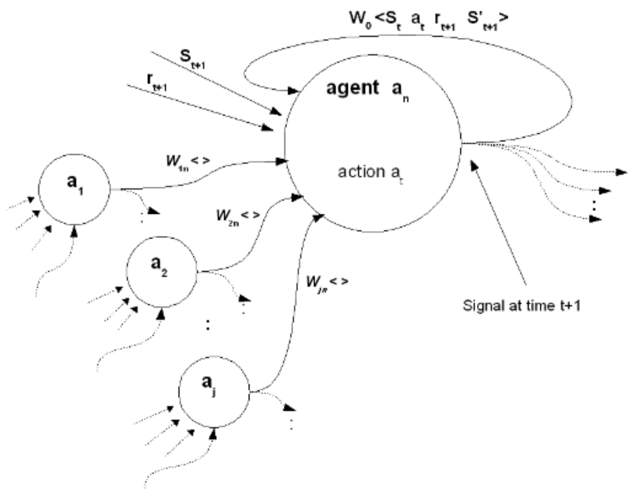

Figure 3: Detail of nodes in neural net model

On the weighted input vectors from agents $a_1 \dots a_j$ shown in Figure 3, the symbol <> is used to indicate a 4-tuple of the form $<S_t, a_t, r_{t+1}, S'_{t+1}>$ as previously described. The 4-tuples are sent by the different agents in order for the receiving agent to be able to effectuate multiple updates to the learning matrix. These form the input components of the weighted vectors. The weight component of a specific input vector to agent a_i, say w_{ji}, becomes Boolean-valued variable acting as a switch for the input vector

$$w_{ji} = \begin{cases} 1, & dist(a_j, a_i) \leqslant \Delta \\ 0, & dist(a_j, a_i) > \Delta \end{cases} \quad j=1,\dots,n, \quad j \neq i \quad (2)$$

If the distance between the two agents is less than a specific quantity, then the input vector is active, otherwise it is inactive. The fluctuation of w_{ji} corresponds with edge changes within the dynamic graph model. However, the distance between two agents isn't the only factor that determines whether an input vector is active. If agent a_j does not experience a learning event at time t, then at time $t+1$ the input vector from node a_j to a_i will not be active. Thus we can reformulate (2) as

$$w_{ji} = l_j \cdot d_{ji} \quad (3)$$

where

$$l_j = \begin{cases} 1, & a_j \text{ experiences a learning event at time } t \\ 0, & a_j \text{ does not experience a learning event at time } t \end{cases}$$
$$j = 1, \dots, n$$

$$d_{ji} = \begin{cases} 1, & dist(a_j, a_i) \leqslant \Delta \\ 0, & dist(a_j, a_i) > \Delta \end{cases} \quad j=1,\dots,n, \quad j \neq i$$

Using this model we can develop a model for the communication for a specific node a_i as

$$c = \sum_{j=1\ldots n,\, j\neq i} \left[l_j \cdot d_{ji} \right]$$

whereas, the communication levels for the group of nodes $a_1 \ldots a_n$ becomes

$$c = \sum_{j=1\ldots n,\, j\neq i}^{i=1\ldots n} \left[l_j \cdot d_{ji} \right]$$

V. Defining the Objective Function

One of the crucial elements to the successful implementation of the reinforcement learning algorithm by a neural network is the development of the Objective function, or cost function. The objective function is used to train a neural-net and is used to provide a measure of system performance. Thus the objective function can be seen as an error function providing an error measurement between the actual output of the net, and the optimal output. We usually need to minimize this function. However, this implementation of a neural network is based on an underlying reinforcement learning model, with the objective function governed by the underlying reward function. The reward function from Table II is governed by the simple algorithm

$$r = \begin{Bmatrix} 1 \\ 0.5 \\ -0.5 \\ 0 \end{Bmatrix}$$

depending whether the agent killed, damaged or was damaged by an enemy agent during the last time period. The reward function for an individual agent a_n is returned as part of the signal $\langle S_{t-1} a_{t-1} r_t S'_t \rangle$, where the reward received r_t at time *t* was for taking action a_{t-1} at time *t-1* while in state *S*, which resulted in new state *S'*. In this case, the object is to maximize the reward, as each reward results in an update to the agents state-action matrix, which in turn is used to decide which action to perform next based on selecting the action with the maximum state-action value.

Although the reward function is discrete, the signal produced by the reinforcement learning update function in (1) is not. We can use the difference between the action value function $Q(s,a)$, the value of taking action *a* in state *s*, and $Q^*(s,a)$, the optimal action value function as the error differential. It is this difference we need to minimize. Thus the overall cost function of the network at time *t* can be denoted as

$$E(t) = \frac{1}{N} \sum_{i=1}^{N} (Q^*(s_i, a_i) - Q(s_i, a_i))$$

where N denotes the number of nodes (agents) in the neural network, and $Q(s_i, a_i)$ is the value of agent i taking action a_i in state s_i. For the purposes of the cost function each agent will only need concern itself with its own feedback input, and not the input from the other agents, as we can assume each other agent is also attempting to minimize its own cost function.

The difficulty with such a cost function from a practical viewpoint is that in a multi-agent simulation representing a complex system such as the one described, the optimal policy $V^*(s)$, and hence the optimal action value function, $Q^*(s,a)$ are unknown. Additionally, given the complex nature of the system, they are not stationary as simulation time progresses. Thus the reward function offers the only way for each node in the network to pursue a minimization of the error, by attempting to maximize the reward.

VI. Conclusion

A multi-agent system is one in which there are many agents interacting with the environment, and each other in order to pursue a goal, sometimes a shared goal. This is representative of a complex system. The importance of communication between agents in such a system for achieving a common goal has been researched and demonstrated many times. Yet it is difficult to realize a model for this communication given the nature of the algorithms used to study these complex systems. Multi-agent reinforcement learning is a popular paradigm for studying these systems, by allowing us to observe emergent behavior of the system by defining the interactions between the agents. This allows us to model the behavior of the elements in the simulation, but it is difficult to model the communication between them concisely which can be a vital component for success.

We can model a reinforcement learning problem using neural networks in such a way that the communication between the agents is more easily modeled. In this paper we have outlined an overall method for achieving this, and briefly discussed the objective function of the system. Further work needs to be done in refining the objective function in terms of the underlying reinforcement learning reward function. Additionally, we presented a simple discrete weighting system as a function of distance between the agents which will evaluate to either 0 or 1 depending on a distance threshold. This assumes relevance of actions taken by agents based on locality. Further simulations need to be performed to determine if a weighting system based on a continuous distance function might be more beneficial.

Acknowledgment

Paul Darbyshire thanks Dr. Justin Wang from Latrobe University Department of Computer Science and Computer Engineering, for his initial comments which led to this paper.

References

[1] T. M. Mitchell, "Reinforcement Learning," in *Machine Learning*, Anonymous, Ed.: Mc Graw-Hill International Editions, 1997, p. 367.
[2] C. U. M. Smith, "The Complexity of Brains: A Biologist's View," *Complexity International,* vol. 2, 1995.
[3] P. Marcenac, "Emergence of Behaviours in Natural Phenomena Agent-Simulation," *Complexity International,* vol. 3, 1996.

[4] C. Lucas, "Complexity & Artificial Life - What are they?," 1/4 2000.
[5] R. Khoussainov, "Towards Well-defined Multi-agent Reinforcement Learning," in *Proceedings of the 11th International Conference on Artificial Intelligence: Methodology, Systems, and Applications*, Varna, Bulgaria, 2004, pp. 399-408.
[6] J. Ferber, *Multi-Agent Systems, An Introduction to Distributed Artificial Intelligence*: Addison-Wesley, 1999.
[7] M. Tan, "Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents," in *Readings in agents*, M. N. Huhns and M. P. Singh, Eds. San Francisco, CA: Morgan Kaufmann Publishers Inc, 1993, pp. 487 - 494.
[8] L. Nunes and E. Oliveira, "On learning by Exchanging Advice," *Interdisciplinary Journal of Artificial Intelligence and the Simulation of Behaviour,* vol. 1, p. 241, 2003.
[9] M. J. Mataric, "Using communication to reduce locality in distributed multiagent learning," *Journal of Experimental & Theoretical Artificial Intelligence,* vol. 10, pp. 357-369, 1998.
[10] C. Versino and L. M. Gambardella, "Learning real Team Solutions," in *Distributed Artificial Intelligence meets machine Learning, Learning in Multi-Agent Environments*. vol. 1221, Lecture Notes in Computer Science, G. Weiss, Ed.: Springer, 1997, pp. 40-61.
[11] P. Darbyshire and D. Wang, "Learning to Survive: Increased Learning Rates by Communication in a Multi-agent System," in *AI 2003: Advances in Artificial Intelligence*. vol. 2903/2003, Anonymous, Ed. Heidelberg: Springer Berlin, 2003, pp. 601-611.
[12] M. Nickles, M. Rovatsos, W. Brauer, and G. Weib, "Towards a unified model of sociality in multiagent systems," *International Journal of Computer and Information Science (IJCIS),* vol. 5, pp. 73-88, 2004.
[13] M. Ghavamzadeh and S. Mahadevan, "Learning to Communicate and Act Using hierarchical Reinforcement Learning," in *Third International Joint Conference on Autonomous Agents and Multi-agent Systems*. vol. 3 New York: IEEE Computer Society, 2004, pp. 1114-1121.
[14] P. Skrzypczynski, "Managing the Communication in a Complex System of Robots and Sensors," Poznan, Poland: Institute of Control and Information Engineering, 2004.
[15] D. Mandic and J. Chambers, *Recurrent Neural Networks for Predictions*, 1 ed.: Wiley, 2001.
[16] J. S. R. Jang, C. T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing*: Prentice Hall, 1997.
[17] C. Goldspink, "Methodological Implications of Complex Systems Approaches to Sociality: Simulation as a foundation for Knowledge," *Journal of Artificial Societies and Social Simulation,* vol. 5, 2002.
[18] D. G. Green, "Emergent behaviour in biological systems," in *Complex Systems - From Biology to Computation*, D. G. Green and T. J. Bossomaier, Eds.: IOS Press, Amsterdam, 1993, pp. 24-35.
[19] R. O'Dell and R. Wattenhofer, "Information Dissemination in Highly Dynamic Graphs," in *2005 Joint Workshop on Foundations for Mobile Computing* Cologne, Germany: ACM, 2005, pp. 1-4-110.