

Reducing Inter-Agent communication due to negotiation in Multi-Agent systems through Learning

Bradley Van Aardt
School of Electrical and Information Engineering
University of the Witwatersrand
Johannesburg
South Africa
b.vanaardt@ee.wits.ac.za

Tshilidzi Marwala
School of Electrical and Information Engineering
University of the Witwatersrand
Johannesburg
South Africa
t.marwala@ee.wits.ac.za

Abstract – This paper studies the effect that agent learning can have on inter-agent communication in a Multi-agent system. The agents are equipped with MLP neural networks to learn solutions to problems that have been solved through explicit negotiation and communication. We implement a test problem in the form of a Pursuit game, where the Multi-Agent system is a set of captor agents. The result is up to 44% fewer negotiation sessions with learning-enabled agents. The importance of learning, in terms of agent knowledge and overall system effectiveness is discussed.

I. INTRODUCTION

The concept of Multi-agent systems is an engineering paradigm that has been gaining momentum over the past years [1]. A particular form of Multi-Agent systems, Swarm based systems, have been successfully applied to a number of problems [2]

Swarm systems are popular because of the lightweight processing power needed for them. A particular advantage in some instances is the low level, or often no direct, communication between agents, especially when communication bandwidth is limited or costly [2]. In [3], for example, the time taken for a system during negotiation to solve a task for a particular planning scheme, GraphPlan, is noted. The amount of time can be seen to increase significantly on some problems. This is potentially very inefficient, especially if the agents face the same problem frequently. However, it is often considered an absolute fundamental attribute of agents to be able to communicate with other agents in the system to improve the overall performance of the system [4].

In this paper, a study is performed on the effect that learning has on communication between agents. It is reasoned that once an agent community has solved a particular problem successfully, there should be no reason to repeat all the steps that led to the solution. In other words, the agents should be able to re-use solutions that they have discovered previously. In the experiment described here, the agents first negotiate a solution for each problem they are presented. Each agent then learns the problem situation from its point of view, and the resultant action that it took after successful negotiation with other agents. Once they have learned the appropriate behaviour, they need not perform the task of negotiation for that situation, as long as the same conditions hold. It is reasoned that this can significantly cut down on inter-agent communication traffic.

II. BACKGROUND

A Multi Agent Systems

The multi-agent paradigm, in which many agents operate in an environment, has become a useful tool in solving large scale problems through a “divide and conquer” strategy [5]. The Multi-agent system is a distributed, decentralised system. The paradigm of individual entities collaborating to solve a particular problem that is beyond each entities own capabilities is a natural concept, and one that is proving to be very powerful in practice [6]. However, while the concept is easily understandable, the implementation is not trivial. There are many complexities and subtleties in these such as [5]:

- Decomposing and allocating problems to the agents
- Describing the problem to the agents
- Enabling communication and interaction among the agents

Decentralised systems, in the context of Multi-Agent systems, promise the following advantages [5][7][8][9][10]

- No single failure point, therefore greater robustness. Multi-agent systems have the capacity to degrade gracefully.
- Possibility of faster response times and fewer delays as the logic/intelligence is situated nearer to the problem domain.
- Increased flexibility to take account of changes occurring in the problem domain.
- Modularity and Scalability. Multi-agent systems can be increased in size dynamically according to the demands of the problem.

The problems associated with Multi-agent systems are [10][2][5][1]:

- Difficulty in measuring and evaluating the stability and security of the system.
- Excessive communication between agents can slow down the system. This is often countered by heavily restricting the amount of communication between agents.
- Possibility of getting stuck in non-optimal solutions of the problem, often due to the lack of global knowledge of the problem from each agent’s point of view.

- Most Multi-agent systems are built in an ad-hoc way since there is no absolute theory for these types of systems. There have been recent attempts however, to formalise the design of agent based systems, such as the Gaia Methodology [7]. However these are not yet in widespread use.

B Swarm Based systems

Swarm intelligence is a particular paradigm for multi-agent systems which emphasizes distributedness and agent simplicity. It is based on the observations of social insects in nature, such as ants, termites and bees [2]. Such insect societies are extremely organized, even though there is no central control or planning. Each agent in the system is programmed only to achieve its own Local Goal. The agent's behaviours in Swarms are very simple: the intelligence of the system is 'emergent' from the overall behaviours of all the agents in the system. The communication between agents is usually performed indirectly [9], by agents making changes to the environment, which other agents act upon. This is analogous to insects laying pheromone trails to food source, etc. Emphasis is therefore placed on reactivity in these systems. Swarm systems have been successfully applied to many problems, notably routing in computer and telecoms networks [2] and recently to a manufacturing control system [9].

The disadvantage of swarm based systems is that no agents actually have a global view of the problem to be solved. All agents are entirely focused on achieving their own Local Goals, whether or not these goals are to the benefit or detriment to the overall community. This can exacerbate the problem of the system getting stuck in local optima, or worse, cause the system to fail.

C Neural Networks

Artificial neural networks are inspired by the functioning of biological neurons in animal and human brains. Artificial neural networks differ from most other computing techniques in that they are able to learn arbitrary relations between sets of data presented to them. That is, rules are not explicitly programmed or set, but are learned from experience by the network [10]. The basic architecture of a neural network is also based on that of their biological counterpart: both networks consist of many simple elements, known as neurons, operating in parallel [12] [13]. The most widely used neural network models are the Multi-layer perceptron (MLP) and Radial Basis Function (RBF) networks.

III. HYPOTHESIS AND METHOD

Reference [11] states that the method of human learning when presented by a new task is to use rational reasoning to perform the decision making process behind solving the task. After using this deliberative approach, and we begin to "master" a task, we are able to perform it

"naturally", without explicitly performing rational reasoning. At this point, people use their pattern recognition skills to perform the task.

It is hypothesised that the human model of discovery and learning can be applied to agent strategising. An algorithmic discovery stage using negotiation between agents can be used on new problems. This will be the deliberative stage, where a problem solution is formulated. Once a strategy has been discovered and used, a neural network can be trained with the results. This is the Pattern Recognition stage. Thus over time, the neural networks should learn a large number of strategies, while generalising these strategies to other scenarios. Thus, as agents perform more tasks, the problem solving for known tasks should become "natural" to the *entire community*, as the neural networks start taking over from the negotiation modules.

Our proposal is to create a system that uses negotiation to enable the agents to find solutions to problems they encounter, and then learn this solution, in order to avoid the necessity of re-negotiating a solution, with the resultant communication and time costs, if the problem occurs again. Learning is considered to be an important feature of true autonomous agents [5]. One aspect of learning can be expressed as the ability to perform old tasks better as a result of learning and observing [4]. In this case it can be interpreted as the agents learning to perform old tasks more efficiently.

A Test Problem

The proposed system is applied to the game of Pursuit. A Pursuit board is represented by a 2 dimensional grid pattern, as shown in Figure 1. The asterisks represent the Captor agents, while the circle represents the Fugitive agent. The aim of the game is for the Captors to surround or corner the Fugitive. The fugitive and captors take alternate turns to move. Each agent may only move one block per turn. Legal moves on a Pursuit board are Up, Down, Left, Right and Stay. No diagonal moves are allowed. Furthermore, the board does not wrap around. A game is ended when the fugitive is captured, implying that the fugitive has no place to move to.

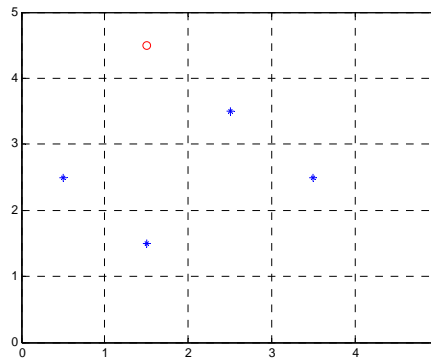


Figure 1 Pursuit Board representation

B System Description

The system has four separate agents representing the captors. Each of the agents has a neural network that is trained independently of other agents, as well as a negotiating algorithm. Each agent is responsible for making a valid move. This is defined as a move that is within the bounds of the board, and does not infringe on any block that another agent is in.

The fugitive agent has no real intelligence; it merely chooses a (legal) move at random.

The agents interact by proposing the move that they would prefer to make. This is sent to all other agents. If there is a conflict among the proposals, the agents send out a signal, and the agents negotiate a new proposal. This occurs until there are no conflicts among the agents. The agents then implement their moves. The first move proposed for each agent is always drawn from the neural network. If this is not successful, the agents start the negotiations.

C Captor Agent Description

As mentioned above, the Captor Agents consist of two parts: a pattern recognition unit, implemented as a neural network, and a negotiating algorithm that solves disputes with other agents.

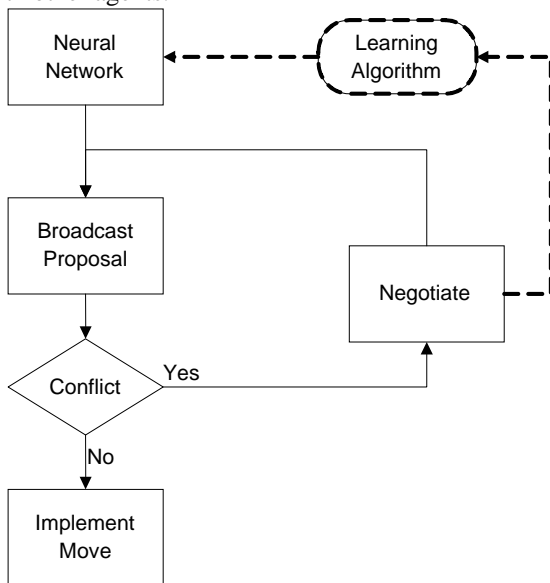


Figure 2 High Level Agent execution

As can be seen in Figure 2, the agents start off by proposing the solutions predicted by their neural networks. Only if this fails is the negotiation algorithm invoked. Whenever there is a conflict, all the agents are involved in the negotiation. The dotted feedback lines from the negotiation block to the neural network represents the network learning from the successful move made by the negotiation algorithm. In this way, the neural network learns cooperative behaviour.

The negotiation algorithm is shown in Figure 3. The agents each have a local goal based on the global goal of

the system. The global goal of the system is to win the game, i.e. capture the agent. In this system the global goal is expressed as the minimisation of the distance of all the Captor agents from the Fugitive agents. This is given by:

$$S = \sum_{N=1}^M \sqrt{(X_N - X_A)^2 + (Y_N - Y_A)^2} \quad (1)$$

Where: S is the sum of the distances to the fugitive; M is the number of Captor agents in the system; X_A, Y_A are the Fugitive agent's coordinates and X_N, Y_N are the current Captor's coordinates. If the captors follow this rule, they will win on a finite sized board, although not always in the least number of moves.

It would follow that if each captor agent minimised its own distance from the fugitive, the global goal would be satisfied. However, since each block on the board can hold only one agent exclusively, not all agents will be able to reach their own minimum on every board state. Therefore, to minimise the function in reality, some agents may not be in their optimal positions. In a Centralised system, where one agent controls the movements of all the Captors, this would be straightforward to implement, as the controller would have a global view of the problem, as well as only having to act in its own best interest. However, in a decentralised or Multi-Agent system, the implementation is not as clear. Since each agent will have its own local goal to pursue, it is not clear what the formulation of this local goal would be in order to reap the maximum benefit for the entire community [14].

In this case, the local goal of each captor agent is to minimise its own distance from the fugitive through negotiation with the other Captors. The negotiation algorithm of each agent is depicted in Figure 3. The agents are aware of the possible moves they can make at each turn. By evaluating the payoff that each of the possible moves will bring, the agents construct a Preference List of the possible moves and payoffs ordered from highest payoff (least distance) to lowest (furthest distance). Once the agent has made its list, it transmits its preferred move (which is the block on the board it would like to occupy), and the payoff value of the next preferred move on the list. In this way, the agent expresses its "need" for the block. The agent then creates a list of all the moves and payoffs received from the other agents. If there is a conflict between itself and one or more other agents on the choice of blocks, the agent attempts to resolve this. An agent will give way to the other agent(s) if it does not value the position as highly as any other conflicting agent (i.e. if it does not need the block as badly as other agents). An agent gives way by using the next move on its preference list. Once it gets to the end of its list, it wraps around to the start. If there is a tie for the highest value on the block, the agents involved in the tie go into a "lotto" mode, where they each randomly pick a number, with the highest number winning the right to keep the block. The losing agents then choose the next move on their Preference list. This procedure is iterated until no conflicts exist. At this point the agents implement their moves. In this manner, an approximation to the global optimum is performed. This sequence can be performed a number of times, which can

be seen to create many messages between agents, and thus a high usage of the communication medium.

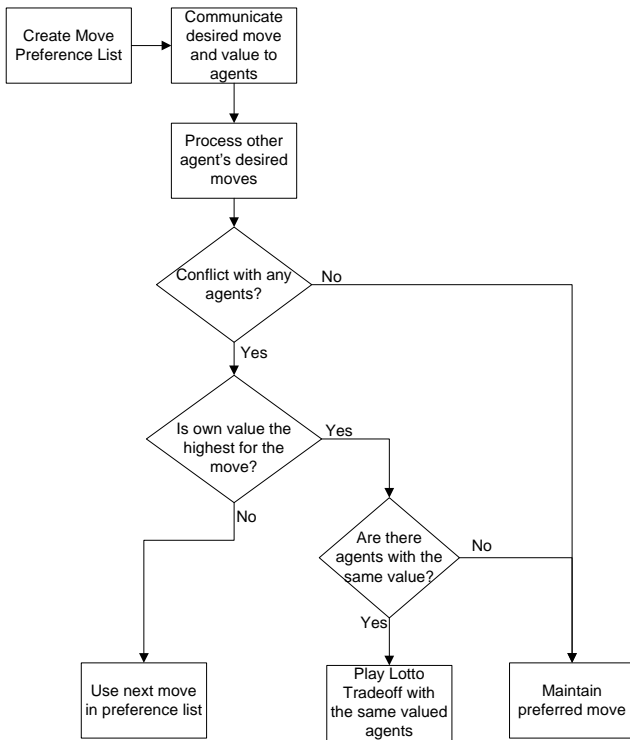


Figure 3 Agent Negotiating Algorithm

From this, it can be seen that the neural network of each agent learns solutions to board problems that are not only legal, but also cooperative with other agents on the board. Once these moves have been learned, the agents need only inform each other of the predicted move, and if there are no conflicts, proceed with the move. This then eliminates the need for the explicit negotiation transmissions.

Board Representation and Training

The input representation to a neural network is extremely important to the success of practical network applications [15][16][17]. Smooth representation of the input data, as well as using an input representation that describes features of the data can help reduce the number of states the network has to learn in a lookup-table type fashion, and increase the ability of the network to generalise learned data to new situations [17].

For this study, we implement the board representation as a set of relative differences in position. Each Captor agent has the relative positions of the other captors inputted in order of Cartesian distance. The relative position of the fugitive is then given as the last two inputs. This is depicted in Figure 4.

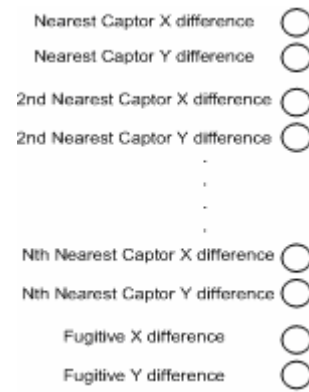


Figure 4 Network Input representation

This representation allows various formations of the agents to be captured, regardless of the exact location on the Pursuit board. Such formations are common when the captors are chasing the fugitive. s

Each Captor is trained with data specific to its own experience on the board. Training the agents from their past experience means that the direct communication between agents can be drastically reduced. This is due to the fact that each agent will take into account other agents moves when moving itself. Therefore, in terms of communication, the system acts as a swarm system, but with one major difference: each agent is “aware” of the global goal through the experience it has gained through negotiation, and so inherently acts cooperatively with other agents.

IV. IMPLEMENTATION

The system is implemented in MATLAB. NETLAB [19] is used to implement the neural networks for the Captor Agents. A two-layer MLP architecture is used for the networks. The networks use a Logistic function for the output units. A variable number of hidden units are used, depending on the amount of data the network has seen.

Since NETLAB does not directly support on-line training, an alternative scheme is used. Every time the agents negotiate a solution, the board position is recorded, as well as the moves agreed upon through negotiation. This data is added to the set of training data. After each game is complete, the agents are re-trained using the updated training data.

V. RESULTS

The simulation was run for 300 games. The results are summarised in Table 1. The measure of effectiveness used is the average number of game moves per game successfully predicted by the neural network. This is shown as the Average Learned moves/Game. The average number of moves per game when not using learning, i.e. using only the negotiation modules, is 6.5 over 300 games. This figure is used as the control number, to measure the quality of the predicted moves. If the average number of moves per game increases from this control, it implies that learning is actually having a detrimental effect on the agent’s performance.

The results of the learning simulation are broken into 3 stages of the game: 0 -100 games, in which it can be seen that the neural networks are still learning appropriate moves. The result is that the average number of moves per game in this phase actually increases from the control number, and it is for this reason that a percentage saving in bandwidth is not given, as it is meaningless in this case. A suggestion in this case would be to ignore the neural network's proposal until a critical number of training data has been obtained.

In the second phase, from game number 100 -200, it can be seen that the neural networks have learned sufficient appropriate moves to be effective in the system. In this phase, the average moves/game is at the level of the Control, but the amount of communication is decreased. This shows that the learning is useful in this stage.

In the third stage, the neural networks account for nearly half of the game moves. This represents a substantial reduction in negotiation communication.

TABLE 1
BREAKDOWN OF SIMULATION RESULTS AFTER
400 GAMES

	Game numbers		
	0 -100	100- 200	200-300
Average moves/ game	7.6	6.3	6.23
Average learned moves/game	2.4	1.92	2.7
% Bandwidth Reduction	N/A	30%	44%

VI. ANALYSIS

The knowledge that the agents have obtained as a result of their learning can be thought of as obtaining a model of their environment, or more specifically, as a model of the behaviour of the other agents within the system. This is due to the fact that when the agents correctly predict moves using their neural networks, they are not only maximising their own goal, but taking into account the needs of the other agents in the system. This is despite the fact that they have no explicit knowledge of the behaviour or workings of the other agents: the knowledge that they have obtained is purely through observation and interaction. In the negotiation, or deliberative algorithm, the agents take no account of the moves that the other agents might make: they wait for another agent to actively inform of a problem. To programme into such algorithms

the ability to anticipate other agent's moves would effectively mean that the agents would each have a global view of the problem, as well as having intimate knowledge of every agent's abilities and desires in the system. This would defeat the object of a Multi-agent system, where each agent is expected to operate in a highly distributed environment, and having a limited viewpoint of the problem, as well as considerable duplication of abilities. Furthermore, it may not even be appropriate for each agent to have this knowledge, especially when the agents are divided into sub-teams [3].

After a large number of games, each agent will have observed a large variety of board positions, and for this system, the neural networks will tend to be the same. This is only because the agents in this system all have the same "abilities" or allowed moves. In a scenario where there are agents with different abilities, as for example the separate pieces in a chess game, the networks will each train on totally different output moves, and so will tend to be different even after observing an infinite number of moves. However, there is no reason why the agents should not still be able to anticipate each others moves, since they will be trained with this data.

The reduction in communication as shown here can have large benefits in certain situations, such as when the communication channel carries a significant monetary cost, or if the channel is not reliable or slow. All these factors contribute to the overall effectiveness of the agent community, and should be minimised.

VII. FUTURE RESEARCH

There is already considerable research in the field of agent learning. We believe that learning is not only useful for added functionality in agents, but also to enhance existing agent operation. The main problem in any learning system is often the presentation of the information to the system [15]. With other means of representation, it may be possible to increase the benefits of learning, and create more efficient agents. In complicated environments, which are common of multi-agent domains, it would be desirable, and probably necessary, to have the agents automatically determine the relevant representation of information, as well as choosing the information that would help in the decision.

Furthermore, the area of agent coordination is a currently an active area of research. . A methodology whereby agents are equipped with a host of coordination strategies at design time, and then learn through interaction the appropriate local strategy to use in a given situation has been proposed by [18]. A mechanism to accommodate this type of agent behaviour will need to be investigated.

VIII. CONCLUSION

The possibility of reduced communication between agents, while still cooperating with each other is demonstrated in this paper. We show that learning can be successfully applied to this aim. This results in agents that

effectively anticipate other agent's behaviours, despite having no explicit model of the agents programmed.

A reduction in communication of up to 44% is achieved in this study. This level of reduction can have extremely positive effects if communication in the system is in some way a significant cost factor in the overall effectiveness of the system.

IX. REFERENCES

- [1] Maria Chli, Philippe De Wilde, Jan Goossenaerts, Vladimir Abramov, Nick Szirbik, Pedro Mariano, Rita Riberio, "Stability of Multi Agent Systems", *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics* 2003, Volume 1, 2003, Pages: 551-556
- [2] E. Bonabeau, M. Dorigo, G Theraulaz, *Swarm Intelligence – From Natural to Artificial Systems*, Oxford University Press, 1999.
- [3] D. Kalofonos, T.J. Norman, "An Investigation into Team-Based Planning," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics* 2004, pp 5590- 5595.
- [4] Z. Ren, C.J. Anumba, "Multi-agent systems in construction – state of the art and prospects," *Automation in Construction*, no. 13, 2004, pp. 421-423.
- [5] K.P. Sycara. "Multiagent Systems". *AI Magazine, American Association for Artificial Intelligence*, Summer 1998. pp 79-92.
- [6] J.J. Castro-Schez, N.R Jennings, X. Luo, NR Shadbolt. "Acquiring Domain Knowledge for negotiating agents: A case study", *International Journal of Human-Computer Studies* 61 (1), 2004, pp 3-31.
- [7] N.R. Jennings, M. Wooldridge, D. Kinny, "The Gaia Methodology for Agent-Oriented Analysis and Design", *Autonomous Agents and Multi-Agent Systems*, 3, 2000. pg 285-12
- [8] N.R Jennings, M. Woodridge,"Intelligent Agents: Theory and practice", *The Knowledge Engineering Review*, 10 (2), pg 115-152, 1995.
- [9] Hadelia,, P. Valckenaersa, M. Kollingbaumb, H. Van Brussel, "Multi-agent coordination and control using stigmergy," *Computers in Industry* 53, 2004, pp 75–96
- [10] Nils. J Nilsson, *Artificial Intelligence: A new synthesis*, Morgan Kaufmann Publishers, San Francisco, California, 1998.
- [11] R. Kurzweil, *The age of intelligent Machines*, Massachusetts Institute of Technology, 1990. pp 231- 234
- [12] Alison Cawsey, *The Essence of Artificial Intelligence*, Prentice Hall Europe, 1998.
- [13] M. Jordan, C. Bishop, *Neural Networks*, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Last accessed 12 August 2003, <ftp.publications.ai.mit.edu>.
- [14] J.A. Marshall, Z. Lin, M.E. Brouke, B.A Francis, "Pursuit Strategies for Autonomous Agents", *Proceedings of the 2003 Block Island Workshop on Cooperative Control*. Springer-Verlag Series: Lecture Notes in Control and Information Sciences, 2004.
- [15] Sebastian Thrun, "Learning to Play the Game of Chess", *Advances in Neural Processing Systems* 7, 1995.
- [16] G. Tesauro, "Programming backgammon using self-teaching neural net", *Artificial Intelligence* 134(2002) pp 181-199.
- [17] G.Tesauro, "Practical Issues in Temporal Difference Learning", *Machine Learning* 8 (1992), pp. 257-277.
- [18] C. B. Excelente-Toledo and N. R. Jennings "The dynamic selection of Coordination Mechanisms" *Autonomous Agents and Multi Agent Systems* 9, 2004, pp 55-85
- [19] I. Nabney, *Netlab – Algorithms for Pattern Recognition*, John Wiley and Sons, New York, 2001