USING DYNAMIC NEURAL NETWORK TO MODEL TEAM PERFORMANCE FOR COORDINATION ALGORITHM CONFIGURATION AND RECONFIGURATION OF LARGE MULTI-AGENT TEAMS

JUMPOL POLVICHAI
MICHAEL LEWIS
School of Information Sciences
University of Pittsburgh
Pittsburgh, Pennsylvania, USA

PAUL SCERRI
KATIA SYCARA
School of Computer Sciences
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

ABSTRACT
Coordination of large numbers of agents for performing complex tasks in complex domains is a rapidly progressing area of research. Because of the high complexity of the problem, approximate and heuristic algorithms are typically used for key coordination tasks. Such algorithms usually require tuning of algorithm parameters to get the best performance in particular circumstances. Manually tuning of parameters is sometime difficult. In this paper, we introduce a new concept of dynamic features for a neural network, called dynamic networks, to model the way a coordination algorithm will work under particular circumstances. Genetic algorithms are used to train the networks from an abstract simulation, TeamSim. At the end, the model is used to rapidly determine an appropriate configuration of the algorithm for a particular domain. Users specify required tradeoffs in algorithm performance and use the neural network to find the best configuration for those tradeoffs. Algorithm reconfiguration can even be performed online to improve the performance of an executing team as situation changes. We present preliminary results showing the approach promisingly facilitating users to configure and control a large team executing sophisticated teamwork algorithms.

INTRODUCTION AND PREVIOUS WORK

Sophisticated, complex coordination allows large groups of agents to perform complex tasks in domains such as space (Kortenkamp et al., 2000) and the military (Glade, 2000). Cooperation between heterogeneous robot teams is very complex when the tasks cannot be completely divided up and assigned a priori and robots must work together, dynamically interacting to achieve common goals, for example see work of Parker et al (2001). In a large team (Scerri et al., 2004), the teamwork algorithms include with several coordination algorithms, such as algorithms for task allocation, communication, and planning. The relationships between these algorithms to the behaviors of team are highly non-linear, stochastic, and extremely complex. Knowing very well only in a part of the system is not likely to understand how the team can be controlled and get the best performance out of them. Therefore, it is critical for human operators to have better understanding of overall teamwork algorithms. However, this broadly understanding can not easily obtain by running a few of the real missions. In multi-robot systems, it is usually impossible to have lots of physical experiments, therefore simulation is used to gain understanding about the developing system ((Dixon et al., 1999), (Fu et al., 2003)). On the other hand, using simulation is difficult when the system is highly non-linear, stochastic, and extremely complex, since the simulation can only provide information of one setting at a time. Fortunately, because the simulation is fast it can be used to create large amounts of data containing relationships between parameters and team performances. Therefore, for representing these complex relationships, a meta-model of teamwork algorithms is created by modeled from the teamwork simulation. As a result, the team meta-model will play a big role in facilitating the process of gaining better understand of the system.

In cooperative control, selecting the right coordination strategy can have a significant impact on the team performance. In many multi-robot applications (e.g., (Gerkey et al., 2004), (Parker, 2000), (Parker, 2002), (Parker et al., 2001) etc.), collaborative robots must coordinate their plans or actions. Selecting the right action, plan or resource to resolve some problems can have a significant impact on the performance particularly in a larger multi-robot system. Unfortunately, selecting the right strategy is difficult. There is often a wide diversity of strategies available, and they can lead to significant variations in the team performance in multi-robot systems. Due to the high computational complexity of coordination, critical coordination algorithms typically use heuristics which are parameterized and need to be tuned for specific domains for best performance. When several coordination algorithms are used together, the performance of one algorithm will likely affect the performance of the other algorithms, thus tuning parameters of the individual algorithms must be performed together. Moreover, the relationship between coordination configurations, environmental conditions, and team performance is highly non-linear and extremely complex. Hence, getting best performance from a set of coordination algorithms often involves a complex parametric tuning process that must be performed on a per problem basis.

Process of tuning control parameters for better team performance is not only done in coordination algorithm level, but also done in robot levels as well. For example, Parker (2000) demonstrated an approach to allow a heterogeneous group of robots to adapt their actions while environmental conditions are changing over time. In order to get the good performance, tuning of these control parameters is required. Parker used initial learning for parameter settings before going on to the performing period. In Parker (2002), she demonstrated a distributed approach using the potential field technique for cooperative team of robots to track multiple moving targets by considering actions to keep moving objects under surveillance. The ALLIANCE architecture is used to provide mechanisms for fault tolerant cooperative control, and allows robot team members to adjust their actions based upon the actions of their teammates. The results show successful in achieving the target task. However, in order to get a good solution, the system required considerable tweaking of parameters.

Previous approaches to configuring coordination for a team in a particular domain typically either required hand-tuning (Falcone and Castelfranchi, 2001) or learning (Bui et al., 1997) in the domain. Hand-tuning of parameters is a time consuming process that typically requires extensive experience with the algorithms for good performance. Learning requires that the team perform many trials in the specific circumstances it is to be used. If the environment can vary dramatically, e.g., the specific characteristics of a disaster response scenario can vary greatly from disaster to disaster but the same team should respond to each one, learning may be infeasible (Riley and Veloso, 2003). Thus, previous work does not provide a good solution to the problem of rapid team configuration.

We have developed an approach to configuring coordination algorithms that incorporates three key ideas. The first idea is to create a team performance model that captures the relation between the environment and team configuration parameters and measures of performance in a way that allows rapid exploration by users. We have developed an abstract simulation of the coordination algorithms and a highly configurable environment to test these ideas. Because the simulation is fast it can be used to create large amounts of data containing relationships between parameters. Due to the non-determinism of environments and coordination algorithms and the sensitivity of performance to circumstances these relationships are highly non-linear. They are also highly variable even for the same configuration. To create a concise model of the data we use genetic algorithms to learn a two hidden layers neural network with dynamic features (Polvichai and Khosla, 2002). A neural network with two hidden layers is sufficiently powerful (Perlovsky, 2001) to capture the complexity of these data and thus provides a rapid mapping from environment and configuration parameters to performance parameters. The aim of this work is to remove some of the art from configuring a team for a particular environment. The team performance model captures complex relationship between the environment, configuration and team performance.

The second idea in this work is to use the team performance model to find the best configuration of the team to meet specific performance constraints, e.g., the tradeoff between communication bandwidth and good allocation of resources. Using the team performance model in "reverse" allows users to specify performance tradeoffs and rapidly receive a configuration that best meets those constraints. Since not all parameters are configurable, e.g., the observability of the domain cannot be changed during execution, we cannot simply use backpropagation method of the neural network to find input parameters that meet our output requirements. Instead we perform a search over the changeable configuration parameters to find a configuration that best meets the required performance tradeoffs.

The third idea is to allow the team to be reconfigured online, using the team performance model to determine appropriate parameters for the prevailing conditions. When users have changing preferences or know of changing constraints, they can simply specify these requirements and allow the team performance model to find algorithm parameters that will best meet those requirements. This approach provides a powerful and effective way for manipulation of team performance during execution time and provides an additional mechanism for the supervisory control of executing teams.

We have implemented and tested this approach with an abstract teamwork simulation, TeamSim. The abstract teamwork simulation was also used to provide a large amount of data from which the dynamic neural network model was created. The model was shown to accurately capture the behavior of the coordination algorithms across a wide range of configurations. The model was then used to configure teams and change configurations online with results as predicted by the model.

ALGORITHM AND APPROACH

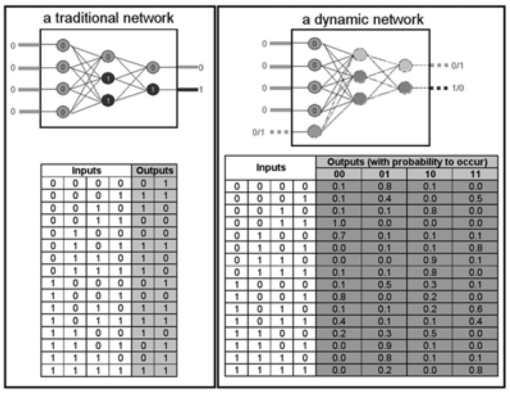
We have developed an approach to facilitate users in understanding and manipulating the relationship between configurations of coordination algorithms and measures of their performance. The approach has several steps. It starts with the collection of large data sets generated by our abstract teamwork simulation. Then, an evolutionary computation approach is used to learn team performance models. An input/output model of artificial neural networks with dynamic features is used to represent the team performance model. Finally, mission experiments with the teamwork simulation are used to investigate the performance of the team performance model.

Data sets

TeamSim is an abstract teamwork simulation developed to investigate coordination algorithms in cooperative multi-agent teams within simulated environments. Running the simulation provided a huge training data set, used to learn models of team performance. To create a precise team performance model, very large amounts of data were needed to capture the complex relationships between parameters. The complete configurable space is very large. Providing full coverage of all possible configurations was impossible. To overcome this difficulty we collected simulation data using two methods: non-random sampling and random sampling. The non-random sampling method gathered data by specifying cases chosen to roughly to cover the possible ranges of input parameters. Although this data set roughly represents the entire configuration space it has sparse coverage. The random sampling method was designed to collect data from throughout the space but without guarantees of coverage. Each data entry consists of all parameters and output performances from a run.

Dynamic Neural Network Models

To represent the highly non-linear relationship between environment, configuration and team performance, we used multilayer feed-forward neural networks. The network topology consists of sixteen nodes in the input layer (one input node representing each configurable or system parameter), sixteen nodes in the first hidden layer, eight nodes in the second hidden layer, and seven nodes in the output layer. The two hidden layers used sigmoid units, and the output layer use linear units. This three-layer feed-forward network is capable of representing any arbitrary function (Perlovsky, 2001).



Inputs				Outputs	
0	0	0	0	0	1
0	0	0	1	1	1
0	0	1	0	1	0
0	0	1	1	0	0
0	1	0	0	0	0
0	1	0	1	1	1
0	1	1	0	1	0
0	1	1	1	1	0
1	0	0	0	0	1
1	0	0	1	0	0
1	0	1	0	1	1
1	0	1	1	1	1
1	1	0	0	1	0
1	1	0	1	0	1
1	1	1	0	0	1
1	1	1	1	1	1

Inputs				Outputs (with probability to occur)			
				00	01	10	11
0	0	0	0	0.1	0.8	0.1	0.0
0	0	0	1	0.1	0.4	0.0	0.5
0	0	1	0	0.1	0.1	0.8	0.0
0	0	1	1	1.0	0.0	0.0	0.0
0	1	0	0	0.7	0.1	0.1	0.1
0	1	0	1	0.0	0.1	0.1	0.8
0	1	1	0	0.0	0.0	0.9	0.1
0	1	1	1	0.1	0.1	0.8	0.0
1	0	0	0	0.1	0.5	0.3	0.1
1	0	0	1	0.8	0.0	0.2	0.0
1	0	1	0	0.1	0.1	0.2	0.6
1	0	1	1	0.4	0.1	0.1	0.4
1	1	0	0	0.2	0.3	0.5	0.0
1	1	0	1	0.0	0.9	0.1	0.0
1	1	1	0	0.0	0.8	0.1	0.1
1	1	1	1	0.0	0.2	0.0	0.8

Figure 1. Two abstract examples of networks are shown. On the left, a traditional network maps input-output pairs as shown in the table below. On the right, a dynamic network maps all input patterns with all possible output patterns with different probabilities to occur.

Eggenberger et al. (1999) introduced the idea of dynamic rearrangement of biological nervous systems to accommodate learning in non-stationary environments. Their approach allows neural networks an additional mechanism to dynamically change synaptic weight modulations and neuronal states during execution. This capability of changing the modulation types allows the control networks to change their structures when the environment is changed. With inspirations from the dynamic rearrangement idea, we use the concept, called Dynamic Networks (Polvichai and Khosla, 2002, 2003), which allows all internal nodes in the network to act stochastically and independently even if all external input data remain unchanged. Figure 1 shows an abstract example of a dynamic network compared with a regular network. Dynamic networks capture randomness from the additional input nodes fed with internal random signals. These random signals along with weights between the additional nodes and the hidden nodes bring dynamic states into internal nodes within the network and output nodes. Changing weights result in changing behaviors of the dynamic networks. This kind of network enlarges the capability to deal with non-deterministic problems. Moreover, this stochastic-ness adds robustness and flexibility to the network. If a target system has high variation in output even for the same input configuration, the dynamic network adapts the weights to match this variance. If the target system were deterministic, these weights would adapt to zero. Because team coordination algorithms are dynamic and non-deterministic a dynamic network provides a good model. Dynamic networks were applied into our multilayer neural network by adding four special nodes into the input layer, so that the total number of input nodes becomes twenty. These special nodes insert random values between 0.0 and 1.0 into the network.

Genetic Algorithms

Genetic Algorithms is a search technique loosely based on the mechanism of natural selection and genetics. In relation to problem domains, structures of a possible solution are represented in string formats. Given an environment and a goal formulated as a fitness function, an initial population and selecting genetic operators are generated at random. The new generation of possible solutions is generated using three common genetic operators. A number of processes of generating new populations based on prior populations are repeatedly executed until the termination condition is met. The solution of the problem is found in the final population.

For this work, each generation of the population was set to have a thousand individuals. Each individual represented a neural network. The chromosomes of each individual were the weights of the neural network. All weights are randomly generated at the start. After evaluation with the training data set, the first 500 best performances are kept and then used to produce the new 500 individuals randomly by genetic operations. The fitness function was the average of square error between target output and actual output as follow:

$$\sum_{d\in D}\sum_{p\in P}(O^{d}_{p,t}-O^{d}_{p,a})^2 \Big/ sizeof(D)$$

Where D is the set of training data ($d\in D$), P is the set of system performance measures ($p\in P$), $O^{d}_{p,t}$ is the target output of the pth performance parameter of the data entry d, and $O^{d}_{p,a}$ is the actual output of the pth performance parameter of the data entry d. This squared sum is taken across all performance parameters and these measures were normalized so it makes sense to sum over them. The lower this score is, the better.

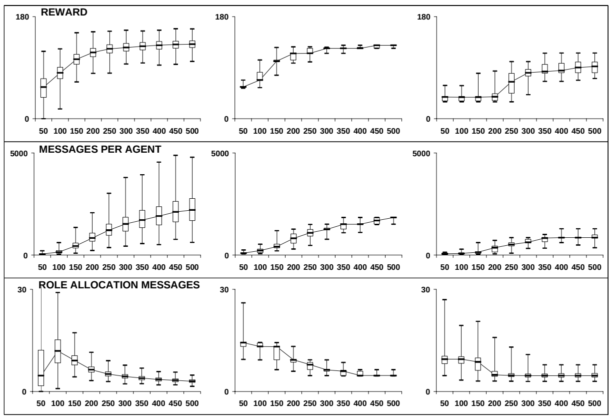


Figure 2. These graphs are plotted between three performance measures and number of team members. In each row, three graphs are plotted. The left most graph is plotted from the verification data set. The middle and right most graphs are plotted from the team performance model learned from verification data set and random data set respectively.

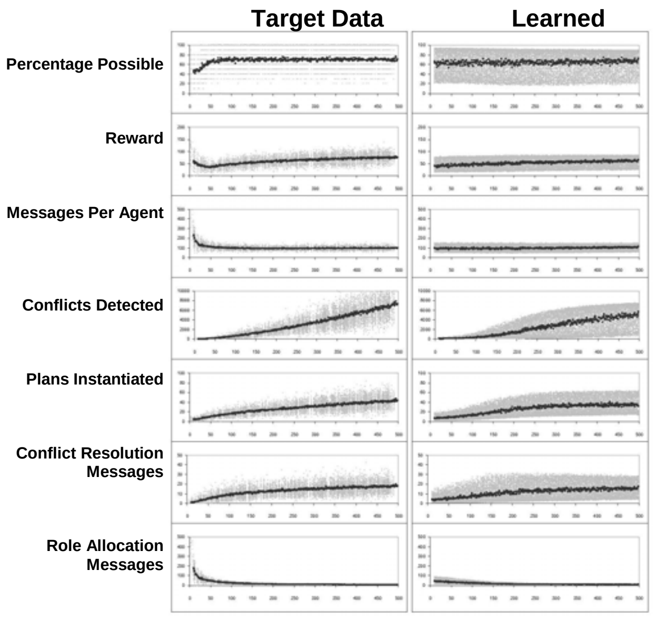


Figure 3. Example plots show comparisons between target and learned results of seven output performance measures from a particular setting. The gray dots are the actual outcomes (showing variances) and the black dots are the averaged ones.

The training data was sampled from the data set, so that different training data was used in each generation. The size of each training data set was 5000. The termination conditions were either reaching the maximum iteration (10000) or reaching the minimum error of the network output (0.05). When the termination condition was met, the best performing individual was chosen to be our team performance model. We learned our neural network using genetic algorithms because the relationship between variables was not only non-linear, but also highly dynamic and non-deterministic, which is demanding for backpropagation method which must overcome a huge number of local minima. Secondly, in genetic algorithms, the unit of adaptation is not an individual agent, but a population of agents, which is excellent for dealing with very huge and noisy training data.

EXPERIMENTS AND RESULTS

In this section we present evidence that our approach achieves our primary goals which are: 1) capturing the effects of configuration parameters on performance measures in the team performance model 2) demonstrating the correspondence between predicted and obtained changes in performance using the team performance model.

Team Performance Model Verification

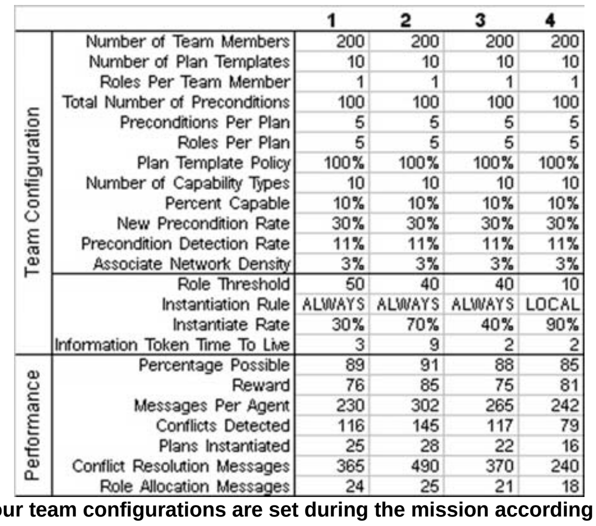
The objective of the team performance model is to capture the relation between a vast space of possible system and configuration parameters and the performance measures. To obtain high resolution for comparisons, we created a small reference data set consisting of data obtained from running our abstract simulation by varying only 3 input parameters: *Number of Team Members*, *Associate Network Density*, *Plan Template Policy*, leaving other input parameters fixed. A second data set of much lower resolution was created by generating the same number of observations using random configurations over all the input parameters. These two data sets were separately used to train new team performance models which in turn generated graphs which were compared with those plotted from the original reference data set. As shown in figure 2, there is close correspondence between the team performance models and data set in the relation between reward and number of agents, and between messages per agent and number of agents for each level of network density. As expected the correspondence is closest for the high resolution model yet even the low resolution model captures the increases in messaging with team size and the flattening of the curves as network density decreases.

On another test, a larger data set is collected with another fixed scenario. When using this bigger data set, with enough learning time, the team performance model can capture the uncertainty of the target system. As shown in figure 3, the learned results are approximately the same as the target data especially the averages (the black dots). However, there are small problems in the area where the number of team members are lower than 50. This problem can be solved by biasing the fitness function to give more score in that area.

Reconfiguration Verification

In this subsection, we examine the performance of the team performance model and its use through the team control interface. We use TeamSim, our abstract simulation, as the target system. The interface is connected directly with TeamSim, so that users can set team configurations and monitor team performance measures online. The user configures the team at the start of the mission. Performance measures from the simulation are graphically displayed on the user interface at every time step. When performance changes are requested the offline features are used to find suitable reconfigurations.

The team control interface and reconfiguration assistance were evaluated over 10 scenarios. Scenarios were selected to provide situations that would require users to reconfigure their team in order to meet performance targets. For example, in a mission involving a very large team of 300 agents the user might be requested at some point in the mission to reduce the number of messages per agent or increase the number of plans instantiated. Performance measures are recorded throughout the execution. Each scenario was run for 250 time steps, with each step taking 5 seconds. The data presented here represents 4 hours of runtime with a user in the loop. One scenario with a team of 200 agents is shown in figure 4. At event point 1, the initial team configuration is set. At event point 2, the user is asked to increase level of rewards obtained by the team disregarding other performance measures. Using the output-to-input feature of the team performance model the user finds a new coordination configuration that increases reward performance and reconfigures the team. At event point 3, the network communication bandwidth is reduced limiting the time-to-live for information tokens to 2 hops requiring another team reconfiguration to lessen the degradation in performance. At event point 4, again, the user is asked to reconfigure to improve reward performance. Results for six of the performance measures are shown in figure 5. The bold lines show average values of variation in actual performance measures over two event points for the configured system while the dotted lines indicate the values predicted by the output-to-input model. Despite the high variability of team performance measures the model accurately predicts the effects of reconfiguration on average performance values across all six measures. Although the predicted values are not precisely close to the actual values, due to the effect of normalization in learning process and uncertainty of the target system, the predicted values are very precise correspondence to what happen to the actual performance.



		1	2	3	4
Team Configuration	Number of Team Members	200	200	200	200
	Number of Plan Templates	10	10	10	10
	Roles Per Team Member	1	1	1	1
	Total Number of Preconditions	100	100	100	100
	Preconditions Per Plan	5	5	5	5
	Roles Per Plan	5	5	5	5
	Plan Template Policy	100%	100%	100%	100%
	Number of Capability Types	10	10	10	10
	Percent Capable	10%	10%	10%	10%
	New Precondition Rate	30%	30%	30%	30%
	Precondition Detection Rate	11%	11%	11%	11%
	Associate Network Density	3%	3%	3%	3%
	Role Threshold	50	40	40	10
	Instantiation Rule	ALWAYS	ALWAYS	ALWAYS	LOCAL
	Instantiate Rate	30%	70%	40%	90%
	Information Token Time To Live	3	9	2	2
Performance	Percentage Possible	89	91	88	85
	Reward	76	85	75	81
	Messages Per Agent	230	302	265	242
	Conflicts Detected	116	145	117	79
	Plans Instantiated	25	28	22	16
	Conflict Resolution Messages	365	490	370	240
	Role Allocation Messages	24	25	21	18

Figure 4. Four team configurations are set during the mission according to occurred situations, and along with performance measures predicted.

By demonstrating the team performance model's effectiveness for predicting the effects of team configurations, these tests demonstrate the potential of our approach for both the initial configuration of teams and supervisory control of executing teams. This basically shows that control interventions produce approximately the predicted changes across the vector of controlled variables, which we couldn't do with the C4.5 model.

CONCLUSIONS

In this paper, we have presented a preliminary approach to building a tool that makes it possible for users to quickly explore relationships between configuration parameters of coordination algorithms and performance measures for large teams. Initial experiments showed that the approach was able to aid users configuring and controlling a large team executing complicated coordination teamwork algorithms. While this preliminary work suggests that the parameters of team work algorithms may be effectively controlled to manage team performance in much the same way that individual control parameters have been used to manage robotic behaviors much research remains to be done. While we have qualitatively demonstrated that our methods produced a model that behaves as we believe it should, quantitative measures and comparisons with alternate approaches are needed to support this claim with greater certainty.

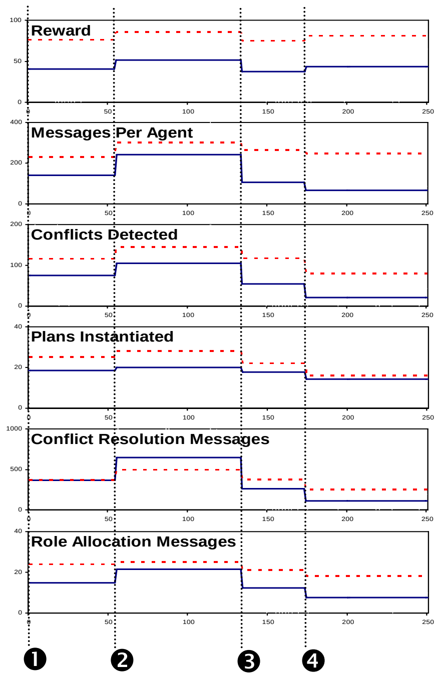


Figure 5. Six performance measures recorded from TeamSim are plotted during the mission. Thick lines show the average values of actual performance measures of each configuration setting. Dotted lines are the predicted values by the user interface. Four events of configuration and reconfigurations are shown.

ACKNOWLEDGMENTS

This research has been sponsored in part by AFRL/MNK Grant F08630-03-1-0005 and AFOSR Grant F49620-01-1-0542.

REFERENCES

Bui, H. H., Venkatesh, S., and Kieronska, D., 1997, "A framework for coordination and learning among team members," *In Proceedings of the Third Australian Workshop on Distributed AI (DAI-97)*, pages 116-126, Perth, Australia.

Dixon, K., Dolan, J., Huang, W., Paredis, C., and Khosla, P., 1999, "RAVE: A real and virtual environment for multiple mobile robot systems," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems. (IROS'99)*, Kyongju, Korea, October 17-21.

Eggenberger, P., Ishiguro, A., Tokura, S., Kondo, T., and Uchikawa, Y., 1999, "Toward seamless transfer from simulated to real worlds: A dynamically-rearranging neural network approach," *in Proceeding of the Eighth European Workshop in Learning Robot (EWLR-8)*, pp. 44-60.

Falcone, R., and Castelfranchi, C., 2001, "Tuning the collaboration level with autonomous agents: A principled theory," *In Proceedings of the 7th Congress of the Italian Association For Artificial intelligence on Advances in Artificial intelligence,* September 25 – 28, F. Esposito, Ed. Lecture Notes In Computer Science, vol. 2175. Springer-Verlag, 212-224.

Fu, D., Houlette, R., Jensen, R., and Bascara, O., 2003, "A Visual, Object-Oriented Approach to Simulation Behavior Authoring," *The Interservice/Industry Training, Simulation & Education Conference*.

Gerkey, B., and Mataric, M. J., 2004, "A Formal Analysis and Taxonomy of Task Allocation in Multi-robot systems," *International Journal of Robotic Research*, vol. 23, no. 9, pp. 939-954.

Glade, D., 2000, "Unmanned aerial vehicles: Implications for military operations," *Center for Strategy and Technology Air War College, Tech. Rep. Occasional, Paper No. 16*.

Kortenkamp, D., Keirn-Schreckenghost, D., and Bonasso, R. P., 2000, "Adjustable control autonomy for manned space flight," *in IEEE Aerospace Conference*.

Lu, M., AbouRizk, S. M., and Hermann, U. H., 2001, "Sensitivity Analysis of Neural Networks In Spool Fabrication Productivity Studies," *Journal of Computing in Civil Engineering*, Vol. 15, No. 4, pp. 299-308, October.

Parker, L. E., 2000, "Lifelong Adaptation in Heterogeneous Multi-Robot Teams: Response to Continual Variation in Individual Robot Performance," *Autonomous Robots*, V 8, pp. 239-267.

Parker, L. E., 2002, "Distributed algorithms for multi-robot observation of multiple moving targets". *Autonomous Robots*, 12(3).

Parker, L. E., Guo, Y., and Jung, D., 2001, "Cooperative robot teams applied to the site preparation task," *In Proceedings of 10th International Conference on Advanced Robotics*, pages 71–77.

Perlovsky, L. I., 2001, *Neural Networks and Intellect: Using Model-Based Concepts*, Oxford University Press.

Polvichai, J., and Khosla, P., 2002, "An evolutionary behavior programming system with dynamic networks for mobile robots in dynamic environments," *in Proceedings of 2002 IEEE/RSJ International Conference on Intelligent Robots and System*, vol. 1, pp. 978-983.

Polvichai, J., and Khosla, P., 2003, "Applying dynamic networks and staged evolution for soccer robots," *in Proceedings of 2003 IEEE/RSJ International Conference on Intelligent Robots and System*, vol. 3, pp. 3016-3021.

Quinlan, J., 1993, *C4.5: Programs for machine learning*, Morgan Kaufmann.

Riley, P., and Veloso, M., 2003, "An overview of coaching with limitations," *in Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 1110-1111.

Scerri, P., Xu, Y., Liao, E., Lai, J., and Sycara, K., 2004, "Scaling Teamwork to Very Large Teams," *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '04)*, pages 888--895. IEEE Computer Society.