Determining Improved Values for Coordination and Communication Parameters in Multiagent Systems: A Case Study

Michael L. Dowell
Department of Math and Computer Science
Georgia Southern University
Statesboro, GA 30460
dowell@gsu.cs.gasou.edu

and

Larry M. Stephens
Department of Electrical & Computer Engineering
University of South Carolina
Columbia, South Carolina 29208
stephens@sc.edu

ABSTRACT

This paper introduces new coordination parameters and experimental results for the multiple-team pursuit problem. Our results confirm that reducing the costs of coordination increases the effectiveness of a multiagent system; however, even in simple problem domains, determining the best coordination strategies may require extensive experimentation. A simple and flexible performance metric is defined to assess the effect of changes in coordination strategy. This paper presents results from a series of experimental runs, each of which help to determine improved values for parameters that influence coordination strategies. Several of the newly defined parameters are useful for other multiagent systems. The process presented in this paper can be the basis for a machine-learning algorithm for improving the coordination of groups of agents for distributed problem solving.

1 Introduction

Improving the performance of multiagent systems is a complex and time-consuming process. As these systems become larger and more sophisticated, performance becomes dependent upon an increasing number of parameters. Designers and programmers typically use intuition or guesswork in determining these parameters, some of which govern when and how agents interact. This paper examines optimizing the performance of a multiagent system by experimenting with *coordination*[*] parameters. A simple coordination strategy is used among a group of agents solving a problem: one team of agents determines when to interact with another team to exchange roles in a pursuit game. The values of coordination parameters determine if and when agents should attempt to interact given that a goal is to reduce the time required to solve the problem. However, the act of attempting to interact itself has a cost, and the system must balance the frequency of attempts with the benefits of successfully cooperating.

The experimental results we obtained suggest how to automate the parameter-adjustment process to find improved values for coordination parameters based on given performance metrics. In this sense, the experiments—although carried out by a person rather than a machine—are a form of *organization* learning for multiagent systems [3], [4]. Organization learning is different from other forms of multiagent learning (such as those described in [8], [11], [15], [16]) in that the learning is about how one *group* of

[*] In the introduction to [2], Bond and Gasser view cooperation as a specialization of the more general concept of *coordination*: a property of interaction among some set of the agents performing some collective activity. In [2] the term "cooperation" is simply a special case of coordination among nonantagonistic agents; we choose to use the more general concept for this paper.

agents interacts effectively with another group; the learning in [8], [11], [15] involves individual learning by agents in a single group.

2 Multiple Team Pursuit

2.1 Pursuit Problem Background

The pursuit game is a well-known test problem for the field of distributed artificial intelligence [1], [5-7], [9-10], [12-14]. While different researchers have used different versions of the game, the overall structure of the problem has remained the same. The game takes place on a finite grid of squares that can be empty or occupied by an agent. There are two types of agents within the system: prey agents and predator agents. During each turn, an agent can remain in its current location or move up, down, left, or right. A collision occurs when two or more agents attempt to move to the same location. Collisions are resolved by randomly selecting one of the agents to occupy the disputed location and blocking the other agents' moves.

The goal of the pursuit problem is to surround a prey agent with four predator agents. A group of predators must occupy the gird locations above, below, to the left and to the right of the prey agent, as shown in Fig. 1. Typically, the game starts with four predator agents and one prey agent in a random initial configuration and continues until the predator agents capture the prey agent.

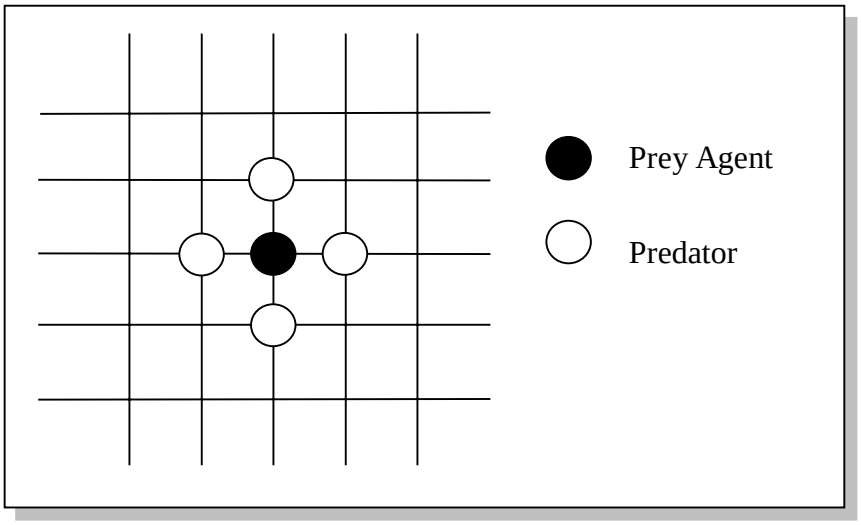

Fig. 1: Capture Configuration

Recent work [16] has introduced two other types of goals for the predator agents. The first involves “killing” the prey agent by having a predator occupy the same grid location. This goal is much easier to achieve than surrounding the prey since it can be accomplished by only one agent. The second type of goal, called “surrounding” in [16], traps the prey in a corner of the grid and requires only two agents. Cornering an agent is more difficult than killing an agent but is not as difficult as surrounding an agent. Our research permits neither killing nor cornering; in all of our experiments, the four predator agents must completely surround the prey. In the case of cornering, our predator agents back away from the prey and wait for it to move out of the corner before trying to surround it. This choice makes capturing prey more difficult than cornering in [16] because the leader of the predators has to determine that the prey is in the corner and instruct all the team members to retreat.

2.2 Multiple Team Pursuit Environment

The *m*ultiple *t*eam pursuit *p*roblem (MTP) is conceptually motivated by [5], which describes a testbed called MICE—the Michigan Intelligent Coordination Experiment—and presents variations of the standard scenario. The most common scenario described in [5] involves two prey agents and six predator agents. The prey agents' goal is to move across the grid to reach a "safe area" before the predators capture them.

The multiple team pursuit problem defined in our paper differs from [5] by using three *teams* of predators attempting to capture three prey agents. A team of predators has four participating agents and seeks to capture an assigned prey agent, which is assumed to move at random for simplicity; a MTP configuration is shown in Fig. 2. Unlike some other research involving the pursuit game, our research is not concerned with how the *individuals* in a single team perform or interact with other team members. The main focus of the MTP problem is the *coordination between the teams*; in particular, two teams might agree to exchange assigned prey agents because each team is better located to capture the other's prey.

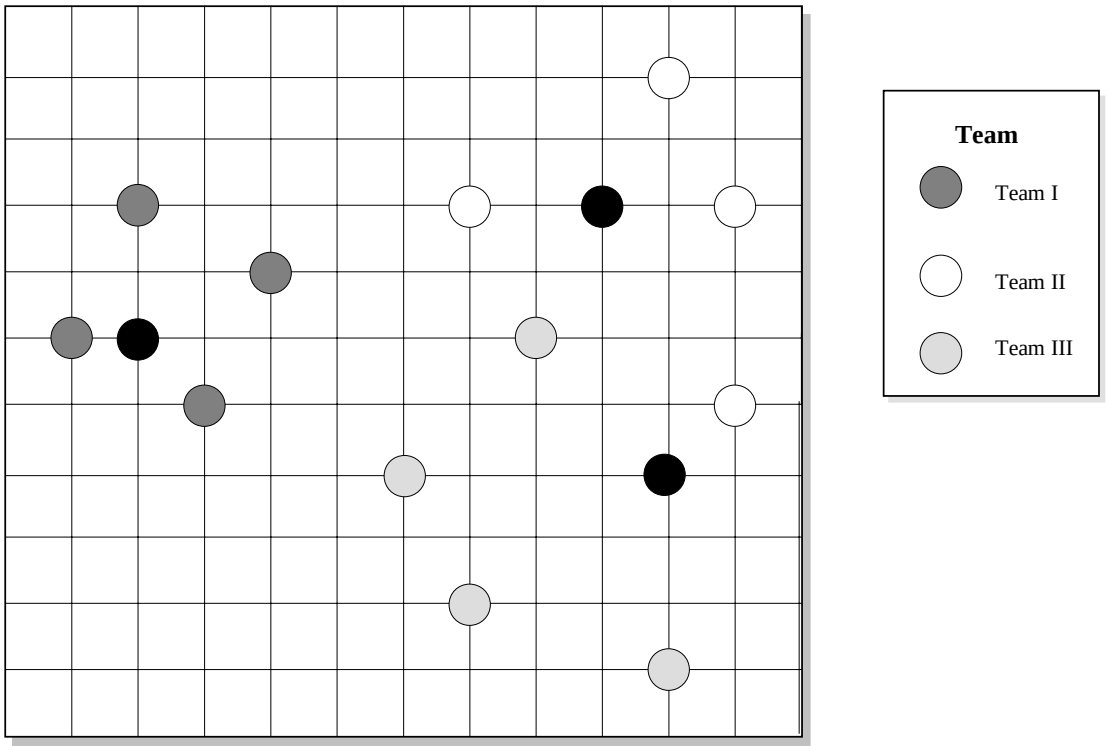

Fig. 2: Typical MTP Configuration

Each team has a fixed set of predator agents that work in capturing an assigned prey agent. In our experiments, no team member may switch to a different team, and no communication between teams is done for the purpose of a single team capturing its prey.

2.2.1 Control Strategy for Capture

The strategy used by each team is called the *controlling agent* strategy [7], [10], [14]. In reference [7] the area around a prey agent was partitioned into four quadrants by using diagonal lines that cross at the prey agent's location. Each predator agent then tries to occupy a location within its assigned quadrant to attain a *Lieb configuration*. Once the predator agents have attained the *Lieb configuration*, they use a set of "Lieb rules" that determines how each of the predator agents will move to capture the prey.

In our study, each team has one controlling agent, and the other three agents are subordinate. The controlling agent directs each of the predator agents (including itself) to its assigned capture regions. Once each predator agent is in its assigned capture region, the controlling agent directs the group to converge on the prey agent using the Lieb rules.

2.2.2 Coordination and Communication States

The focus of this paper is coordination between teams. The controlling agent of one team is permitted to communicate with the controlling agents of the other teams. None of the subordinate team members communicate with any agent from another team. The coordination in these experiments consists of selecting appropriate prey agents for each of the teams. While the experiments used three teams, the algorithm is designed to work with a larger number of teams.

Initially, each team is assigned a prey agent. Based on the relative location of assigned prey during the pursuit, two teams may agree to coordinate their efforts by swapping their assigned prey agents. The transition diagram shown in Fig. 3 illustrates the different states and communications that lead to switching prey agents between two teams.

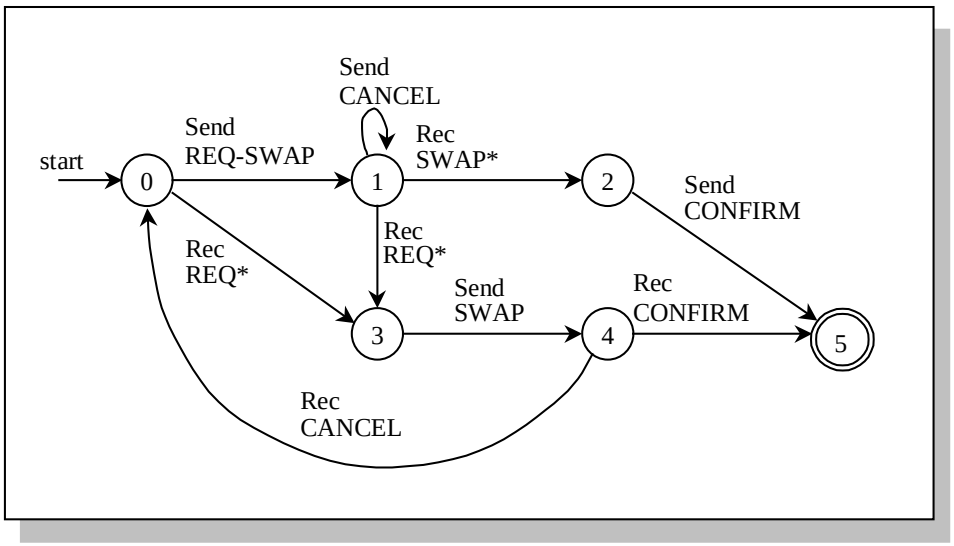

Fig. 3. Transition Diagram for Swapping Prey

The starting state of the transition diagram is state 0. In this state, a controlling agent listens for messages requesting swaps (REQ) from other teams or it can send a message requesting a swap (REQ-SWAP). If a controlling agent sends a REQ-SWAP message, it then moves to state 1 and sets a flag indicating the REQ-SWAP message has been sent. The decision to send a REQ-SWAP message is based on ranking all of the prey agents with respect to their distance from each team member. If the nearest prey agent is not the assigned prey agent, the controlling agent broadcasts a REQ-SWAP message.

In state 0 or 1, if the controlling agent receives a REQ message from another team, it evaluates the message to determine if there is an acceptable swap, denoted as a REQ* message. An acceptable swap message is one in which the prey agent being offered in the swap message has a better ranking than the currently assigned prey agent. If the REQ* message is received, the agent enters state 3.

In state 1, the controlling agent is listening for a reply to its own request to swap message (REQ-SWAP). If a reply message (SWAP) is received, the controlling agent

evaluates the message to determine if it is an acceptable swap (SWAP*). If the SWAP message is not acceptable, the controlling agent sends the other team a CANCEL message informing them the swap is not acceptable and then waits in state 1 for another reply. If the SWAP message is acceptable, the controlling agent enters state 2.

In state 2, the controlling agent sends a CONFIRM message to the offering team and then enters state 5. If the controlling agent receives other SWAP messages while in state 5, it sends a CANCEL message to any team sending a SWAP message.

A controlling agent may also reach state 5 by replying to another team's REQ-SWAP message. After receiving an acceptable request to swap message (REQ*), the controlling agent enters state 3. In this state, the controlling agent sends a SWAP message to the other team and enters state 4.

From state 4, a controlling agent waits for a CONFIRM or CANCEL message from other teams. If a CONFIRM message is received, the controlling agent enters state 5. If a CANCEL message is received, the controlling agent returns to state 0 where it transitions to state 1 if it has already sent a REQ-SWAP message.

State 5 is the terminal node. In this state, the two controlling agents swap prey assignments and end their current coordination cycle. If any SWAP messages are received, a CANCEL message is immediately returned.

Most coordination cycles do not end in this manner. For most cycles, the controlling agent enters state 1 or state 4 and waits a user-specified amount of simulated time before timing-out and ending the current coordination cycle. This timing-out behavior is required when coordination is not needed for the current problem configuration. Timing-

out is done only in state 1 or 4 since states 2 and 3 are temporary states the agent moves through without waiting.

3.0 Experimental Results

The MTP prototype is implemented using the MICE testbed [5] with some minor modifications to improve performance. The MTP prototype was used to analyze its performance with several different starting configurations. The starting configurations for the results presented in this paper use the configuration shown in Fig. 4. In this configuration, three prey agents are randomly distributed within the shaded area in the center of the grid. This random distribution of prey ensures that a given team will, on average, have no advantage over the others. However, to make valid comparisons between different trials, we restricted the placement of the predator agents. For each trial, each pursuit team is placed in a different grid corner with its team members grouped together as closely as possible. This starting configuration ensures some randomness in starting configurations; however, the number of runs in a given experiment makes it possible to obtain valid performance comparisons for different values of the coordination parameters.

Each team is initially assigned a specific prey that is placed at random within the center region of the grid. The assigned prey can be exchanged by teams cooperating with each other. The controlling agents of each team communicate to exchange prey agents only during coordination cycles. At other times, the controlling agent directs its team members in capturing the assigned prey agent.

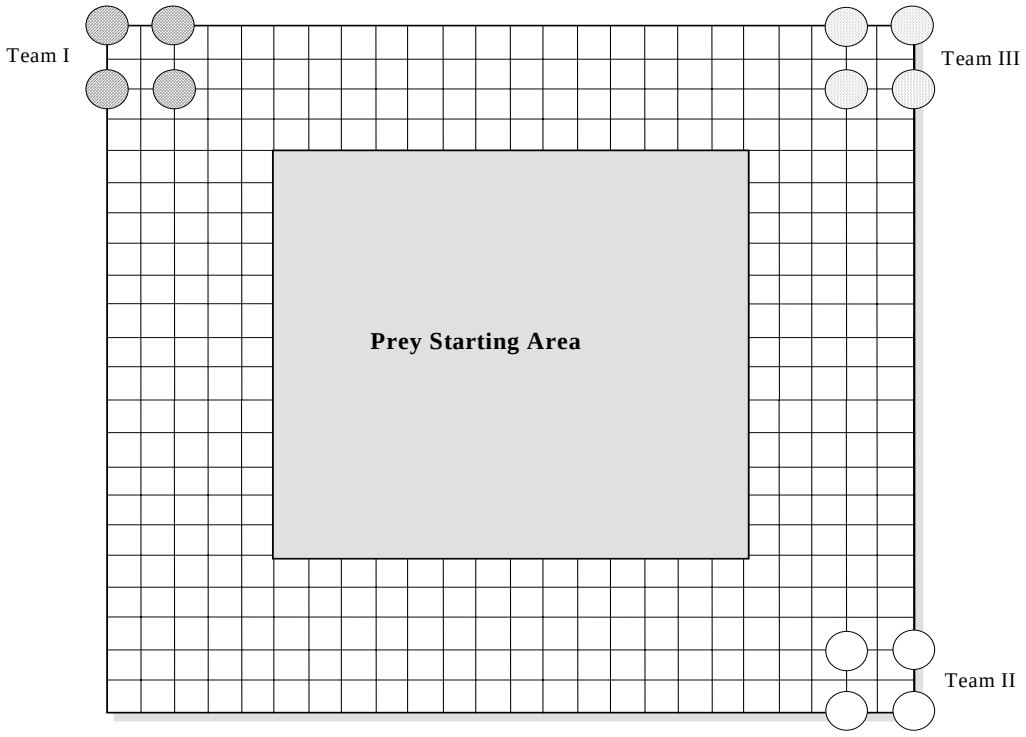

Fig. 4: Starting Configuration

The MICE testbed uses a simulated clock to keep track of time. Each clock tick is denoted as Δt. The amount of time required for an agent to move or for a message to be transmitted between agents is user definable. For this study, the total simulated time for a message to be sent by one agent and received by another agent is assumed to be 3 simulated time units: Δt to send a message, Δt for delay in the communication channel, and Δt for message receipt. The time allowed for an agent "move" (the total perceive, reason, and act cycle) is set to $100\Delta t$, an amount of time much greater than the time spent in communicating with another agent. Thus, the total time for a message to be sent and received is 3 Δt, which is only 3% of the move time. These values were selected to represent the domain of naval vessels pursuing enemy submarines—a domain in which

communication time is much smaller than movement time. Times can be changed to suit other domain applications.

3.1 Coordination Parameters and a Performance Metric

Coordination costs are also measured in simulated time units. In most coordination cycles, the teams do not switch prey agents but run out of the allotted interaction time. If teams can minimize the number of non-productive coordination cycles, they save time and improve their performance. For the experiments described below, the teams' cooperative behavior is determined by three coordination parameters:

Nth-Move Value

Teams repeatedly wait *N* moves before attempting to communicate. In the first set of experiments (described in section 3.2), *N* is set to 5 so teams made 5 moves between each attempt to swap prey agents. By increasing *N*, teams spend less time attempting to communicate but may lose opportunities for swapping prey assignments.

Time-Out Counter

The wait time in states 1 and 4 of the transition diagram, Fig. 3, is specified by the time-out counter. This parameter limits the time spent by the teams in the coordination cycle and is measured in simulation time units of Δt. (In the experiment described in Section 3.2, the time-out counter is set to 15 Δt; that is, during a coordination cycle, each agent waits 15 clock ticks for messages from other agents before timing out and selecting a move.)

Team-Distance Threshold

This parameter is used to decide *when to stop attempting* to swap prey agents. As the team members converge on their prey agent, the sum of each agent's straight-line distance to the prey agent decreases. Once this distance,

$$D = \sum_{i=1}^{4} |\, location(agent_i) - location(prey) \,|,$$

falls below the *Team-Distance Threshold*, the team captain will not enter a coordination cycle. The motivation for this restriction is that once a team converges on its prey agent, it is improbable that switching prey agents will be of benefit. The minimum distance, $D = 4$, is attained when the team has captured its prey agent. (In the set of experiments described in Section 3.2, the threshold was set to 5, a value that permitted a near-maximum number of coordination attempts.)

In the second set of experiments (Section 3.3), we introduce a performance measure *P* that is the weighted, linear combination of both the average time (*Avg.*) and the maximum time (*Max*) for the three teams:

$$P = w \cdot Avg + (1-w) \cdot Max,$$

where *w* is the weight. We chose a weight of 1/3, which is conservative in that the contribution of the slowest team has twice as much influence as the average of all three teams. Other metrics could be chosen depending on what is important to the system builder.

3.2 Initial Results with and without Coordination

The first experimental results confirm that coordination among teams on prey selection improves the average performance of the teams. Teams attempt to coordinate on every 5th system move and are allowed 15 Δt to complete their interactions. After 15 Δt, any coordination attempt is aborted. Each trial was run to completion in that all prey agents were captured. The results from running 30 trials using the starting configuration are shown in Table 1. The *Average-Moves* and *Average-Time* columns represent the average number of moves and the average time cost for the 30 trials for each team. The third column is the average time cost of all three teams. The last column shows the standard deviation for time costs with and without coordination.

	Average Moves	**Average Time**	**Average Time/Move**	**StdDev Time**
Without Coordination	43.7	4376	100	551
With Coordination	35.3	3670	104	680
Percent Improvement	**19.1**	**16.1**	**- 4.0**	**-23.4**

Table 1: Comparison of Coordinated versus Noncoordinated Teams

Table 1 shows that coordination reduces the average number of moves a greater percentage than the average time. This difference is because coordination reduces the number of moves but increases the time per move; coordination requires spending time sending messages and waiting in coordination cycles. In this first experiment, coordination reduced the number of moves by 19.1%, but the time per move increased 4% resulting in a 16.1 % reduction in the total time required to capture all three prey agents.

However, coordination increased the variance of the results measured in both the number of moves and the expended time. Swapping depends on the positions of the prey and predator agents. Since prey agents start in random positions, not all trials required swapping. In trials where swapping had no benefit, the predators still spent time in coordination cycles only to find that a swap would not help.

To better illustrate this result, a frequency count of the results is shown in Table 2. The frequency count is the number of trials that fall within a 500 unit time range. For example, in the *No Coordination* column at the time of 4000, the count is 7: the number of trials whose average time is between 3500 and 4000. The probability distribution graph is shown in Fig. 5. The frequency count for the coordinated system has a local maximum at 4000 compared to the local maximum of 5000 for the system without coordination. This indicates the cooperating agents benefited from the time spent trying to interact.

Time	No Coordination	Coordination
2000	0	0
2500	0	1
3000	0	4
3500	1	8
4000	7	10
4500	9	2
5000	11	4
5500	1	1
6000	1	0
6500	0	0

Table 2: Frequency Count

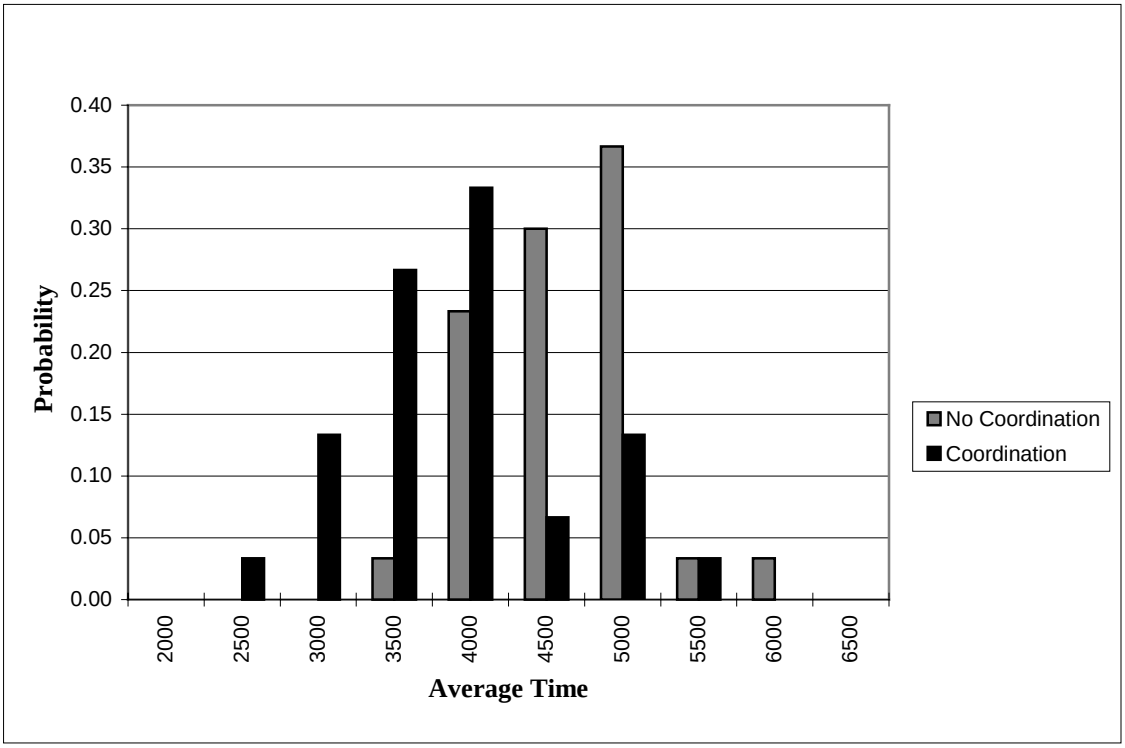

Fig. 5: Probability Distribution of Average Time for Systems with and without Coordination

3.3 Experiments to Determine Improved Values for Coordination Parameters

Experiments were run using the starting configuration of Fig. 4 and varying the *Team-Distance Threshold*. The *Time-Out Counter* is set at 15 Δt and the *Nth-Move Value* is set at 5—the teams enter a coordination cycle of 15 Δt every fifth move. The results shown in Table 3 are averages over 25 trials for each value of the *Team-Distance Threshold*. The *Average-Time* and *Average-Moves* columns are the average time and average number of system moves for the teams to capture their prey. The *Maximum-Time* and *Maximum-Moves* columns are the average time and number of moves for the last team to capture their prey—the team that required the maximum number of system moves

and simulated time. The last column shows the value of *P*, the overall performance metric defined above.

Team-Distance Threshold	Average Time	Maximum Time	Average Moves	Maximum Moves	Standard Deviation Moves	*P*
05	3647	4774	35	46	9	4398
10	3603	4472	35	43	7	4182
15	3415	4412	33	43	8	4080
20	3412	4358	33	42	8	4043
25	3601	4396	35	43	7	4131

Table 3: Comparison of Team Distance Thresholds

Shown in Fig. 6 are the values of *P* plotted for each of the values of the *Team-Distance Threshold*. The values of *P* are based on an average of 25 trials for each value of the *Team-Distance Threshold*. The best performance is obtained with the *Team-Distance Threshold* set at 20. This value is the total of each team member's distance, measured in units of grid squares, from the location of the team's prey agent. When the team's total distance to the prey is less than 20, the team captain will not enter a coordination cycle. If the *Team-Distance Threshold* is set above 20, then too many coordination opportunities are missed. If the *Team-Distance Threshold* is set below 20, then (1) there are too many coordination cycles taking place in which the teams do not switch prey agents and (2) there is too much time spent waiting without cooperating. The value of 20 seems to be a good balance between having too few opportunities to interact and wasting too much time in unproductive coordination cycles.

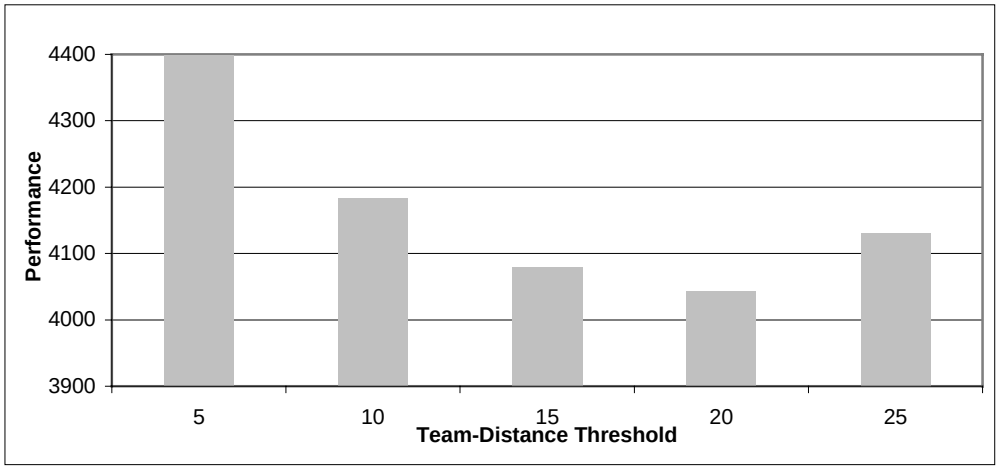

Fig. 6: Performance vs. Team Distance Thresholds

The next two experiments focus on finding an improved number for the *Nth-Move Value* given that the*Team-Distance Threshold* is set at 20 and the *Time-Out Counter* remains at 15 Δt; that is, teams enter a coordination cycle of 15 Δt unless the *Team-Distance Threshold* is 20 or less. The results shown in Table 4 are averages over 25 trials for each value of the *Nth-Move Value*. Shown in Fig. 7 are the values of *P* plotted for each of the values of the *Nth-Move Value*.

Nth-Move Value	Average Time	Maximum Time	Average Moves	Maximum Moves	Std-Dev Moves	*P*
05	3484	4357	34	42	7	4066
10	3819	4781	37	47	8	4460
15	3528	4466	35	44	8	4153
20	3595	4579	35	45	8	4251
25	3701	4625	36	46	8	4317
30	3659	4733	36	47	9	4375

Table 4: Comparison of Different Nth-Move Values

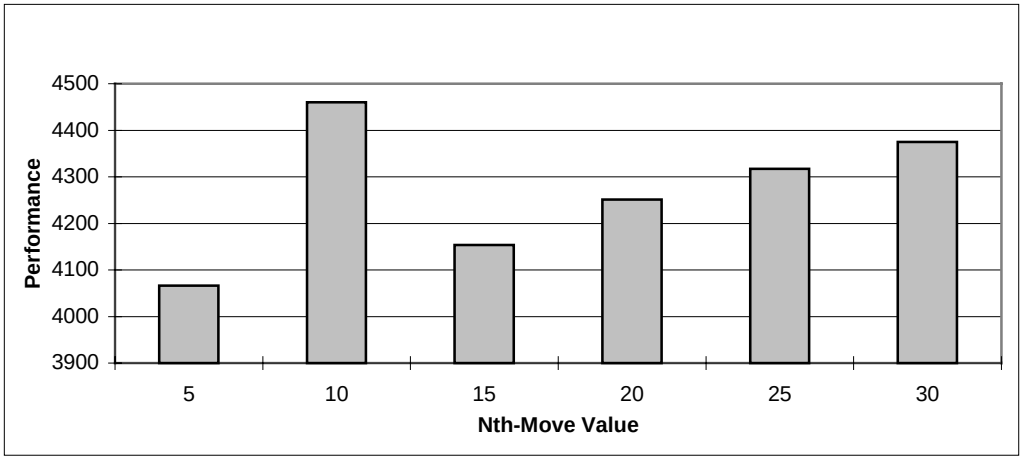

Fig. 7: Performance vs. Nth-Move Values

The best result is obtained with the *Nth-Move Value* set at 5. However, there is no clear pattern indicated by the graph; the maximum performance value occurs at $N = 10$, but this value is inconsistent with the rest of the results. Based on this anomaly, we examined all values of N from 1 to 10, again with the *Team-Distance Threshold* set at 20 and the *Time-Out Counter* at 15 Δt. The results of this experiment are shown in Table 5 and the performance rating, P, is charted in Fig. 8, which shows that the optimum value for N is 5.

Nth-Move Value	Average Time	Maximum Time	Average Moves	Maximum Moves	Standard Deviation Moves	*P*
1	3913	5124	36	47	9	4720
2	3567	4600	34	44	8	4256
3	3607	4582	35	44	8	4257
4	3475	4456	34	43	8	4129
5	3479	4411	34	43	7	4100
6	3679	4641	36	45	8	4320
7	3713	4802	36	47	9	4439
8	3585	4641	35	45	9	4289
9	3646	4698	36	46	8	4347
10	3756	5112	37	50	11	4660

Table 5: Comparison of Lower Nth-Move Values

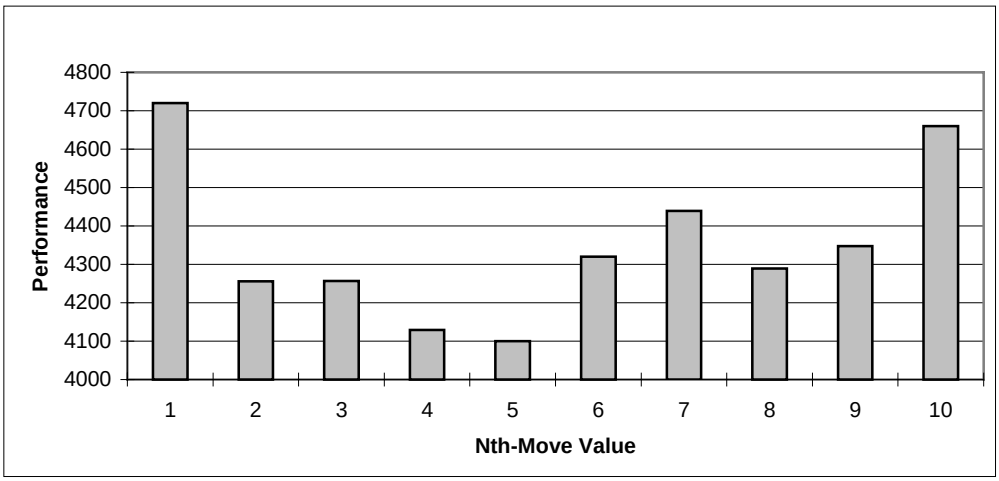

Fig. 8: Performance vs. Lower Nth-Move Values

The final set of experiments finds an improved value for the *Time-Out Counter* with the *Team-Distance Threshold* set at 20 and the *Nth-Move Value* set at 5. The *Time-Out Counter* is varied from 5 Δt to 35 Δt in increments of 5 Δt with the results shown in Table 6 and Fig. 9.

Time Out Counter	Average Time	Maximum Time	Average Moves	Maximum Moves	Std-Dev Moves	*P*
5	3898	4772	38	47	7	4481
10	3691	4490	36	44	7	4224
15	3458	4389	34	43	8	4079
20	3619	4376	35	42	6	4124
25	3577	4736	35	46	9	4350
30	3767	4918	36	47	9	4534
35	3698	4945	35	47	10	4529

Table 6: Comparison of Time-Out Counter Values

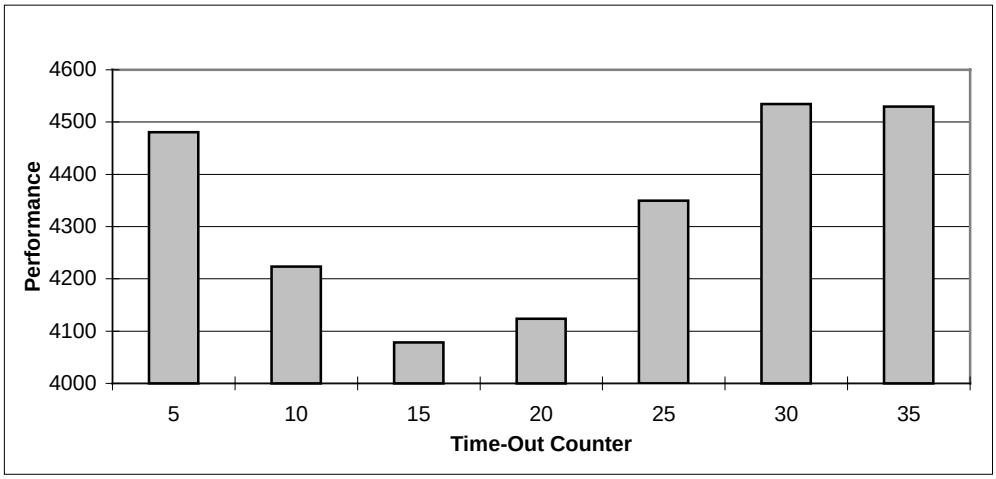

Fig. 9: Performance vs. Time Out Counter Values

3.4 Summary of Experimental Results

Table 7 summarizes all our experimental results and shows the performance improvement for both the initial and optimized coordination parameters. The *Moves* and *Time* columns represent the average number of moves and the average time for 25 trials. Prior to optimizing the coordination parameters, the improvement over a noncoordinated system was 19.1% for the average number of moves; this value increased to 22.2% after optimization. The average time improvement increases from 16.1% initially to 21% after

optimization. Optimization also lowers the time per move since fewer unsuccessful coordination cycles are attempted.

	Moves	**Time**	**Time/Move**
No Coordination	43.7	4376	100.0
Initial Coordination Results	35.3	3670	104.0
1st Pass Improved Coordination Results	34.0	3458	101.7
1st Pass Improvement with Coordination (%)	19.2%	16.1%	-4.0%
2nd Pass Improvement with Coordination (%)	22.2%	21.0%	-1.7%

Table 7: Comparison After Optimizing Coordination Parameters

In addition, the following table summarizes the effect of changing the value of *w* in the performance metric on the best values found in searching the parameter space:

	***w* = 0 (Slowest team only)**	***w* = 1/3 (Favor slowest team)**	***w* = 1 (Avg. time)**
Team-Distance Threshold	20	20	20
Nth Move Value	5	5	4
Time-Out Counter	20	15	15

Table 8: Best Parameter Values

Table 8 shows that for different weighting values the multiple team pursuit task is invariant to the *Team-Distance Threshold* and nearly invariant to the *Nth-Move Value*. This result is encouraging because it suggests that these two parameters are largely independent of each other. The only parameter value that varied significantly with the weighting factor was the *Time-Out Counter*, which requires a larger value $w = 0$. This result suggests that if the most important performance factor is the time for the slowest team to capture the prey, then the deliberation cycle should be extended.

4 Conclusions

Our contributions are (1) the definition of three new parameters that can be used to tune the performance of a multiagent system, (2) the introduction of a simply-computed and flexible performance metric, (3) an experimental testbed that allows the system designer to vary the parameters and measure the results, and (4) experimental results that provide improved parameter values for the multiple team pursuit problem. In a broader context, the process of searching the parameter space itself is perhaps even more important than the specific values determined from the experiments.

The testbed has provided a useful tool for deriving new coordination parameters and running simulations to determine better parameter values. Two of the three parameters introduced, the *Nth-Move Value* and *Time-Out Counter*, are general enough to be used for any multiagent system that has communication or coordination costs between teams or individual agents. The third parameter, *Team-Distance Threshold*, while problem specific, is relatively simple to derive yet effective in improving performance. Many problem domains will likely have similar problem specific coordination parameters that can be effective in helping to determine when coordination should be attempted.

Our experimental results show that even simple types of coordination can be effective but that the system performance metric should include the costs of coordination. For our experiments the teams perform best if they look for opportunities to change prey assignments about every 5 moves and spend no more than about 15-20% of move time waiting for replies from other teams. Tuning the system to determine when coordination is most beneficial increases the effectiveness of the system by lowering the number of

unsuccessful coordination cycles. Finally, tuning the coordination parameters reduces the average time per system move while also reducing the average number of moves.

Future work includes the automation of the parameter adjustment process by building a machine-learning algorithm. The algorithm would take as inputs the coordination parameters and performance metrics and produce as outputs improved values for the parameters. There may be other parameters and performance metrics that need to be defined.

5 References

1. M. Benda, V. Jaganathan, and R. Dodhiawala, "On optimal cooperation of knowledge sources," *Technical Report*, Boeing Advanced Technology Center, Boeing Computer Services, Seattle, WA, September 1986.
2. A. H. Bond and Les Gasser, *Readings in Distributed Artificial Intelligence*, Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1988.
3. M. L. Dowell, “Learning in Multiagent Systems,” Ph.D. Dissertation, Department of Electrical and Computer Engineering, University of South Carolina, Columbia, SC, 1995.
4. M. L. Dowell and R. D. Bonnell, "Learning for Distributed Artificial Intelligence Systems," *Proceedings of the Twenty-Third Southeastern Symposium on System Theory*, Robert Werner (Ed.), 1991 pp. 218-221.
5. E. H. Durfee and T. A. Montgomery, "MICE: A flexible testbed for intelligent coordination experiments," in *Proceedings of the 9th Workshop on Distributed Artificial Intelligence*, September 1989, pp. 25-40.

6. E. H. Durfee and J. S. Rosenschein, "Distributed Problem Solving and Multi-Agent Systems: Comparisons and Examples," in *Papers from the Thirteenth International Workshop of Distributed Artificial Intelligence*, Mark Klein, Chair, AAAI Press, Menlo Park, CA, 1994.
7. L. Gasser, N. F. Rouquette, R. W. Hill, and J. Lieb, "Chapter 3: Representing and Using Organizational Knowledge in Distributed AI systems," in *Distributed Artificial Intelligence, Vol. 2,* L. Gasser and M. N. Huhns (Eds.), Morgan Kaufmann, San Mateo, CA, 1989, pp. 55-78.
8. M. N. Huhns *et al.*, "DAI for Document Retrieval: The MINDS Project," in M. N. Huhns, ed., *Distributed Artificial Intelligence*, Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1987.
9. R. E. Korf, "A Simple Solution to Pursuit Games," in *Proceedings of the Eleventh International Workshop on Distributed Artificial Intelligence,* Glen Arbor, Michigan, February 1992, pp. 183-194.
10. M. Merx, "The Influence of Organizational Hierarchies on the Performance of Distributed Intelligent Systems," Ph.D. Dissertation, The University of South Carolina, Columbia, SC, 1989.
11. U. Mukhopadhyay, L. M. Stephens, M. N. Huhns and R. D. Bonnell, "An Intelligent System for Document Retrieval in Distributed Office Environments," *Journal of the American Society for Information Science*, vol. 37, no. 3, 1986, pp. 123-135.
12. M. P. Singh, M. N. Huhns and L. M. Stephens, "Declarative Representations of Multiagent Systems," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 5, No. 5, October 1993, pp. 721-739.

13. L. M. Stephens and M. Merx, "Agent Organization as an Effector of DAI System Performance," *Proceedings of the Ninth Workshop on Distributed Artificial Intelligence*, Eastsound, Washington, Sept. 1989, pp. 263-292.
14. L. M. Stephens and M. Merx, "The Effect of Agent Control Strategy on the Performance of a DAI Pursuit Problem," *Proceedings of the Tenth International Workshop on Distributed Artificial Intelligence*, Bandera, Texas, October 1990, pp. 263-292.
15. G. Weiss, "Learning to Coordinate Actions in Multi-Agent Systems," *IJCAI-93*, Chambery, France, August 1993, pp. 311-316.
16. J. Denzinger and M. Fuchs, “Experiments in Learning Prototypical Situations for Variants of the Pursuit Game”, Proceedings of the Second International Conference on Multiagent Systems (ICMAS-96), 1996, AAAI-Press, pp. 48-55
17. L. Gasser, "Distributed Artificial Intelligence," in *AI Expert*, Vol. 4, No. 7, July 1989.
18. M. L. Dowell and R. D. Bonnell, "Knowledge Acquisition for Distributed Artificial Intelligence Systems," USC Technical Report ECE-MLD-060-90, University of South Carolina, Columbia, SC, 1990. Mike,