

Emergent Power Delineation in Multi-Agent Swarms

Brian McLaughlan and Henry Hexmoor

Computer Science Dept, Southern Illinois University, Carbondale, IL, 62901, USA

{brianm, hexmoor}@cs.siu.edu

Abstract

An organization is well delineated when all its members have a relative power ranking that is neither too high nor too low. Since responsibility and authority follow fully deconflicted patterns, a delineated organization is well suited for solving problems. We have developed a low-latency, agent-level algorithm for non-deterministically establishing power levels and provide a metric for quantifying the appropriateness of rankings. An agent's rank is appropriate if it has a unique chain of superiors leading to the highest-ranking individual.

1. Introduction

Increased operational tempo of modern warfare, the fight against terrorism, challenges in joint urban operations (JUO), and the *autonomic* requirements from networked systems in the business world require a transformation of interactions in intelligence and operational processes. Automatic methods are needed for adaptation and self-organization of authority ranking, herein power, among individuals in the organization (Deloach and Matson, 2003). There are clear tradeoffs for organizational structure and the military-style rigid chain of hierarchical command structure may not be ideal (Horling and Lesser, 2005).

Agent organizational structures range from peer-to-peer democracies as in (Rahman, et. al, 2004), socialistic organizations (Hexmoor, et. al., 2004), to highly structured tree organizations (Yokoo, et. al, 1988). The power in a peer-to-peer organization is similar to a senate floor where each contingency is thoroughly debated. Assuming that the debating entities are benevolent, an ideal solution is eventually found. In such a system, the individual messages are typically important and have a high latency, and are therefore vital to the outcome. Although such a system is guaranteed to find the ideal solution, reaching that solution may take a long time. In contrast to democratic structures, exact commands issued in military hierarchies may not be ideal, but potentially fatal delays caused by debate are eliminated. The tree organization is more appropriate when (a) time-critical decision are needed and (b) high latency debate is not possible due to poor, limited, unreliable, or dynamic communication pathways.

Many methods of electing leaders have been proposed in studies of computational collective intelligence. A few election algorithms provide for a single group leader, e.g., (Cideon, et.al., 1988), while others include provisions for a group of coordinators of equal power (Chen et. al., 2002). Whereas some are deterministic, others are non-deterministic, and a few combine elements of both deterministic and emergent

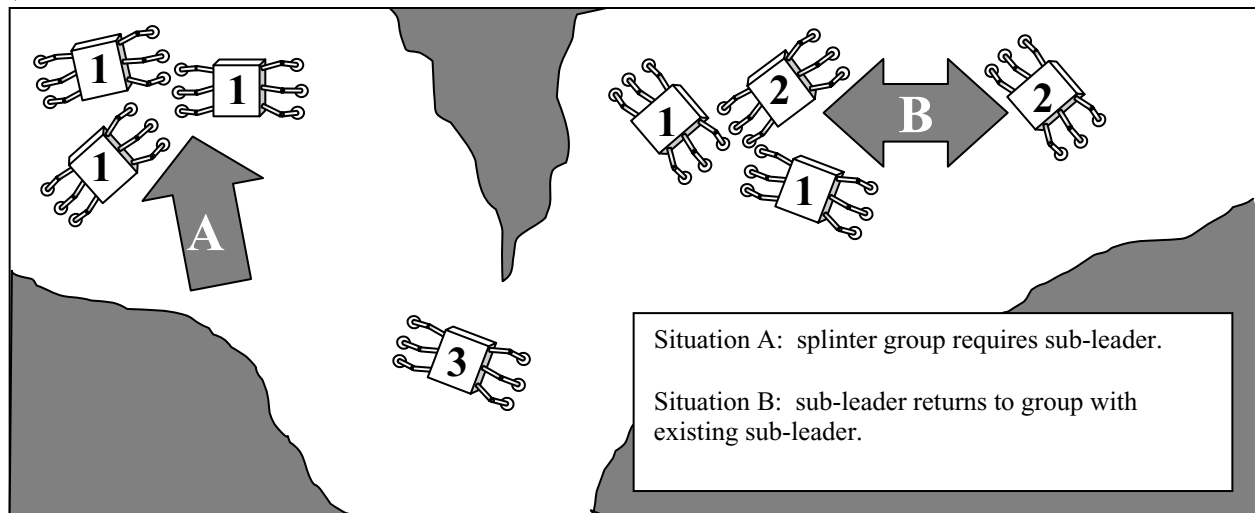


Figure 1: Multi-agent power delineation cave exploration example

components.

We make the following assumptions. Communication bandwidth is limited and unreliable. Agents are anonymous and have uniform properties including an equal capability. Individual agents form a multi-agent swarm (Eberhart, et. al., 2001). Swarms have been studied in ant colonies (Gaertner, et. al, 2005; M. Dorigo and T. Sttzl 2004; Kong, et. al., 2005), fish schools (Hut, et. al, 1992; Tu, et. al, 1994), and bird flocks (Reynolds, 1987, 1999). Individuals within a swarm react to stimuli from their environment and nearby peers.

Despite lack of global awareness by individuals, swarms have emergent properties that allow interesting group behaviors to emerge. We are particularly interested in emergent election algorithms (Anthony, 2004).

Typically, traditional election algorithms do not fully delineate power structure among individuals. Given assumptions of unreliable communication and the need for instantaneous decision-making, arbitration by a single leader is not desirable. We seek a ranking system where each agent knows its relative status in relation to all other agents.

To motivate the situation where such a fully delineated algorithm is desirable, consider the case of a swarm of robotic cave exploration depicted in Figure 1. The robots are identical in mobility, sensing, and communication. The cave has a finite set of tunnels (branches) unknown to robots. Once the swarm is released the goal is full exploration.

We assume poor quality of communication due to the cave environment. Messages are abrupt and brief. One robot will elect itself the nominal leader to ensure that all tunnels are explored. As the robots disperse to map their surroundings, cave walls will limit communications to and from the leader depicted by situation marked A in Figure 1. This situation precipitates the need for the promotion of

sub-leaders that can coordinate the actions of their immediate group members and relay that information to the leader.

As leaders and sub-leaders return to communication range depicted by situation marked B in Figure 1, inconsistencies due to multiple leaders and conflicting commands will occur. The robots would then need to deconflict the chain-of-command.

In both situations A and B, individual robots must proactively and unilaterally alter their rank within the chain-of-command. For instance, one lieutenant that hears another lieutenant could not order the second lieutenant to demote itself to a sergeant and expect the second lieutenant to hear that command. The first lieutenant would have to self-determine its own rank, perhaps by demoting itself to resolve the potential conflict with subordinates in range of both lieutenants.

Finally, an ideal algorithm would allow some functionality, even if limited, while setting up the power structure rather than wait until the ultimate organization has emerged.

Our proposed algorithm in the following section provides the non-deterministic functionality appropriate for low-latency, ad-hoc applications while maintaining the robustness of a system that can reorganize its structure to meet the dynamic requirements of the situation.

2. Algorithms

We make the following assumptions. Communication among agents is unreliable or expensive. Agents are identical and any position or task could be appropriate for any agent. There is no personal gratification in garnering power; i.e., it is just as important to have the lowest rank as it is to possess the highest rank. Thus, demoting oneself to avert a conflict is just as viable to self-promotion.

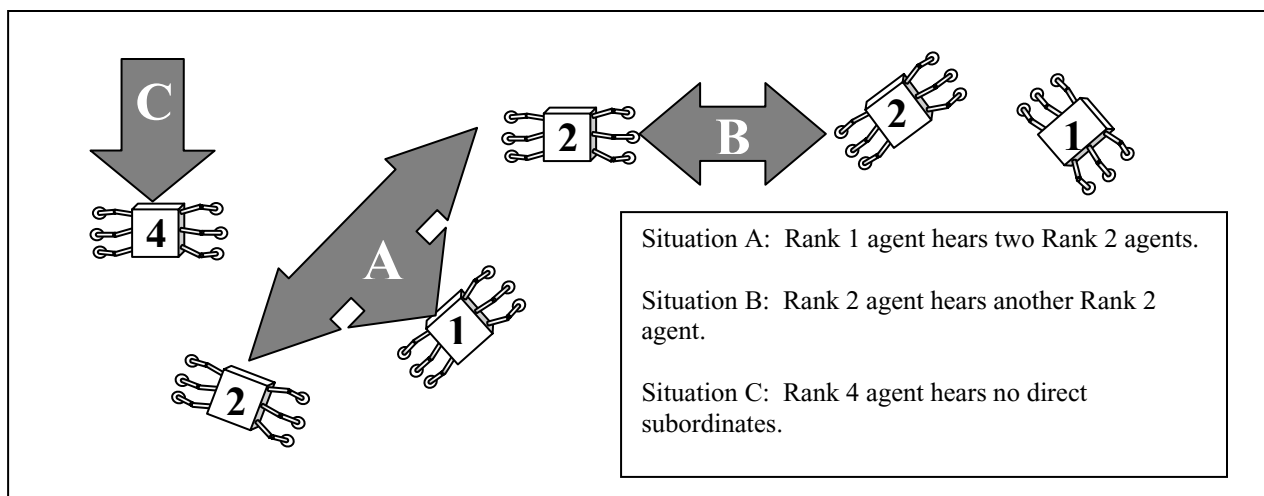


Figure 2: Multi-agent power delineation situations

Message sending agents must assume that their messages are lost. Therefore, agents must take proactive steps to avert conflict rather than to instruct others.

A few distinct situations are depicted in Figure 2. Situation marked A depicts a situation where an agent hears multiple superiors. If those superiors had been of differing ranks, then there would be no conflict; if conflicting orders were heard, the agent would obey the command issued by highest-ranking superior. However, the agent in this situation would hear orders from two equally ranked superiors, causing a conflict of action.

Situation B occurs when an agent at a particular rank hears a peer with the same rank. Since the agent can hear the peer, it can assume that others in their proximity can hear both of them. Thus, the agent must take proactive steps to remedy the situation.

Situation C is the least critical, and involves a single agent that cannot hear any direct subordinates, in this case, no agents of rank 3. In this situation, the agent will realize that its rank is excessively inflated and take steps to correct it.

The basic outline of our algorithm is shown in Figure 3. The details of each step are described next. The system designer can modify each of these steps to provide optimal performance for the specific application in consideration.

Individual agents in the system determine the fitness of their surroundings by listening to broadcasts of their neighbors. At relatively regular intervals, each agent broadcasts its current rank. This could be a special broadcast or could easily be added to the agent's normal communication. When not broadcasting its rank, an agent will listen for other broadcasts and tabulate the results.

An agent will continue this listening process for some period of time, after which it will process the data and determine whether it should change its rank to remedy

1. Listen
2. Check for Superiors
 - If multiple equally ranked superiors are found, self-promote above them.
3. Check for Peers
 - If peers are highest ranking, self-promote above peers.
 - Otherwise, if superiors exist, self-demote below peers.
4. Check for Subordinates
 - Self-demote to one rank above highest-ranking subordinate.
5. Announce Rank

Figure 3: Basic Algorithm Outline

a perceived conflict. This period of time is roughly standard but has some variation to prevent multiple agents from triggering at the same time. For instance, if the listening period is 60 seconds, the agent might trigger at 58 seconds, or at 63 seconds, or at some other time around 60 seconds.

After listening for rank announcements, the agent then analyzes the data that it has collected. First, it considers Situation A – multiple superiors at the same rank. If an agent hears rank announcements by two superiors of the same rank, it could signal a conflict. It is possible that the two superiors are not even aware of a conflict. For instance, they could be out of communication range with each other and not realize that an agent is receiving commands from both of them.

Due to the inability to rely upon communication to announce the conflict, the agent that notices the problem must correct the situation. The simplest way to remove the conflict is to not be affected by directives given by the offending superiors. Thus, the agent will self-promote to a rank above the immediate superiors and will proceed to coordinate the actions of its former superiors.

Whether it has promoted itself or not, the next step is for the agent to check for peers in order to eliminate Situation B. This situation can be considered similar to Situation A, except that the two offending agents are close enough to hear each other's announcements and realize that a potential conflict exists. Because of this situation's similarity to the previous situation, a knee jerk reaction by the agent could be to simply self-promote one rank to eliminate the conflict. While that is acceptable, the agent could just as easily demote itself one rank to gain the same localized effect. The greater impact of either action on the system, although subtle, can be different depending on the composition of the power structure of the system. Therefore, the agent must decide whether promoting or demoting would be more beneficial to the system as the agent perceives it.

It is difficult for an individual agent to be aware of the impact of its actions when its only source of information is the limited communication it is receiving from nearby agents. Therefore, rather than suggesting a complicated set of rules to determine which direction to change rank – rules that could easily be inappropriate for the situation due to incomplete information – the agent consults a simple guideline that utilizes information that it has already gleaned and analyzed. If the agent currently has a superior, then it demotes itself. If it and its peers are the highest-ranking agents that it hears, it retains its local superiority by self-promoting. The purpose of changing rank in this fashion is to preserve relative positioning as much as possible. For instance, if the agent in question were to self-promote despite having a superior, it could create a cascade that causes the superior to lose its status as leader. This could lead to unnecessary shifts in attitude,

goals, and even data as the new leader takes on new responsibilities from the old. Thus, by demoting when a superior exists, the superior is not exposed to the possibility of cascading effects.

The final situation, Situation C, involves an agent that notices that it has no direct subordinates. There is no need for agents to have a rank above the initial rank (typically 0 or 1) unless it has subordinates, and then the rank only needs to be one greater than its subordinates. Thus, the agent tabulates all occurrences of subordinate announcements and self-demotes to one rank above the highest subordinate that it hears. Such a demotion conserves ranks while maintaining the same relative power structure.

The problem presented in Situation C is not theoretically fatal. Since power is relative, a system that contains agents with well-compacted ranks will function identically to a system whose ranks have been multiplied by some factor. The danger exists when the theoretical simulation is transposed onto a real-world system. In large systems with hundreds of agents running on processors that are attempting to conserve limited memory, a power delineation algorithm that did not attempt to eliminate rank inflation could easily overflow the space allocated for storage of its own rank as well as the space allocated for the ranks that it hears. With rank conservation in place in a system of n agents, the largest rank value is $n+1$. While a stable system will have at maximum a rank n , rank $n+1$ is a possible transitory state. Thus, the system designer can plan the hardware for the agents according to the maximum number of agents projected in the system.

After all three situations have been analyzed, the only remaining task for the agent is to announce its rank. It announces its rank whether it has changed or not. This allows other agents to maintain a record of what agents are present and able to communicate.

It is necessary to quantify the system fitness with respect to the power structure to determine if the system is fully deconflicted. Such a metric would not be available to the individual agents due to their localized ability to receive information. In fact, one might say that each agent is already intuitively quantifying the local system fitness during its analysis of communication. However, a system-level fitness metric is still needed, particularly when simulating the ability of the system and may be useful in some specific applications as well.

Before system fitness can be quantified, a definition of what constitutes fitness must be identified. We say an agent is *fit* when it can trace a communication path to the highest-ranking agent in the organization. In such a case, the orders given to the agent can be assumed to be in accord with the system as a whole. Any agent whose communication path ends at a local maximum is considered unfit. Similarly, the system is *maximally fit*

when every agent is fit. The fitness of the system is computed as the percentage of agents that are fit.

One case that needs mentioning is when two or more agents hold the same maximal rank. In such a case, the potential confliction is the same as if all but one of those agents were local maximums. Thus, the system can treat all but one maximally ranked superior as local maximums. Fortunately, states such as these are typically transitory as the system evolves to a more stable state.

As an aside, an individual agent is considered *absolutely fit* when its communication path greedily follows the agent's highest-ranking superiors. Such a distinction between being merely fit and absolutely fit is only necessary if orders relayed from superior to subordinate through an intermediary "middle manager" do not reference the originating superior. For instance, agent X may receive orders from a locally maximum superior of rank 3 and another superior of rank 2, which is in turn receiving orders from the globally superior agent of rank 4. In this case, agent X is fit, but not absolutely so. If the agent of rank 2 were to give agent X an order that conflicted with orders given by agent of rank 3, the orders from agent 2 would be ignored. However, if agent 2 included in the order that the command originated from agent 4, agent X would know to obey the order. This simulation assumes that such information is given in the orders. Thus, the distinction is not needed for this case.

3. Observations and Conclusions

To analyze the effectiveness of this algorithm, a simulation was implemented that tested several metrics. First, we tested how quickly and accurately the algorithm would converge on a solution. Next, we tested the algorithm's ability to re-stabilize an established power structure after changes had been made to the communication network. Finally, we determined the algorithm's capability to adapt to changes made to the communication network before the power structure has stabilized. These three tests together formed a simulation run.

Since the algorithm is most appropriate for ad-hoc communications, further deliberation is needed for the communication network of the simulation. Unfortunately, that ad-hoc nature means there is no real "standard" or "typical" network; the network may range from several physical agents in close proximity that can communicate with most of their neighbors to a completely random and disjointed organization, although most situations would likely be somewhere in between. Therefore, the simulation examines the extremes. First, it tests a very "regular" network where agents can be conceptualized as physical entities situated in a line with each agent able to talk to a certain number of agents to either side of it. The second network is completely random; each agent can communicate with some semi-random number of other agents.

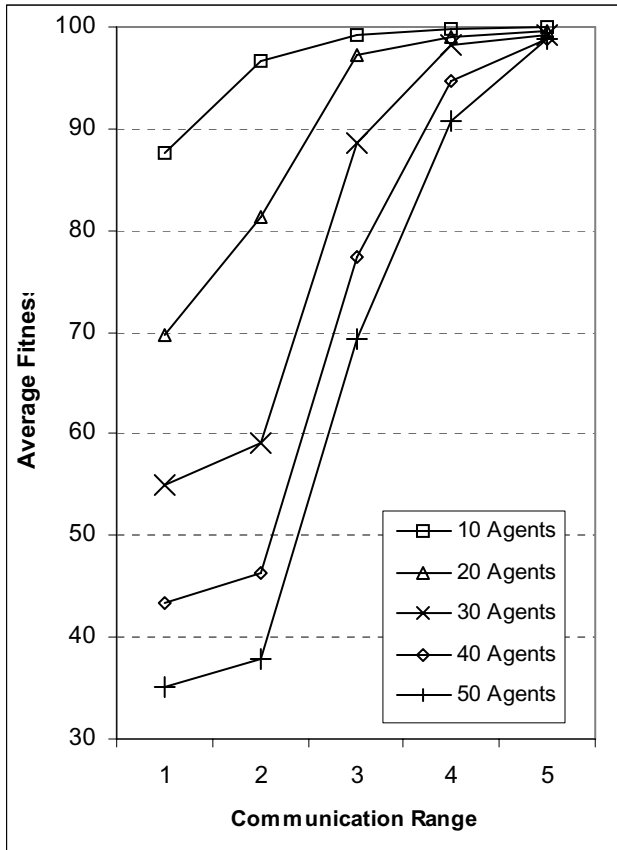


Figure 4: Average fitness of system

The basic test was straightforward and provided a baseline. A regular or random network was generated with each agent at a rank of zero and was allowed to converge to a solution. To determine a trend, the test was performed several thousand times over a wide range of network parameters (number of agents vs. number of communication pathways). The results showed the algorithm easily found an optimal solution for the regular network. The algorithm's ability to handle the random network varied with the exact structure of the network, but usually converged to an acceptable solution.

The second test builds upon the first and represents an agent that moves into or out of communication with the group. After the first test is finished, an agent is randomly added or removed from the network, the network is allowed to re-stabilize, and the system notes the time taken to re-converge on a solution as well as the change in fitness of the system. The time taken to re-stabilize depended on the rank of the agent added or removed; higher ranking agents caused more cascading effects upon arrival or departure than lower ranked agents. However, the algorithm rarely had trouble returning the system to the same fitness level as it had before the change; the change in final fitness is statistically zero.

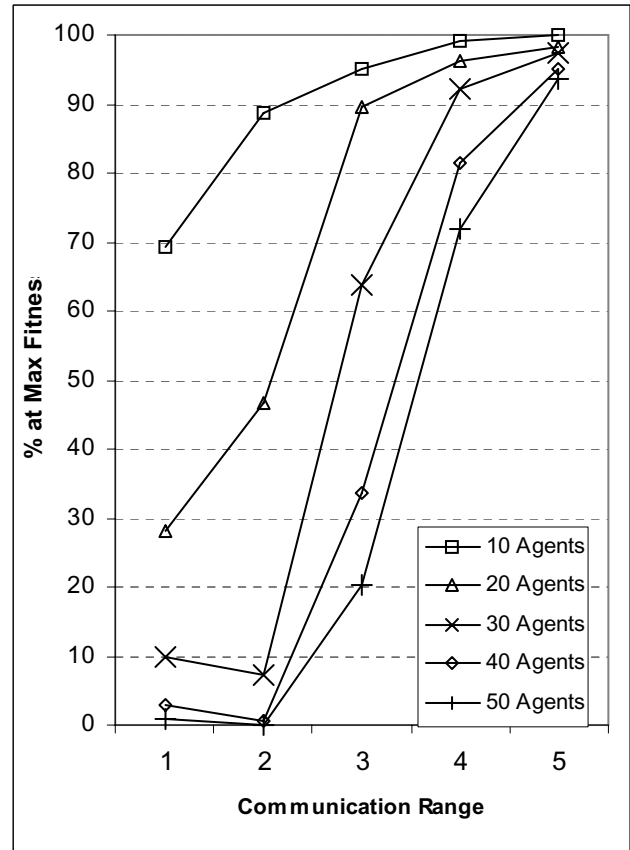


Figure 5: Percent of systems at max fitness

The third test examines the algorithm's ability to handle changes while the algorithm is in the process of converging. The same communication network that was used in tests one and two is used, but the power rankings are reset. The system is then allowed to converge, but at some random point before convergence, a node is added or removed as in test two. If the algorithm converges before the random point, the test is executed again with a different random point. After convergence, the system fitness is compared to the fitness achieved in test one. As found in test two, the change performed in test three had no statistical effect on the system's ability to converge on a solution.

Interesting features can be found by examining the data collected by these experiments. First, it is interesting to note that the average system fitness depicted in Figure 4 follows the same pattern as the percentage of systems that achieved maximal fitness shown in Figure 5. In particular, there exists a point where the return on investment is significantly greater than at other points. In other words, increasing the communication ability beyond a point produces little additional benefit. The exact point where this occurs is most likely application dependant, but it is interesting to note that the point tends to slide slightly higher as the population size increases.

Similarly, it is also remarkable that this point of greatest return does not slide at the same rate as the change in population. This indicates that an increase in population does not require a similarly proportioned increase in communication ability to maintain the same level of fitness. This is one area that requires more study.

Finally, a note on the anomaly found in Figure 5's lower communication ranges. There exists a dip in percentage at maximum fitness from communication range 1 to range 2. Interestingly, this dip does not exist on the average fitness graph in Figure 4. Although the average fitness consistently rises, the percentage that reaches maximal fitness temporarily dips. At this point, we cannot explain this dip, although it is statistically correct and not spurious data. Fortunately, the dip is minor, but it is nevertheless a thought-provoking feature.

The algorithm presented in this paper is not guaranteed to find the ideal solution to delineating the power structure of an organization. In fact, it is not guaranteed to find any solution. However, as shown in the simulation, it usually finds an optimal chain of command as well as providing other key benefits. These benefits include the ability to handle changes to the system structure during computation. In dynamic systems, it would be infeasible to require the agents to "maintain their position" while the algorithm performed its calculations. Similarly, the algorithm does not require any particular communication to be received. Such low latency communication reduces the time and bandwidth needed and is particularly useful in typical field conditions. Additionally, the algorithm provides some benefit while in the process of running. Although typically not immediately optimal, the algorithm provides the agents with an incrementally better solution while in the process of converging on an ideal state. A system designer is advised to weigh the non-deterministic nature of the algorithm against the dynamic benefits to determine if its use is appropriate.

References

R. Anthony. 2004. Emergence: A Paradigm for Robust and Scalable Distributed Applications. In proceedings of IBM Conference on Autonomics, 132-139.

B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance. *Ad Hoc Wireless Networks*. ACM Wireless Networks Journal, 8(5), September 2002.

S. DeLoach and E. Matson. 2003. Autonomously Reorganizing Information Systems. In proceedings of International Conference on Advanced Technologies for Homeland Security (ICATHS). Storrs, CT.

I. Cidon, I. Gopal and S. Kutten. 1988. New Models and Algorithms for Future Networks. In proceedings of 7th ACM Symposium on Principles of Distributed Computing, Toronto, Ontario, Canada. 74-89.

M. Dorigo, T. Sttzl. *Ant Colony Optimization*, MIT Press, 2004.

R.C. Eberhart, Y. Shi, J. Kennedy. 2001. *Swarm Intelligence*. Morgan Kaufmann.

D. Gaertner and K.L. Clark. 2005. On Optimal Parameters for Ant Colony Optimization Algorithms. *Proceedings of International Conference on Artificial Intelligence*. 83-89.

H. Hexmoor and K.M. Krishna. 2004. Social Reasoning and Collaboration Among a Large Group of Robots. In proceedings of the Fifth International Symposium on Collaborative Technologies and Systems. 224-229.

Horling, B. and Lesser, V. 2005. A Survey of Multi-Agent Organizational Paradigms. *Knowledge Engineering Review*.

A. Huth and C. Wiesel. 1992. Simulation of Movement of Fish Schools. *J. Theoret. Biology*. 156:365-385.

M. Kong and P. Tian. 2005. A Convergence Proof for the Ant Colony Optimization Algorithms. *Proceedings of International Conference on Artificial Intelligence*. 118-121.

A. Rahman and H. Hexmoor. 2004. Negotiation to improve Role Adoption in Organizations. *Proceedings of International Conference on Artificial Intelligence*. 476-480.

C. Reynolds. *Flocks, Herds and Schools*. 1987. *Computer Graphics*, 21:25-34.

C. Reynolds. 1999. Steering Behaviors for Autonomous Characters, In proceedings of Game Developers Conf. 763-782.

X. Tu, and D. Terzopoulos. 1994. Artificial Fishes: Physics, Locomotion, Perception, Behavior. In proceedings of ACM SIGGRAPH. 42-48.

M. Yokoo, E.H. Durfee, T. Ishida, and K. Kuwabara. 1988. The Distributed Constraint Satisfaction Problem: Formalization and Algorithms. *IEEE Transactions on Knowledge and DATA Engineering* 10(5).