Cooperating Mobile Agents for Mapping Networks

Nelson Minar, Kwindla Hultman Kramer, and Pattie Maes

MIT Media Lab, E15-305 20 Ames St, Cambridge MA 02139, USA
(nelson,khkramer,pattie)@media.mit.edu
http://www.media.mit.edu/~nelson/research/routes-coopagents/

To Appear in the Proceedings of
the First Hungarian National Conference on Agent Based Computing
May 24, 1998

Abstract. Contemporary computer networks are heterogeneous; even a single network consists of many kinds of processors and communications channels. But few programming tools embrace, or even acknowledge, this complexity. New methods and approaches are required if next-generation networks are to be configured, administered and utilized to their full potentials. The growing field of mobile agents research seeks to address problems in this domain. In this paper we describe a strategy for mapping a network using a collection of cooperating mobile agents. We present results from a simulation of such a system and discuss the relationship between diversity of the agent population and overall efficiency of the system.

Knowledge of the topology of a computer network is a prerequisite for all higher-order interactions between nodes on that network. In current systems, routing maps are usually generated in a centralized (and often human-mediated) manner. Our mobile-agents approach, in contrast, is highly distributed and decentralized with agents spread across the network working to accumulate connectivity information. The agents in our system move around the network, discover topology information, and carry with them the information they have gathered as they explore. Our agents also collaborate; when they meet on a node they share information with each other, so an individual agent can acquire knowledge about parts of the network that it has never visited.

We apply three different types of agent algorithms to a mapping task on a static network and comparing their effectiveness. We find that agent cooperation greatly improves system performance. Furthermore, we find that diversity of behavior between collaborating agents also improves the efficiency of the system as a whole. Agents that consistently exhibit identical behaviors duplicate one another's efforts, to the detriment of overall performance. As agents interact with one another – and so have the potential to become "like" their peers by learning from them – preserving behavioral diversity becomes a major challenge. Efficient division of labor in the absence of centralized control is a subtle, important problem.

Keywords: mobile agents, mobile code, distributed, decentralized, cooperation, collaboration, networks, routing, diversity

1 Introduction

As computer networks grow in scale, scope, and importance and as their topologies become more dynamic, the complexity of configuring, administering, and utilizing networks grows as well. We are studying new ways of thinking about and designing networks and network applications; in particular, we are building systems of cooperating mobile software agents. Mobile agents are programs that are free to move themselves and their thread of execution from computer to computer [20]; the environment the agents live in is the computer network. Agents can also interact with each other, sharing information to better accomplish their goals.

We believe that mobile agents offer exciting possibilities for the design of next-generation systems. The modularity of the mobile agent approach encourages the design of flexible, adaptive architectures. The malleability and dynamic reconfigurability of movable code allow us to re-tailor systems in the real world by adding to and changing running systems. And the metaphor itself — mobile, encapsulated programs — provides a new framework for thinking about computer networks.

This paper investigates a straightforward problem for a system of cooperating agents. We present results from an initial round of simulations along with a discussion of some strengths and weaknesses of this particular approach. The problem, building a map of a computer network, was chosen because it servers as a good test case for studying the behavior of mobile agents working together to solve a problem.

The mapping of network topology is a problem suitable for exploring the dynamics of mobile, multi-agent systems. The data of interest, the connectivity of nodes in the network, is inherently distributed and decentralized. The potential users of that data (for example, an application that needs to open a socket between a specific pair of hosts) are likewise spread across the system. Typically, routing problems are solved by collecting the far-flung data of network connectivity in a centralized store. Our experimental alternative is to send mobile agents out into the network to discover information about connectivity, building a map which can be used to make routing decisions. This decentralized solution presents an interesting alternative for network mapping and routing, particularly in cases where traditional centralized solutions are difficult to implement because the network topology is itself dynamic.

2 Experimental Model

The results we present here are from a simulation of mobile agents living in a network of interconnected nodes that work cooperatively to build a map of the network. We chose a network topology consistent with related work by our colleagues [12]; the nodes in the system are modeled as radio-frequency tranceivers distributed throughout a two-dimensional space. Because the nodes are relatively short-range tranceivers, this type of network is generically similar to many other common types. In particular, the average packet in such a system requires multiple hops to travel from source to destination and, relatedly, resident mobile agents need to be quite peripatetic in order to effectively gather connectivity data.

The goal of the agents is to map the network they are living in, to learn about the connectivity of the nodes and build a model of the network topology. An agent can map the system by simply traveling around the network and learning about connectivity first hand. In addition, agents can also cooperate: when two agents travel to the same node, they can share information with each other about their maps of the network and therefore collaborate in the mapping task.

Our model is implemented with a simple discrete event, time-step based simulation engine. Every step of simulated time an agent does three things. First, the agent learns about all of the edges on the node it is on. This information, which the agent itself has experienced, is added to the agent's store of firsthand knowledge. Second, it learns everything it can from all of the other agents currently on the same node. This information, learned from another agent is stored separately as "rumored" facts. Finally, the agent chooses another node and moves there.

Our experiments test three different algorithms governing the movement of agents. As a baseline we first implemented "random" agents, which simply move to a random adjacent node every update. "Conscientious" agents are more sophisticated, choosing at each time step to move to an adjacent node that they have never visited or have visited least recently. This algorithm is similar to a depth-first search of the network. Conscientious agents base their movement decisions only on first-hand knowledge, ignoring information learned from other agents when determining where to go. In contrast the third type of agents, "superconscientious agents," also move preferentially to nodes that have not been explored, but they use both their own experience and learned data from their peers in deciding which nodes to move to.

Our model makes several important simplifications and assumptions. Among the most significant are:

- The network is physically situated in two dimensions.
- Computation (including communication between agents) is free and instantaneous. Bandwidth is also free, immediate, and unlimited. Taken together, this pair of simplifications means that all agents execute at the same speed. No agent can be slowed down by computational or communications bottlenecks. This results in a simulation that is well-suited to the examination of "best-case" agent dynamics but not to the evaluation of systems with resource scarcity.
- Links between nodes are reliable and bi-directional. This is assumed to be guaranteed by an underlying protocol layer.
- Learned connectivity data in the system is always correct. The network topology does not change dynamically and neither byzantine failure nor pathological behavior is modeled. This greatly simplifies the task of the agents.

Our simulation system consists of 1200 lines of Java code implementing a discrete event scheduler, a graphical view, a data-collection system, and the simulated objects themselves: network nodes and mobile agents. In order to compare results across population sizes and algorithms, we chose a single connected network consisting of 250 nodes with 2522 edges (figure 1) for all experiments. Each run of the simulator begins with a population of randomly-placed agents and continues until every agent in the system has perfect knowledge — complete information about every node and edge in the communication graph. We measure the time when all agents have perfect knowledge rather than just one, to find when the entire system has converged. We believe this is the

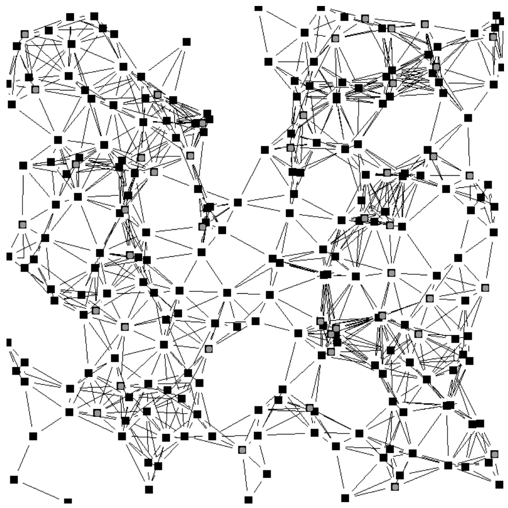

Fig. 1. Network used for all experiments

appropriate measure for this inherently distributed system. The time at which all of the agents have acquired perfect knowledge is called the "finishing time". We examine in detail the finishing times and the accumulation of knowledge about the map over time for each of the three kinds of agents.

3 Presentation of Results

The results we present here compare three different agent algorithms at eight population sizes (1, 2, 4, 7, 10, 25, 50 and 100). All data is calculated from 400 independent runs of each parameter setting with random initial agent placement. A complete table of our data is presented in the appendix; particular results are extracted here for discussion.

3.1 Single Agent Results

Our first experiment was conducted on a population of only one agent. There is no opportunity for agent cooperation here; these results simply show the basic behavior of the two core types of algorithm, random and conscientious. (In a population of size one, conscientious agents are equivalent to superconscientious agents.) Unsurprisingly, the random agent algorithm performs quite poorly when compared to conscientious agents. Random agents take an average of 5300 time steps to map the graph with a standard deviation of 3500. By contrast, a lone conscientious agent finishes the map in an average of 480 steps with a standard deviation of 101. A comparison between these two types of agents of the curves for knowledge over time (figure 2) shows the average relative performance of these two algorithms as well as the distribution of independent runs.

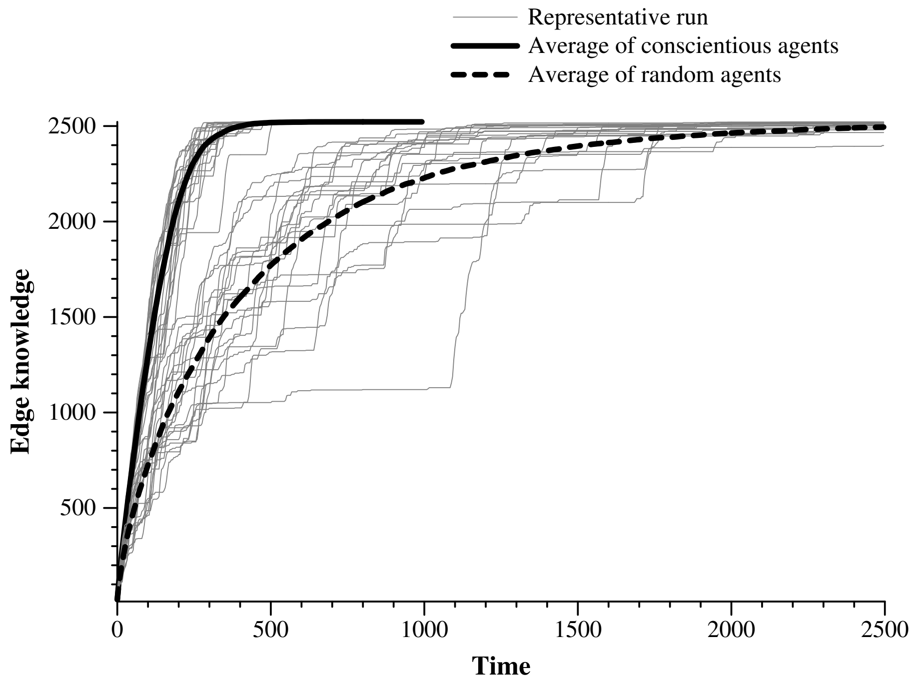

Fig. 2. Knowledge over time for single agent systems. Avg. of 400 runs and 20 representatives.

3.2 Effect of Collaboration

The single agent results provide a basis for comparison of the effects of agent cooperation. When multiple agents work together, coexisting on the graph and sharing knowledge as they meet, the gap between the random and conscientious strategies narrows significantly. As seen in the table in the appendix, two random agents interacting and learning from each other do almost twice as well as one alone, finishing on average in 3100 steps instead of 5300. And while two conscientious agents working together are better than two random agents by more than six-fold, when the population size increases to 100 the conscientious agents finish only 32% more quickly than the random agents.

Collaboration is good. The reason is simple: an agent alone can only learn a limited amount each time step, while agents that share data often acquire a large amount of new information in a single interaction. This effect is clearly visible in figure 3 where the near-vertical jumps in individual knowledge reflect an agent learning from an experienced neighbor. Sharing information is so beneficial that for larger population sizes the positive effect of learning from peers almost compensates for the very poor movement strategy of the random agents.

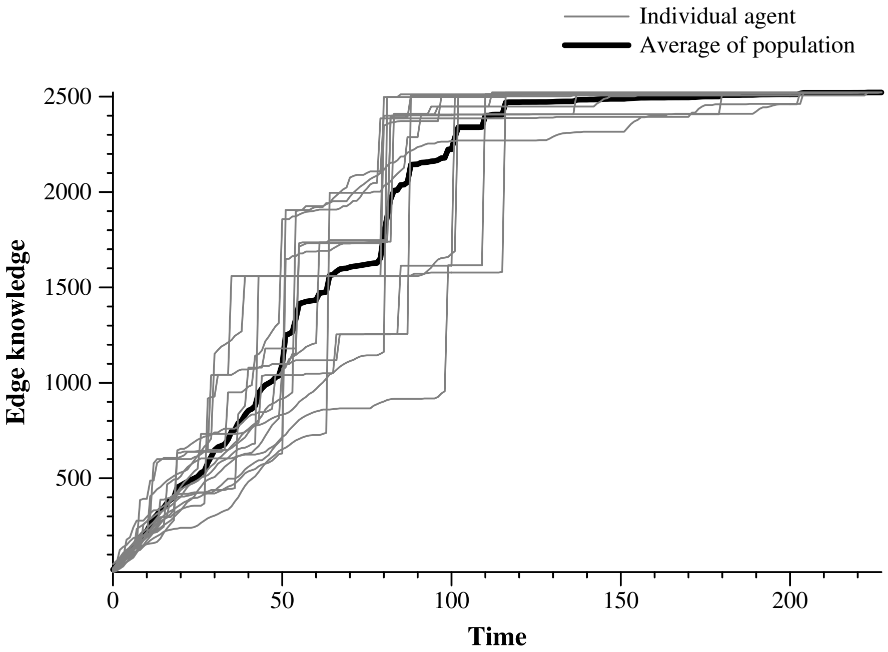

Fig. 3. Knowledge over time for one run of 15 conscientious agents.

3.3 Effect of Population Size

As discussed in section 2, our results ignore issues of resource scarcity. In our model increasing the number of agents has no cost. While distributed systems such as this tend to scale well, in a more detailed simulation (or a real-world situation) per-agent consumption of bandwidth and cpu cycles would need to be accounted for. With these restrictions the point where decreasing returns no longer justify increases in agent resources becomes important. As a graph of finishing time for different populations shows (figure 4), the marginal benefit of increasing population size is far more substantial for small populations than for large ones.

3.4 Superconscientious Agents: A Surprising Result

Our third class of agents are "superconscientious": they use both first-hand and peer-obtained information to make movement decisions. We expected that these agents would be the most successful of the three types. After all, the more information that is factored into a decision the better that decision should be.

It turns out that in small populations superconscientious agents do perform the best, but only just barely. Ten superconscientious agents map the graph in 245 time steps on average, while ten conscientious agents take 271 time steps to finish. But the slight edge that the superconscientious agents have with small populations disappears as the

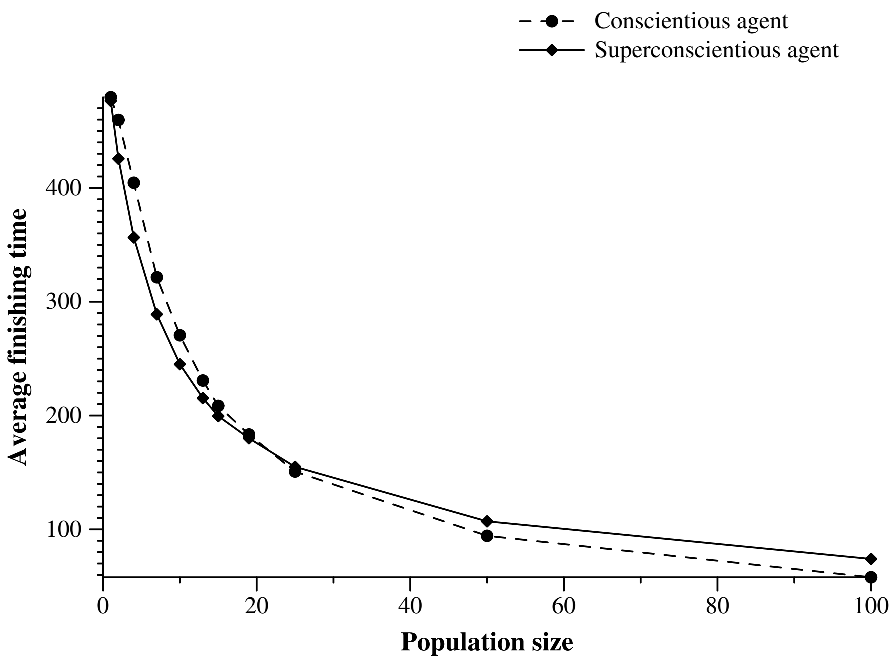

Fig. 4. Finishing time at different population sizes. Average of 400 runs.

number of agents increases; 100 superconscientious agents average 74 steps to finish and 100 conscientious agents take only 58. It should be noted these differences are subtle; for all but the largest populations the averages are within a standard deviation of each other.

The result that superconscientious agents are no more efficient at mapping than the less-deliberative conscientious agents prompted additional investigation into the behavior of the two algorithms. Upon examination, one difference was immediately obvious: superconscientious agents meet far more often over the course of a run than conscientious agents do. A population of 100 superconscientious agents averages 192 meetings per time step, while 100 conscientious agents average only 42 meetings. A closer look at these meeting-event statistics over time (figure 5) shows that the number of meetings increases over the course of a run in the case of the superconscientious agents, while it remains steady throughout the run for conscientious agents. Similarly, the number of nodes occupied by at least one agent decreases throughout a superconscientious run but stays roughly constant for a conscientious agent trial. At the end of a typical run, 100 conscientious agents are distributed across 82 different nodes on average, while in the superconscientious runs the 100 agents are crowded together on only 36 nodes.

Clearly, these two types of agents behave quite differently. Superconscientious agents tend to cluster together over the run while conscientious agents remain evenly dispersed. Because of this effect, superconscientious agents tend to duplicate one another's exploratory efforts and the system as a whole is less efficient.

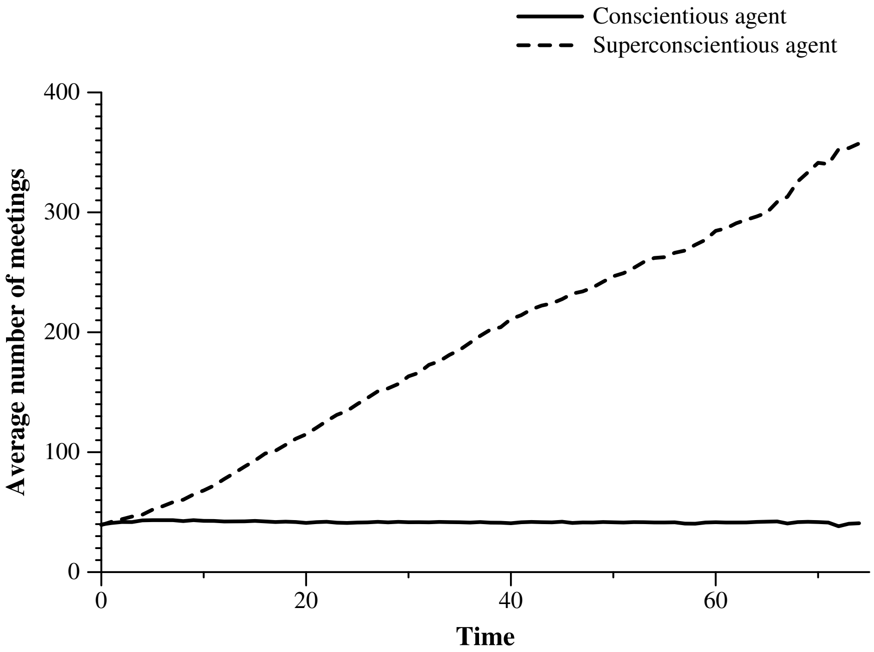

Fig. 5. Meetings over time for 100 conscientious and superconscientious agents. Average over 400 runs.

Why do superconscientious agents cluster together? When two agents meet, they share all information. Conscientious agents assimilate this data but still make their movement decisions based entirely on their own first-hand experience. In contrast, superconscientious agents use all information they have learned in making their decisions. So two superconscientious agents that meet and share information often find themselves making subsequent choices based on identical sets of data, and therefore choose identical paths. Two such twin agents diverge only when multiple equally-good options present themselves (in which case each chooses randomly).

Our results show that the performance of a multi-agent algorithm depends not only on how efficiently each individual agent acts but how the agent population as a whole distributes its effort. Specifically, superconscientious agents are inefficiently distributed when they follow one another around in clusters — they are duplicating effort because they are not using their knowledge effectively. More generally, diversity of behavior is important for the effectiveness of cooperating multi-agent systems. If an agent population is too homogeneous then the benefits of having many agents working together is lost. Effective cooperation requires division of labor.

As a preliminary investigation we studied what would happen if superconscientious agents had a 20% chance of ignoring what they knew and moving randomly. We found that a population of 100 of these agents completed the task in 57 time steps on average, better than the 74 time steps of the ordinary superconscientious agents and about the same as the 58 time steps of 100 ordinary conscientious agents. By contrast, conscientious agents with a 20% random movement probability only completed the task in 62 time steps, no improvement over ordinary conscientious agents. Randomness is an artificial means to induce behavioral diversity; superconscientious agents benefit from it because they are too homogeneous, but conscientious agents already distribute themselves efficiently and so do not benefit from randomness.

4 Related Work

The fields of distributed AI and multi-agent systems have traditionally studied task decomposition and collaboration among sets of agents. Most of the work in these domains has focused on situations where there is a high level of communication and coordination among agents: agents modeling each other [16], for example, or explicit agent negotiation [4] [13] [18]. In particular, work by Lander and Lesser [9] takes a task negotiation approach to the problem of distributed search. Somers [17] and Garijo et al. [6] apply distributed AI techniques to manage networks in a decentralized fashion. Our work is similar in that we also focus on decentralized cooperating agents, but differs in that our agents are mobile, living in the network, and tend to be less knowledge-based.

Mobile agents are an active and exciting research topic, especially as contemporary tools such as Java make mobile agent systems relatively simple to implement. A number of arguments exist as to why the mobile agent approach is potentially useful [3] [20]. Much research has focused on performance enhancement from resource distribution, but we believe that the most interesting and novel possibilities of mobile agent architectures lie in their adaptive nature and inherent flexibility [1] [7]. We share some commonalities of direction with developers of experimental active packet infrastructures [19], including a concern with the dynamic characteristics of network systems and a desire to explore the possibilities of movable bits of code in the context of network protocols. Our mobile agents could be recast as active packets — active routers host the mobile agent programs, which in turn discover the connectivity of the routers and provide topology information to the network.

There are several threads of research on using mobile agents situated in a telecommunications network to manage connectivity and load balancing. Appleby and Steward's work is an early paper suggesting using mobile agents with AI-like strategies to dynamically load-balance a telecommunication network. Followup work using a design inspired by ant behavior [2] [15] extends these ideas in a new direction, using markers in the network environment as a means of indirect inter-agent communication ("stigmergy"). We believe our work is complementary to the ant-based research; the environmentally-mediated approach is efficient and elegant, but trading structured stores of information can be transparent and powerful.

Finally, our particular result regarding the importance of population diversity for efficiency of multi-agent systems has been a theme in biologically-inspired computer system design. Huberman has studied performance characteristics of cooperative processes [8], looking at the concurrent search of a space of possibilities by an ecology of agents and at the effectiveness of hints exchanged among those agents. He also concludes that diversity increases effectiveness. In the field of genetic algorithms [11], the tradeoff between population diversity and efficiency is a well known, if not entirely solved, problem. Results from genetic algorithm work [5] [10] [14] suggest that explicit efforts to maintain population diversity can make genetic algorithms perform more efficiently.

5 Future Research

We are currently working to enrich our results along two fronts: adding to the complexity of the network model and increasing the sophistication of agent collaboration.

In real-world situations network graphs do not stay constant. Nodes fail temporarily or are taken down for maintenance. New nodes are added to the network periodically. And in radio-frequency systems, nodes often physically move so that links in the graph alter as transceiver proximities change. We believe that our mobile agent approach will be particularly well suited for dynamic networks.

Adding dynamic connectivity to the simulation requires that agents treat system data with greater subtlety. As nodes move and edges change, old information about network connectivity becomes incorrect. Similarly, we plan to also consider the effects of byzantine and pathological failures within the system. The possibility of these failure modes implies that unverified information from other agents is somewhat less reliable than information obtained directly from the environment. Next-generation agents will employ three strategies to gauge the validity of data: discarding old information, discounting rumors from potentially-unreliable sources, and making logical inferences about plausibility of data given other available topological information.

Further afield, we would like to explore the effects of agent specialization on overall system effectiveness. As the results of the superconscientious agent trials suggest, division of labor can lead to remarkable changes in overall performance. We are considering two potential strategies to support diversity: geographic focus and resource specialization. As an example of geographic focus, agents could specialize in maintaining accurate knowledge of only specific parts of the network. As an example of resource specialization, some agents might decide to only track information about underutilized paths while others maintain broader network maps. Both approaches could improve the performance of the system as a whole by allowing individual agents to work on different parts of the problem and cooperate to build a complete map.

6 Summary and Conclusion

We have presented analysis of a simulation of mobile agents working together to map a network of computers. While agents that cooperate can collectively solve problems more quickly than agents acting alone, optimizing such a system for maximal performance is subtle. Strategies that appear profitable from the perspective of each individual agent are not necessarily efficient for the system as a whole when pursued by multiple, interacting agents. A major challenge for designers of multi-agent systems is to encourage diversity of behavior in a population of cooperating agents.

References

1. Mario Baldi, Silvano Gai, and Gian Pietro Picco. Exploiting code mobility in decentralized and flexible network management. In *Proceedings of the First International Workshop on Mobile Agents*, Berlin, April 1997. http://www.polito.it/˜picco/papers/ma97.ps.gz
2. G. Di Caro and M. Dorigo. Mobile agents for adaptive routing. In *Proceedings of the 31st Hawaii International Conference on Systems*, January 1998. ftp://iridia.ulb.ac.be/pub/dorigo/conferences/IC.22-HICSS31.ps.gz
3. D. Chess, C. Harrison, and A. Kershenbaum. Mobile agents: Are they a good idea? In *"Mobile Object Systems: Towards the Programmable Internet"*, volume 1222 of *Lecture Notes in Computer Science*. 1997. http://www.research.ibm.com/massive/mobag.ps
4. R. Davis and R.G. Smith. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20(1):63–109, 1983.
5. K. A. DeJong. *An analysis of the behavior of a class of genetic adaptive systems*. PhD thesis, University of Michigan, 1975.
6. M. Garijo, A. Cancer, and J. Sanchez. A multi-agent system for cooperative network-fault management. In *PAAM 96, Proceedings of the First International Conference and Exhibition on the practical applications of intelligent agents and multi-agent technology*, pages 279–294, London, 1996. Practical Applications Company.
7. David Halls. *Applying Mobile Code to Distributed Systems*. PhD thesis, Computer Laboratory, University of Cambridge, June 1997. http://www.cl.cam.ac.uk/users/dah28/
8. B. A. Huberman. The performance of cooperative processes. *Physica D*, 42:38–47, 1990.
9. Susan E. Lander and Victor R. Lesser. Understanding the role of negotiation in distributed search among heterogeneous agents. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 438–444, August 1993. ftp://ftp.cs.umass.edu/pub/lesser/lander-ijcai93.ps
10. Sushil J. Louis and Gregory J.E. Rawlins. Syntactic analysis of convergence in genetic algorithms. In L. Darrell Whitley, editor, *Foundations of Genetic Algorithms*, volume 2, pages 141–151, San Mateo, CA, 1993. Morgan Kaufmann.
11. Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, 1996.
12. R. Dunbar Poor. Hyphos – a self-organizing, wireless network. Master's thesis, MIT Media Lab, 1997. http://ttt.media.mit.edu/pia/Research/Hyphos/
13. Jeffrey S. Rosenschein and Gilad Zlotkin. *Rules of Encounter*. MIT Press, 1994.
14. J. D. Schaffer, L. J. Eshelman, and D. Offutt. Spurious correlations and premature convergence in genetic algorithms. In G. J. E. Rawlins, editor, *Foundations of Genetic Algorithms*, volume 1, pages 102–112, San Mateo, CA, 1991. Morgan Kaufmann.
15. R. Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz. Ant-based load balancing in telecommunications networks. *Adaptive Behavior*, 5(2):169–207, 1997. http://www-uk.hpl.hp.com/people/ruud/abc.html
16. Yoav Shoham. Agent-oriented programming. *Artificial Intelligence*, 60(1):51–92, 1993.
17. F. Somers. Intelligent agents for high-speed network management. In *PAAM 96, Proceedings of the First International Conference and Exhibition on the practical applications of intelligent agents and multi-agent technology*, pages 909–921, London, 1996. Practical Applications Company.
18. Katia Sycara. Multi-agent compromise via negotiation. In L. Gasser and M. Huhns, editors, *Distributed Artificial Intelligence*, volume 2. Morgan Kaufman Publishers, 1989.
19. D. Tennenhouse, J. Smith, W. Sincoskie, D. Wetherall, and G. Minden. A survey of active network research. *IEEE Communications Magazine*, 35(1):80–86, January 1997. http://www.tns.lcs.mit.edu/publications/ieeecomms97.html
20. James E. White. Telescript technology: Mobile agents. In Jeffrey Bradshaw, editor, *Software Agents*. AAAI Press/MIT Press, 1996.

A Table of Results

Data presented here is for 400 runs of the simulation. Each run is for the same network graph (figure 1), but a different random initial placement of agents. We carried out simulations for three different agent movement algorithms, and eight population sizes each. Preliminary results on partially-randomized agents are also summarized here.

Table 1. Table of results of all simulation runs

Agent Type	Pop. Size	Average Finishing Time	Std. Dev.	Average Meetings/Step	Std. Dev.
Random	1	5314.81	3468.99	–	–
	2	3108.18	1801.54	0.0085	0.0035
	4	1670.12	1019.39	0.0521	0.0111
	7	1018.94	465.74	0.1848	0.0267
	10	737.41	416.09	0.3993	0.0460
	25	323.38	159.51	2.6614	0.1839
	50	167.86	68.79	10.8943	0.5157
	100	85.25	29.29	44.1069	1.2899
Conscientious	1	479.63	101.43	–	–
	2	459.70	91.66	0.0082	0.0057
	4	404.51	81.25	0.0489	0.0164
	7	321.46	64.73	0.1717	0.0333
	10	270.50	55.02	0.3742	0.0513
	25	150.79	28.48	2.4997	0.1782
	50	94.33	15.17	10.3580	0.4648
	100	57.89	8.72	42.4973	1.2728
Superconsc.	1	476.44	95.10	–	–
	2	425.62	98.32	0.0606	0.1115
	4	356.44	85.57	0.3812	0.3017
	7	288.91	64.60	1.3609	0.7144
	10	245.09	54.02	2.6783	1.2991
	25	154.93	32.84	14.7959	5.1667
	50	107.06	23.30	53.8820	17.4123
	100	74.00	15.33	191.7927	58.3752
Conscientious/ 20% Random	100	61.19	10.75	42.6930	1.3199
Superconsc./ 20% Random	100	56.61	8.86	79.2550	7.5048