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

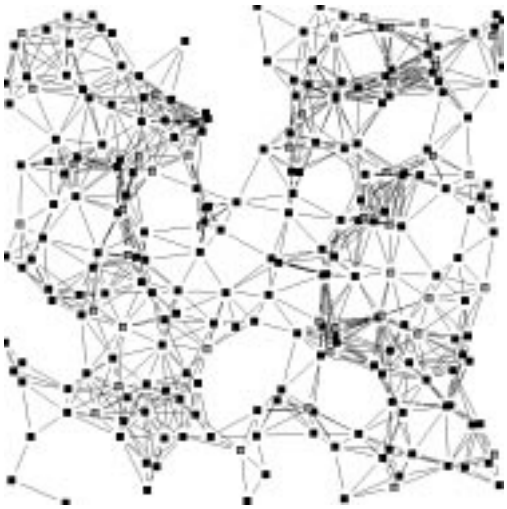


Fig. 1. Network used for all experiments

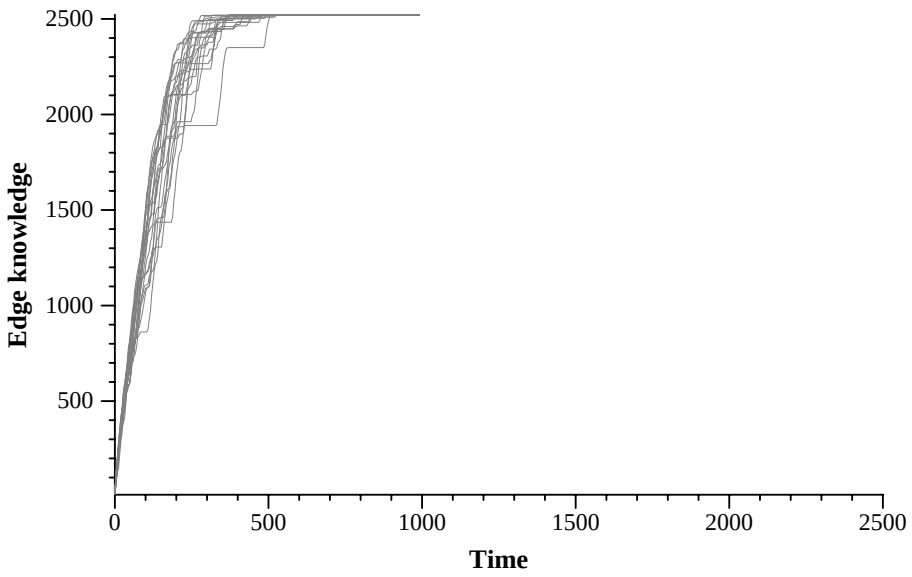
appropriate measure for this inherently distributed system. The time at which all of the agents have acquired perfect knowledge is called the "finishing time". We examine in detail the finishing times and the accumulation of knowledge about the map over time for each of the three kinds of agents.

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.


2500
2000
1500
1000
500
Edge knowledge
0
500
1000
1500
2000
2500
Time