GA-Based Parameter Tuning for Multi-Agent Systems

Joseph Haas
Drexel University
Dept. of Computer Science
Philadelphia PA 19104
jjh49@drexel.edu

Maxim Peysakhov
Drexel University
Dept. of Computer Science
Philadelphia PA 19104
maxim@drexel.edu

Spiros Mancoridis
Drexel University
Dept. of Computer Science
Philadelphia PA 19104
mancors@drexel.edu

Categories and Subject Descriptors: D.2.9
General Terms: Management, performance, algorithms.
Keywords: Software agents, GA optimization.

1. PROBLEM FORMULATION

Motivation. A MANET is a challenging environment for software system designers due to its dynamism and unpredictable nature. Network links can go up and down depending on a variety of physical factors, such as movement of hosts, terrain, weather, interference, or available battery power. Agent based systems, with their runtime flexibility, can adapt to such environments better than centralized systems [4]. On the other hand, the tuning and control of the agent based system is more complicated, due to the flexible and decentralized nature of Multi-Agent Systems (MAS). Since it is unlikely that the optimal agent population composition can be derived theoretically, a search based technique should be used to find acceptable suboptimal solutions rather then guaranteed optimal ones. Many references to the example of information dissemination or collection by agent based systems will be used throughout this paper. Envision a distributed system of mobile wireless devices used by first responders rushing to the site of a natural disaster, a police unit in a crowd control situation, or military urban operation unit where information about the individual units should be collected and disseminated to a restricted subset of the team members [1]. Such information can include, but is not limited to, the status of the networked device (i.e., remaining battery life, CPU load and memory usage), sensory information or even biometrics of the users if the device is equipped with appropriate sensors. If units have GPS capabilities, then the problem of disseminating that information becomes a "blue force tracking problem" i.e., problem of tracking friendly forces location on the battlefield. Clearly optimal performance of such systems is of critical importance to their users.

Proposed Approach. In order to address this problem of on-line software configuration, a GA based search with evaluations, performed by an on-board network simulator, is used. The management computing device reads the current network topology and forwards the data to the Network Simulator. The GA performs the optimization of the agent population based on the fitness evaluated by the simulation runs. Once desired levels of the parameter under optimization are achieved the Agent Manager block removes the old agent population and releases the new one onto the network. The overall design of the system is shown in Figure 1(a). In order to simulate normal network traffic, several steps are taken. Each link is given an amount of available bandwidth and weight, and the all pairs shortest path (Floyd-Warshall) algorithm is used to determine the routes on the simulated network. A typical load level for each link was determined by the assumption of constant communication flow between hosts using the shortest path routes. This level defined the amount of bandwidth left available on each link. The most loaded link had 20% of its bandwidth available and the rest of the links' bandwidth usage was inversely proportional to the number of communicating host pairs.

GECCO'05, June 25–29, 2005, Washington, DC, USA.
ACM 1-59593-010-8/05/0006.

2. EMPIRICAL VALIDATION

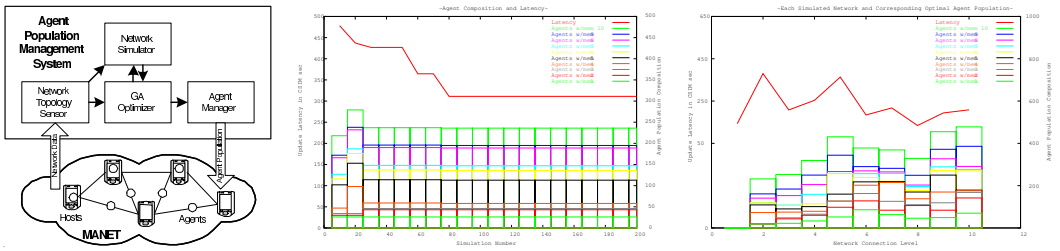
In order to validate the proposed approach, we considered the following information dissemination problem. Each host on the network is a source and a sink for a specific kind of data. For example, each host needs to keep all other hosts updated about its own current position. In order to achieve this a swarm of randomly wandering agents can be used. That update is rather scalable, robust to unexpected events, and completely decentralized [2]. Each agent can hold information about a certain number of nodes, up to as many nodes as in the network. Agents with large memory sizes take long to process and transmit. Agents with small memory sizes take less time, but sometimes cannot update the necessary number of nodes, thus an appropriate amount of each must be found.

Evaluation. In order to evaluate the process of the information dissemination by agents, we measure how up-to-date the information is on each host. Each host utilized a memory status list that contained the time of the last received update from every other host. The longest time without an update was considered to be the measure of how out-of-date that host was. The average of these numbers over the set of all hosts was considered the measure of how out-of-date the whole system was:

$$\frac{\sum_{v \in H}(\max_{u \in H}^{v}(t_{upd}))}{|H|}$$

It was usually necessary to run the simulation for a short period of time before that number would stabilize.

Simulation. In order to evaluate the effectiveness of each agent population, this project ran simulations using Lock-



Optimal Chromosomes		
Memory Size	Agent Amount	
	Initial	Final
1	25	26
2	42	17
3	45	3
4	13	12
5	48	55
6	29	23
7	11	11
8	62	42
9	0	6
10	65	41

Figure 1: System architecture (a). Best agent population over time (b). Optimal populations for each network (c). Initial and final best chromosomes (d).

heed Martin Advanced Technology Laboratories' CSIM [3]. CSIM is a discrete-event simulator for block diagram oriented systems. CSIM can be used for modeling wireless networks, computer networks, distributed systems, and other systems where the discrete event approach is applicable. The site for CSIM, where an overview of CSIM and a list of applications CSIM is used for, can be found on-line.

GA Setup. A population of twenty chromosomes was used for each experiment. Each agent population was encoded as a chromosome with an allele corresponding to agent memory size, while the value of the gene was the number of such agents in the population. The initial population was created randomly and a uniform crossover was used between simulations, with each gene having a one percent chance of mutation. When a gene was mutated, the value was increased or decreased by one agent or ten percent. Simulations were run to evaluate the agent populations. Each simulation started by triggering a thread, that lasted the entirety of the simulation, for every agent in a given population. All agents performed a random walk and also had a hash of host data updates of the size defined by the agent type. Every data update was time-stamped at the moment the agent collected the data from the host. An agent's processing and transmission times were determined by the agent's size, as well as the sizes and number of other agents currently on the same host or link. While processing an agent, a host recorded information about other nodes only if the provided information was more up-to-date than its own. After being processed, an agent was sent to its next randomly chosen destination. When an agent was in the process of being sent across a link, the amount of available bandwidth on the link decreased in accordance with the agent's size. If an agent was queued due to a lack of available bandwidth, it attempted retransmission every two CSIM seconds until it was sent.

System Behavior Over Time. In the first experiment, a randomly generated topology was used. This was done by adding one node at a time assigning a random number of neighbors to the added node ranging from one to the current number of other nodes on the network at that time. Many chromosomes in the initial populations received the default worst fitness possible due to the fact that they failed to update some network hosts. With each generation, the number of good chromosomes increased until they started converging to a particular composition. Each experiment was stopped after 200 generations because although better chromosomes could be found, the fitness would not improve enough to make time spent simulating worthwhile.

Experiment 1 (Random Network)	
Network Size	10 nodes
Network Type	Random
Simulation Length	1500
Search Space	0-700
Initial Chromosome's Latency	750.656
Best Chromosome's Latency	311.151

Table 1: Setup and Latency

A Topology's Effect on System Behavior. An idealized spectrum of network topologies, from linear to completely connected, corresponding to people moving in single-file formation, was used as the underlying topology for the experiments. In each subsequent experiment, the connectivity of the graph increased by one until it was fully connected in the final experiment, as shown in Figure 1(c). In the first network, comprised of a single-file of nodes, the optimal agent population composition was two agents of the largest memory size permissible. The very small agent population was due to there being only one possible path between every pair of nodes. An increase in agents meant more possible queuing. Only agents with a full memory size were used because an agent of any other type would never be able to carry information about a node from one end of the network to the other end. With each increase of connectivity, the use of agents with smaller memory sizes became more and more frequent. This was attributed to the fact that these agents could be transmitted and processed faster, and could also provide information to more nodes as the number of paths between nodes increases.

3. CONCLUSIONS

In this paper we presented a search based approach to the problem of software configuration systems as it applies to MAS. We empirically validated its correctness and applicability to the domain of mobile ad hoc networks and also showed some useful correlations between the network topology and the composition of the agent population performing the data dissemination tasks.

4. REFERENCES

[1] V. Cicirello *et al.* Designing dependable agent systems for MANETs. *IEEE Intelligent Sys.*, 19(5):39–45, 2004.
[2] M. Peysakhov, V. Cicirello, W. Regli. Ecology based decentralized agent management. *Proc. of FAABS*, 2004.
[3] H. Schwetman. Csim: a c-based process-oriented simulation language. *Proc. of Winter Sim. Conf.-18*, pp. 387 – 396, 1986.
[4] E. Sultanik *et al.* Implementing secure mobile agents on an ad hoc wireless network. In *Proc. of IAAI*, 2003.