DESIGNING A NETWORK INFRASTRUCTURE FOR SURVIVABILITY OF MULTI-AGENT SYSTEMS

A. Surana
MIT,
Cambridge, MA 02139
email: surana@mit.edu

N. Gautam, S.R.T. Kumara
Penn. State University,
University Park, PA 16802
email: {ngautam, skumara}@psu.edu

M. Greaves
DARPA, 3701 Fairfax Drive
Arlington, VA 22203-1714
email: mgreaves@darpa.mil

ABSTRACT
In this paper we consider a society of agents whose interactions are known. Our objective is to solve a strategic network infrastructure design problem to determine: (i) number of nodes (usually computers or servers) and their processing speeds, (ii) set of links between nodes and their bandwidths, and (iii) assignment of agents to nodes. From a performance standpoint, on one hand all the agents can reside in a single node thereby stressing the processor, on the other hand the agents can be distributed so that there is a maximum of one agent per node thereby increasing communication cost. From a robustness standpoint since links and arcs can fail (possibly due to attacks) we would like to build a network that is least disruptive to the multi-agent system functionality. Although we do not explicitly consider tactical issues such as moving agents to different nodes upon failure, we would like to design an infrastructure that facilitates such agent migrations. We formulate and solve a mathematical program for the network infrastructure design problem by minimizing a cost function subject to satisfying quality of service (QoS) as well as robustness requirements. We test our methodology on Cougaar multi-agent societies.

KEY WORDS
network design, QoS, robustness, optimization.

1 Introduction

As the number of applications requiring distributed multi-agent systems (MAS) is continuously growing, it becomes extremely important to build a network infrastructure that can guarantee a survivable MAS architecture. By survivable we mean a system that is robust, secure as well as able to provide excellent quality of service (QoS) even when stressed. For example Brinn and Greaves [6] state that the Cougaar MAS in Ultra*Log [14], would be considered survivable if it would maintain at least x% of system capabilities and y% of system performance in the face of z% infrastructure loss and wartime loads (where x, y and z are provided by the system users).

In order to build such a survivable system, there are several decisions that need to be made at different time granularities. These can be broken down as strategic (once a year or just one time), tactical (once a week to once a day depending on how often the configuration changes), and operational (typically milliseconds to seconds, depending on the granularity of information exchange) decisions. The strategic decisions typically involve designing the network infrastructure (in terms of both number and capacity) for the MAS such as computers, servers, cables, etc. The tactical decisions include where to migrate the agents if a node fails or is cut off from the other nodes. Operational decisions include adaptive control methods for deciding which agent should process a task, the fidelity with which to process a task, etc.

There has been a lot of research related to (a) *software technology*, such as, agent architecture, communication, migration, adaptation, learning, etc., and (b) *networking*, such as, QoS provisioning, fault-tolerance, dependability, robustness, etc. However there is very little research that combines the two and studies them from a systems engineering viewpoint. In this research we address that shortcoming. We focus on the strategic problem stated in the previous paragraph of designing a network infrastructure in terms of hardware to support a given society of agents and their interactions. This is with the understanding that in order to solve the tactical and operational problems effectively, the strategic problem must favor a network design that would ease tactical and operational decisions.

We now present some of the related research with the understanding that due to space restrictions it is difficult to cite all relevant articles in the literature. Andreoli et al [2] consider a distributed software network infrastructure for agents performing search tasks (such as search engines). Optimization issues for MAS at software level such as load balancing using non-linear programming techniques is studied in Aiello et al [1] for a given hardware topology. In Hofmann et al [9], a mobile intelligent agent system is built under conditions of low bandwidth to show that it could improve the efficiency of military tactical operations and the mobile agents would outperform static agents. Multi-agent hybrid systems that combine computational hardware and a large scale software residing on it, with an application to air-traffic management is studied in Tomlin et al [13]. In Kephart et al [11], one of the emerging research areas, namely considering a distributed information system as analogous to biological ecosystem and social systems, is presented in order to study their survivability. Cancho and Sole [7] consider a complex network and show that by op-

timizing simultaneously the link density and path distance in a graph (with a fixed set of nodes), leads to a scale-free topology which is robust to random attacks.

The remainder of this paper is organized as follows. In Section 2 we describe the strategic problem in detail. Then in Section 3 we formulate the problem as a mathematical program. We discuss various methods to solve the mathematical program in Section 4. Then we describe numerical examples and results in Section 5. Finally we present our concluding remarks and directions for future work in Section 6.

2 Problem Description

We now present details of the strategic problem of designing a network infrastructure for a MAS. Distributed information systems (DIS) can be viewed as a reconfigurable network with (i) computational infrastructure forming the backbone, and (ii) agents residing on it and moving around, consuming resources and providing services under uncertain and often hostile conditions. Each agent, has access to different information and makes its own local decisions, but must work together with other agents for the achievement of a common, system-wide goal. In this research we consider a MAS and an underlying network infrastructure that can be modeled as a DIS.

One of the key inputs that go into the network design problem is the agent interaction pattern. A typical agent interaction tree is depicted in Figure 1. In the figure, the agents are nodes and if there arcs connecting two nodes, then the corresponding agents interact. The agents also specify the bandwidth requirement for their interaction. Besides the bandwidth (and interaction graph), another input to the design problem is resource requirements such as CPU and memory from the host computer or server. Although the figure suggests a hierarchical network, the model does not require that. In addition, some or all of the agents can be identical (in terms of what they can do).

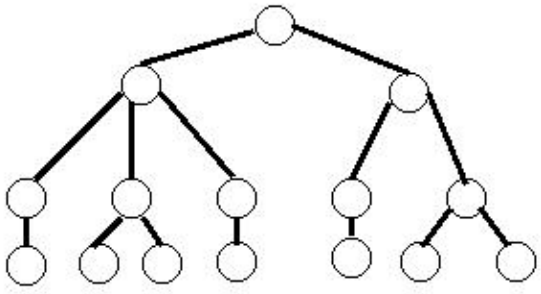

Figure 1. Example of an agent logical network

Given the inputs mentioned above and other inputs based on survivability requirements, the output of the design problem is a physical network of nodes and arcs,

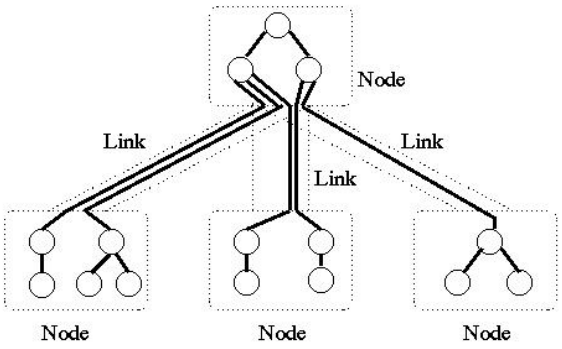

Figure 2. Agent logical network residing in a physical network

where nodes signify processors such as computers and servers, and arcs signify links (it is not required that there be a single link between 2 nodes, however, we use the capacity of the bottleneck link and pretend as though it is a single link). The two extremes of design are if we place all agents in one node and if we place one agent in each node. In case of all agents in one node, the bottleneck would be resources, i.e. whether the CPU and memory requirements of all agents can be met. However if the agents are such that there is one agent per node, there would be a lot of time spent in communication between them. We assume that if two agents are on a single node, the available bandwidth for their interaction is infinite. In Figure 2, we take the logical network of agents (described in Figure 1) and build 4 nodes and three arcs to house the agents in a physical infrastructure.

3 Robust Design: Problem Formulation

In this section we formulate a robust design problem for DIS. As we have discussed previously in Section 2, a DIS consists of two critical components: computational hardware (processors and communication links) and a MAS residing on it. With this viewpoint we have the following robust design problem: given the software agents (MAS) with their interaction pattern and computational resource requirements, we want to decide (i) How much processing power to start with i.e., how many processors and the capacity of each processor (to be selected from a given a set of processors)? (ii) How to lay out the physical network structure, i.e., how to connect the processors and what should be the capacity of each link (to be selected from a given a set of bandwidths)? (iii) How to distribute agents on this network? The above decisions are to be made so that we can meet the *survivability requirements* and at the same time minimize the *information infrastructure cost*. We translate the survivability requirements into following "specifications" for the design problem: (a) Sufficient computational resources to start with and its balanced utilization;

(b) Small average path length and diameter, measuring the connectivity; (c) Resilience to complete node and link failures. Given the above specifications, it is clear that a robust design for DIS would be one with maximum possible computational resources and a fully connected backbone network. However, this would incur a very large infrastructure cost. This leads to the problem of optimally designing the backbone network and distributing agents on it such that it is fairy robust and at the same time cost effective. In order to systematically pose this trade-off as a mathematical programming model, we first give a formal description of various entities involved in the model.

3.1 Agent Society, Nodes and Links

The MAS or the agent society is described by the computational resource each agent consumes and their interaction pattern. Let N_A be the total number of agents in the society, indexed as $\{1, 2, \cdots, N_A\}$. Let for an i^{th} agent Pa_i denote the computational resource (CPU and Memory) it consumes and let Ba_{ij} denote the bandwidth it uses, if it interacts with agent j.

A node represents a computer with a given processing power (power can refer to CPU, Memory, etc.). Each agent in the given society has to be assigned to a node. As a result each node can be assigned one or more agents. For the agents which reside on the same node, the communication requirement is automatically satisfied. Let N_{max} be the total number of nodes numbered $\{1, 2, \cdots N_{max}\}$, that is initially chosen to distribute the agents on. Let N_i denote the decision variable such that,

$$N_i = \begin{cases} 1, & \text{if node } i \text{ is selected from } N_{max} \text{ nodes} \\ 0, & \text{otherwise,} \end{cases}$$

for $1 \leq i \leq N_{max}$. Let $\mathcal{P} = \{P_1, P_2, \cdots, P_{N_p}\}$ denote the set of available processing power for nodes with an associated cost set $\mathcal{C}(P) = \{C_{p1}, C_{p2}, \cdots, C_{pN_p}\}$ and Pn_{ij} a decision variable such that

$$Pn_{ij} = \begin{cases} 1, & \text{if } i^{th} \text{ node uses processor with a power } P_j \\ 0, & \text{otherwise,} \end{cases}$$

for $1 \leq i \leq N_{max}$ and $1 \leq j \leq N_p$. Furthermore, let $\mathcal{A}_d = [A_{ij}]$ denote a matrix of the distribution of agents on the nodes, where A_{ij} is

$$A_{ij} = \begin{cases} 1, & \text{if agent } i \text{ resides on node } j \\ 0, & \text{otherwise,} \end{cases}$$

for $1 \leq i \leq N_A$ and $1 \leq j \leq N_{max}$. It is assumed that there are no multiple links and no self-loops when we connect the nodes with communication links. Let X_{ij} be the decision variable, such that

$$X_{ij} = \begin{cases} 1, & \text{if there is link from node i to j and } i \neq j \\ 0, & \text{otherwise,} \end{cases}$$

for $1 \leq i, j \leq N_{max}$. The matrix, $\mathcal{X} = [X_{ij}]$, is symmetric as the links connecting the nodes form the communication pathways and hence are undirected.

Consider the set $\mathcal{V} = \{N_i | N_i \neq 0, 1 \leq i \leq N_{max}\}$ of occupied nodes and the corresponding index set $\mathcal{I} = \{i | N_i \neq 0, 1 \leq i \leq N_{max}\}$. Let $\mathcal{E} = \{X_{ij} | X_{ij} \neq 0 \quad and \quad i, j \in \mathcal{I}, 1 \leq i, j \leq N_{max}\}$. We shall denote by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the graph with $\mathcal{V}$ as the set of vertices and $\mathcal{E}$ as a set of undirected edges. Let $l_{avg}(\mathcal{G})$ be the average path length and $D(\mathcal{G})$ be the diameter of $\mathcal{G}$. Note that if $\mathcal{G}$ consists of disconnected components, then $l_{avg}(\mathcal{G}) \longrightarrow \infty$ and $D(\mathcal{G}) \longrightarrow \infty$. Furthermore, we are only allowed to choose the capacity of links from an available set of bandwidths $\mathcal{B} = \{B_1, B_2, \cdots, B_{N_b}\}$ with an associated cost set $\mathcal{C}(B) = \{C_{b1}, C_{b2}, \cdots, C_{bN_b}\}$. Let Br_{ijl} be the decision variable which is 1 if link X_{ij} uses a capacity B_l and 0 otherwise, for, $1 \leq i, j \leq N_{max}$ and $1 \leq l \leq N_b$.

3.2 Problem Statement

With the notations given in the previous section, let $\mathcal{D} = \{N_{max}, N_i, Pn_{ij}, A_{ij}, X_{ij}, Br_{ijl}\}$ denote the set of decision variables (which are all binary). We can state the network design problem as follows:

Objective: Let $\mathcal{C}$ denote the infrastructure cost, then we desire to

$$\min \quad \mathcal{C} = \sum_{i=1}^{N_{max}} \sum_{j=1}^{N_p} C_{pj} Pn_{ij} + \sum_{i=1}^{N_{max}} \sum_{j>i}^{N_{max}} \sum_{l=1}^{N_b} C_{bl} Br_{ijl}, \tag{1}$$

subject to the following **constraints**:

1. **Resource Choice Constraints**

$$\sum_{j=1}^{N_p} Pn_{ij} = N_i, 1 \leq i \leq N_{max} \tag{2}$$

$$\sum_{l=1}^{N_b} Br_{ijl} = X_{ij}, 1 \leq i \leq N_{max} \text{ and } i < j \leq N_{max} \tag{3}$$

The above constraints (2) and (3), restricts the choice of one type of processor and one type of bandwidth capacity for a node and a link respectively.

2. **Agent Distribution Constraints**

$$\sum_{j=1}^{N_{max}} A_{ij} = 1, 1 \leq i \leq N_A \tag{4}$$

$$\sum_{i=1}^{N_A} A_{ij} Pa_i + \Delta_1(j) \leq \sum_{l=1}^{N_p} Pn_{jl} P_l, 1 \leq j \leq N_{max} \tag{5}$$

$$\sum_{l=1}^{N_A} \sum_{k=1}^{N_A} A_{li} A_{kj} (Ba_{lk} + Ba_{kl}) + \Delta_2(i, j) \leq \sum_{t=1}^{N_b} Br_{ijt} B_t \tag{6}$$

$$\text{for } 1 \leq i \leq N_{max}, i < j \leq N_{max},$$

where $\Delta_1(j) \geq 0$ and $\Delta_2(i,j) \geq 0$ are given constants, which can vary with the node and the link, respectively.

The constraints (4), force that each agent is assigned to only one node. On the other hand the constraints (5) guarantee that the agents are assigned to only those nodes which have been selected and the processing capacity chosen for that node meets the computational requirements in terms of CPU for all the agents assigned to that node. Note that this constraint also leads to a well balanced utilization of CPU by the agents, to begin with. Similarly the constraints (6), are for the meeting the communication requirements in terms of bandwidth of the links between nodes. Also each of the constraint (6), forces that if two agents which communicate with each other reside on separate nodes, then a direct communication link exists between those nodes. The constants $\Delta_1(j)$ and $\Delta_2(i,j)$, provide for additional or redundant CPU and bandwidth in the network. This redundancy takes into consideration the additional computational resources that may be required due to factors like: variability in computational resource requirements by agents, complete or partial loss resources at nodes and links and migration of agents between nodes. It should be noted that the effect of these constants can be absorbed in the processing Pa_i and bandwidth Ba_{ij} requirements of the agents and hereafter we would assume that this has been done.

3. **Connectivity Constraints**

$$X_{ij} \leq N_i \qquad 1 \leq i \leq N_{max} \quad and \quad 1 \leq j \leq N_{max} \tag{7}$$

$$l_{avg}(\mathcal{G}) \leq l_{max} \tag{8}$$

$$D(\mathcal{G}) \leq D_{max}, \tag{9}$$

where l_{max} is the maximum allowable average path length and D_{max} is the maximum allowable diameter of the network considered. The constraints (7) enforce that link exists only between nodes which have been selected. On the other hand, the constraints (8) and (9) are related to network performance and also guarantee that $\mathcal{G}$ is connected. In general, the constraints (8) and (9), cannot be expressed explicitly in terms of decision variables as equations, and have to be verified algorithmically.

4 Solution Methodology

The problem discussed in the previous section is similar in many respects, to the problems that often arise in the design of telecommunication networks [3], [4], [5]. For example, in [5], the authors consider the problem of "survivable network design" (SND), which seeks to design a minimum cost network with a given set of nodes and a set of possible edges between them, such that the connectivity requirement (which is specified as the minimum number of edge-disjoint paths needed between different nodes) is satisfied. The major distinction of our model from such formulations is that we consider infrastructural design and the distribution of agents on this network simultaneously in the strategic design phase. Note that:

- The maximum number of nodes needed satisfies, $N_{max} \leq N_A$, otherwise the optimization problem has no feasible solution, as the constraints (5) cannot be satisfied. Hence, we can always take $N_{max} = N_A$.
- Our problem, is a generalization of the "bin-packing" problem [12]. Following distinctions of our optimization problem from the "bin-packing" problem can be noted. There are two types of bins: the processors and the network links and the agents are the objects to be chosen. The capacity for both type of bins are variable and can be selected from a given set, rather than being fixed. There is constraint between filling two types of bins i.e., as we fill the processors with agents, the bin which is the link connecting the processors also gets filled based on the agent distribution. Also, there are additional constraints related to the diameter and average path length (7)-(9), that should be satisfied.
- Consider a special case of our optimization problem where Agents do not interact with each other i.e $Ba_{ij} = 0$ $(1 \leq i,j \leq N_A)$; There is only one processor with a capacity P and unit cost; There are no constraints on $l_{avg}(\mathcal{G})$ and $D(\mathcal{G})$, i.e., $D_{max} \longrightarrow \infty$ and $l_{max} \longrightarrow \infty$. Under these conditions the problem reduces to the usual bin-packing problem, as follows:

$$min \quad \mathcal{C} = \sum_{i=1}^{N_{max}} N_i, \tag{10}$$

subject to

$$\sum_{j=1}^{N_A} A_{ij} = 1, 1 \leq i \leq N_A, \tag{11}$$

$$\sum_{i=1}^{N_A} A_{ij} Pa_i \leq N_j P, 1 \leq j \leq N_A. \tag{12}$$

The bin-packing problem is known to be NP-hard in the strong sense [12] . Since our problem is a generalization of the bin-packing problem it is also NP-hard in the strong sense. Given this we either need to develop heuristics or use evolutionary algorithms to obtain solutions.

We have used Genetic Algorithms (GA) with the following important features:

- Rather than using a binary encoding, we used a scheme of integer coding of the decision variables.
- The initial pool of population is generated randomly, with one feasible solution. The feasible solution can

be obtained as follows. Start with N_A nodes, assign each agent to a separate node and choose a lowest possible processing capacity from the available set $C(P)$ such that the processing requirements for each agent is satisfied. Connect the nodes which have agents that interact with each other and assign to that link a capacity with the lowest possible bandwidth from the available set $C(B)$ such that the communication requirements are satisfied.

- We have used NSGA2 [8, 10], as the GA solver. It has capability to automatically handle constraints. It uses a mean-centric crossover and uniform bounded mutation operators for real coded strings.

5 GA Application Examples and Results

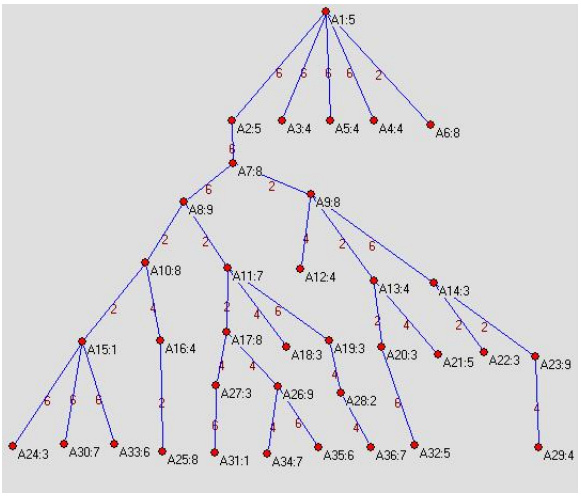

Figure 3. A military supply chain as an agent society

5.1 Inputs

Following are inputs based on notation in Section 3.2:

1. The *cost structure* for processing power and bandwidth used in examples is shown in the Table 1.

2. The *agent societies* in the examples considered were generated randomly. The processing Pa_i and the bandwidth Ba_{ij} requirements for the agents were sampled from uniform distribution. However, the structure of the agent societies in all the cases was restricted to be hierarchical. This is motivated by the fact that most of the organizations, like in Command and Control or in society have hierarchical structure. Note that the optimization problem and formulation we have considered is general enough to be applied to an agent society with any underlying structure. The agent societies, differ in number of agents (Table 2), the nature of branching in the hierarchical structure and the variation in processing and bandwidth requirements for agents. In Figure 3, each node in the tree is labeled by an agent number A_i and its processing requirement Pa_i, whereas each link between the agents A_i and A_j (if they interact), is labeled by $(Ba_{ij} + Ba_{ji})$, their communication requirement.

3. The restriction on the maximum allowable diameter D_{max} and the average path length l_{max} for the DIS are listed in Table 2.

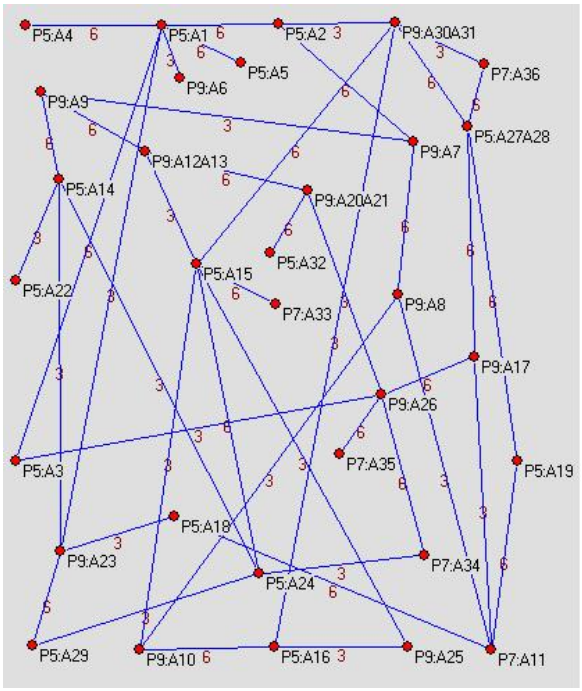

Figure 4. GA Result: DIS Layout for MSC, $\mathcal{C} = 130$

Table 1. Processing Cost and Bandwidth Cost

i	1	2	3
P_i	5	7	9
C_{pi}	1	2	3

i	1	2
B_i	3	6
Cb_i	1	2

5.2 Output: GA Results

The costs $\mathcal{C}$ obtained by running GA have been tabulated below (Table 2), while the DIS layout is shown in Figure 4, next to the corresponding agent society. In the figure, each node in the graph is labeled by the processing power P_i followed by the agents which are assigned to that node, while each link is labeled by the bandwidth B_i chosen for it. It should be noted that many agents can be assigned to a same node. For example, (Figure 4), agents $A12$ and $A13$ are both assigned to a the node labeled $P9 : A12A13$.

Table 2. GA Results

Agent Society No.	N_A	D_{max}, l_{max}	GA $\mathcal{C}$	Ratio $\frac{\mathcal{C}}{N_A}$
1	4	2, 2	15	3.75
2	7	”	27	3.86
3	10	5, 4	30	3.00
4	12	”	33	2.75
5	15	”	50	3.33
6	19	”	61	3.21
7	24	”	87	3.63
8	33	”	108	3.27
9	40	”	164	4.1
10	50	”	188	3.76

The Table 2, shows that the *optimal cost per agent ratio* ($\frac{\mathcal{C}}{N_A}$) is fairly constant, with a small variation. This may be a result of the cost structure assumed and the particular instances of the agent societies considered. This observation, however, can have following implication: given a very large agent society like with $N_A = 5,000$ agents, we can decompose it into smaller agent societies, solve the optimization problem for each of the sub-societies and then combine them to solve the overall problem. Due to the constancy of the ratio $\frac{\mathcal{C}}{N_A}$, this heuristic should lead to solutions which are fairly close to optimal.

As a final example we consider one of the realistic agent societies which has been developed in the Ultra*Log Program [14], [6]. The society is shown below in Figure 3 and represents a typical military supply chain (MSC). The structure of society is an exact replica of the true society, however the processing and band width requirements for the agents have been assigned randomly. The result obtained by GAs has been shown in the Figure 4.

6 Conclusion and Future Work

In this paper we have systematically studied the issue of survivability of DIS. Based on these we formulated a robust design problem for DIS. We showed that this problem is NP hard in strong sense and used GAs to obtain solutions for a number of example agent societies. We also considered a realistic agent society representing a military supply chain. We showed that our robust design problem formulation results in a fairly survivable DIS.

Survivability of DIS is an emerging area and future research is possible in a number of varied directions. Refining the robust design problem we have posed and developing heuristic solution methodologies for it. Further research is required to build mechanisms for survivability against other types of attacks, such as security and DOS attacks. Most of the above stated problems are nothing new for biological systems which have routinely solved them for literally millions of years. Can we draw inspiration from the structures discovered in biology to solve problems of distributed systems? We believe that the quest for “open-ended” survivability for DIS can be achieved only by exporting biological mechanisms into software systems.

Acknowledgements

The authors acknowledge DARPA (Grant#: MDA972-1-1-0038 under Ultra*Log Program) and NSF (Grant#:ANI-0219747 under ITR program) for their generous support for this research. Special thanks to the anonymous reviewers for their comments and suggestions.

References

[1] W. Aiello, B. Awerbuch, B. M. Maggs, and S. Rao. Approximate load balancing on dynamic and asynchronous networks. In *ACM Symposium on Theory of Computing*, pages 632–641, 1993.

[2] J. Andreoli, U. Borghoff, R. Pareschi, S. Bistarelli, U. Montatnari, and F. Rossi. Constraints and agents for a decentralized network infrastructure. In *AAAI Workshop, Menlo Park, California, USA: AAAI Press.*, pages 39–44, 1997.

[3] A. Balakrishnan and K. Altinkemer. Using hop-constrined model to generate alternative= communication network design. *ORSA Journal on Computing*, 4(2), 1992.

[4] A. Balakrishnan, T. l. Magnanti, and P. Mirchandani. A dual-based algorithm for multi-level network design. *Management Science*, 40(5):567–581, 1994.

[5] A. Balakrishnan, T. l. Magnanti, and P. Mirchandani. Connectivity-splitting models for survivable network design. *Submitted*, 2003.

[6] M. Brinn and M. Greaves. Leveraging agent properties to assure survivability of distributed multi-agent systems. In *https://docs.ultralog.net/dscgi/ds.py/Get/File-4088/AA03-SurvivabilityOfDMAS.pdf*, 2002.

[7] R. F. Cancho and R. V. Sole. Optimization in complex networks. In *http://arxiv.org/PS_cache/cond-mat/pdf/0111/0111222.pdf*, 2001.

[8] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multi-objective genetic algorithm-nsga-ii. In *http://www.iitk.ac.in/kangal/pub.htm*, 2000.

[9] M. O. Hofmann, A. McGovern, and K. R. Whitebread. Mobile agents on the digital battlefield. In *Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98)*, pages 219–225, 1998.

[10] KanGAL. Kanpur genetic algorithm laboratory. In *http://www.iitk.ac.in/kangal/pub.htm*.

[11] J. O. Kephart, T. Hogg, and B. A. Huberman. Collective behavior of predictive agents. *Physics D*, 42:48–65, 1990.

[12] A. Martello and P. Toth. *Knapsack Problems Algorithms and Computer Implementations*. John Wiley and Sons, 1990.

[13] C. Tomlin, G. Pappas, and S. Sastry. Conflict resolution for air traffic management: A case study in multi-agent hybrid systems. *IEEE Trans. on Automat. Ctrl*, 43(4), 1998.

[14] ULTRALOG. A darpa program on logistics infromation system= survivability. In *http://www.ultralog.net/*.