Multisensor Data Fusion in Distributed Sensor Networks Using Mobile Agents *

Hairong Qi, Xiaoling Wang
ECE, 319 Ferris Hall
University of Tennessee
Knoxville, TN 37996
hqi,xwang1@utk.edu

S. Sitharama Iyengar
CS, 298 Coates Hall
Louisiana State University
Baton Rouge, LA 70803
iyengar@bit.csc.lsu.edu

Krishnendu Chakrabarty
ECE, 130 Hudson Hall
Duke University
Durham, NC 27708
krish@ee.duke.edu

Abstract – *We describe the deployment of mobile agent in Distributed Sensor Networks (DSNs) to form an improved infrastructure for multisensor data fusion. Compared with the traditional client/server paradigm, mobile agent adopts a new computing model: data stay at the local site, while the execution code is moved to the data sites. Mobile-agent-based DSN (MADSN) saves the network bandwidth and provides an effective way to overcome network latency. It is used to integrate preprocessed data located at local sensor nodes. Progressive accuracy can be achieve during the agent migration. We take target classification as an example to show how MADSN supports high performance distributed integration.*

Keywords: distributed sensor networks, mobile agent, data fusion, target classification

1 Introduction

Multisensor data fusion is an evolving technology, concerning the problem of how to fuse data from multiple sensors in order to make a more accurate estimation of the environment [8, 10, 16]. Applications of data fusion cross a wide spectrum, including environment monitoring, automatic target detection and tracking, battlefield surveillance, remote sensing, global awareness, etc. They are usually time-critical, cover a large geographical area, and require reliable delivery of accurate information for their completion.

So far, client/server computing model has been most popularly used in Distributed Sensor Networks (DSNs) to handle multisensor data fusion. However, as advances in sensor technology and computer networking allow the deployment of large amount of smaller and cheaper sensors, huge volumns of data need to be processed in real-time. In this paper, we explore the usage of mobile agent in DSNs for data fusion tasks as an alternative to the traditional client/server model.

The outline of this paper is as follows: Sec. 2 describes the mobile agent computing model and its advantages over client/server model. Sec. 3 presents the design and implementation of mobile-agent-based DSN (MADSN). Sec. 4 discusses different fusion algorithms including both centralized and distributed design. Sec. 5 uses target classification as an example to show the effectiveness of MADSN in distributed sensor fusion.

2 Mobile agent computing model

Generally speaking, mobile agent is a special kind of software which can execute autonomously. Once dispatched, it can migrate from node to node performing data processing autonomously. Lange listed seven good reasons to use mobile agents [12], including reducing network load, overcoming network latency, robust and fault-tolerant performance, etc.

In the mobile agent computing model, data stay at the local site, while the processing task is moved to the data sites. By transmitting the computation engine instead of data, the mobile agent model offers several important benefits: 1) Network bandwidth requirement is reduced. Instead of passing large amounts of raw data over the network through several round trips, only the agent of small size is sent. This is especially important for real-time applications and where the communication is through low-bandwidth wireless connections; 2) Better network scalability can be achieved. The performance of the network is not affected when the number of sensor is increased. Agent architecture can support adaptive network load balancing automatically; 3) Extensibility is supported. Mobile agents can be programmed to carry task-adaptive processes which extends the capability of the system; and 4) Stability. Mobile agents can be sent when the network connection is alive and return results when the connection is re-established. Therefore, the performance of the

*This research was supported in part by DARPA under grant N66001-001-8946

system is not much affected by the reliability of the network.

Although the role of mobile agents in distributed computing is still being debated mainly because of the security concern [5, 14], several applications have shown clear evidence of benefitting from the use of mobile agents. For example, mobile agents are used in networked electronic trading [4] where they are dispatched by the buyer to the various suppliers to negotiate orders and deliveries, and then return to the buyer with their best deals for approval. Instead of having the buyer contact the suppliers, the mobile agents behave like representatives, interacting with other representatives on the buyer's behalf, and alert the buyer when something happens in the network that is important to the buyer. Another successful example of using mobile agents is distributed information retrieval and information dissemination [6, 9, 15, 22]. Agents are dispatched to heterogeneous and geographically distributed databases to retrieve information and return the query results to the end-users. Mobile agents are also used to realize network awareness [2] and global awareness [19]. Network-robust applications are of great interest in military situations today. Mobile agents are used to be aware of and reactive to the continuously changing network conditions to guarantee successful performance of the application tasks.

In this paper, we deploy the mobile agent paradigm to improve the design of DSN. We use the acronym MADSN to denote mobile-agent-based DSN. Figure 1 provides a comparison between DSN and MADSN from architecture points of view.

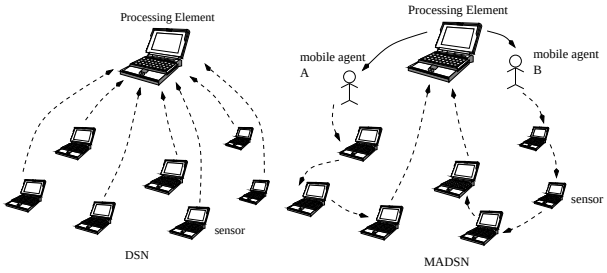

Figure 1: Comparison of architectures between DSN and MADSN

3 MADSN design and implementation

We define mobile agent as an entity of four attributes: *identification*, *itinerary*, *data space*, and *method*, as shown in Fig. 2.

- *Identification:* A number in the format of 2-tuple (i, j), where i indicates the IP address of the dispatcher and j the serial number assigned to agents by the dispatcher. Each mobile agent can be uniquely identified by its identification.

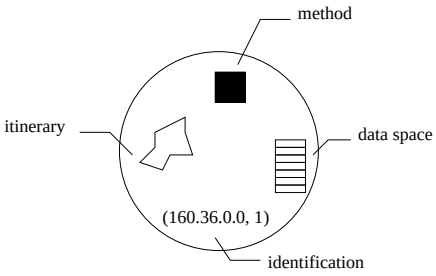

Figure 2: Mobile agent as an entity of four attributes.

- *Data space:* The agent's data buffer which carries the partially integrated results. This result should provide progressive accuracy as the agent migrates from node to node.

- *Itinerary:* The route of migration. It can be fixed or dynamically determined based on the current network status. Currently, we assume the itinerary is fixed.

- *Method:* The processing task (or execution code) carried with the agent. Our focus in this paper is the development of distributed sensor integration algorithms.

MADSN is simulated using a mobile agent framework (MAF) we developed. Its architecture design is shown in Fig. 3. MAF is built upon TCP/IP protocol. A combination of C/C++ and an interpreting language as the front end is usually the setup of the language support. The mobile agent server functions like a daemon. It listens to the network for any valid incoming agents and invokes the corresponding routine to carry out sensor integration. Again, we can see that the local data reside at the local site, only the integrated results are carried with the mobile agent.

There have been many mobile agent systems developed recently. Most of them use Java or the combination of C/C++ and a scripting language, such as IBM's Aglets [1], Dartmouth's Agent Tcl [20], General Magic's Telescript [21], etc. MAF is implemented in Python and has left flexible interface to other processing modules. [13] provides a detailed list of benefits of Python, including its purely object-oriented support, its suitability for rapid prototyping, and its object serialization support which is one way to save information between program executions. It needs to be clarified that object serialization only keeps the data space, but not the execution status. No modification to the Python interpreter is needed in order to support *moderate mobility*.

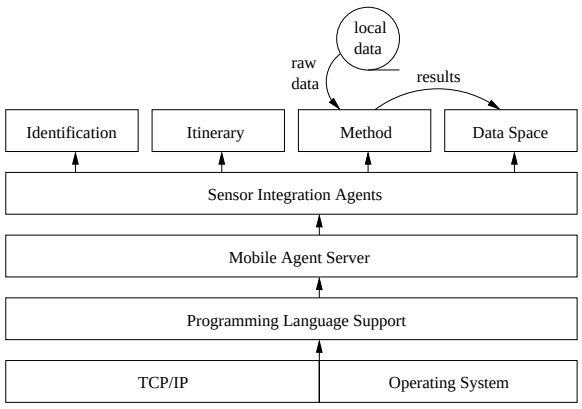

Figure 3: MAF design architecture.

In order to enhance the performance of MAF, and also make usage of existing mature integration modules, MAF has provided flexible interface where the integration modules developed in C/C++ can be dynamically linked and executed by agents.

4 Distributed sensor integration algorithm design

As larger amount of sensors are deployed in harsher environment, it is important that sensor integration techniques are robust and fault-tolerant so that they can handle uncertainty and faulty sensor readouts. Here, the redundancy in the sensor readouts are used to provide error tolerance. In this section, we first describe an efficient multi-resolution integration (MRI) algorithm. Then we modify the algorithm such that the original centralized integration can be carried out distributively. Readers are referred to [18] for detailed derivation and case study.

4.1 Centralized MRI algorithm

The original MRI algorithm was proposed by Prasad, Iyengar and Rao in 1994 [17]. The idea essentially consists of constructing a simple function (the overlap function) from the outputs of the sensors in a cluster and resolving this function at various successively finer scales of resolution to isolate the region over which the correct sensors lie. Each sensor in a cluster measures the same parameters. It is possible that some of them are faulty. Hence it is desirable to make use of this redundancy of the readings in the cluster to obtain a correct estimate of the parameters being observed.

Figure 4 illustrates the overlap function ($\Omega(x)$) for a set of 7 sensors calculated from their characteristic functions. The actual value of the parameter being measured lies within regions over which the maximal peaks of $\Omega(x)$ occur.

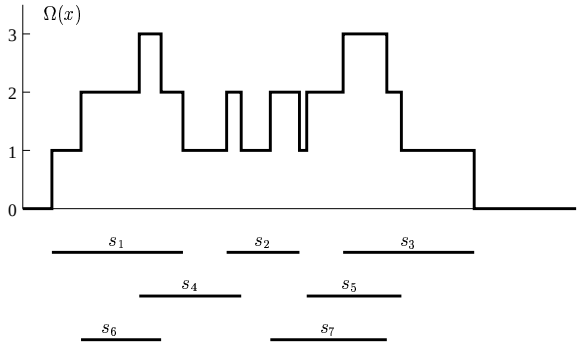

Figure 4: The overlap function for a set of 7 sensors.

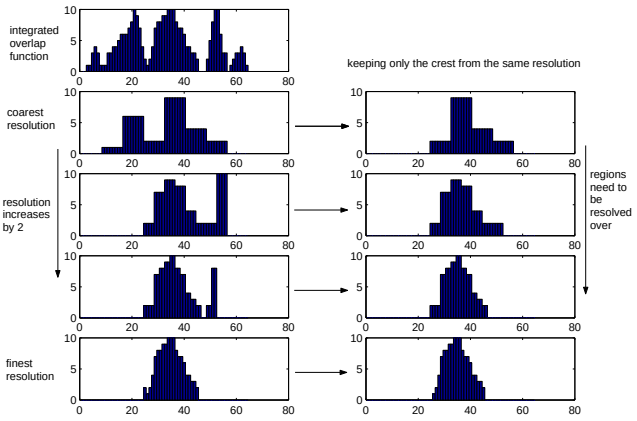

Figure 5: An overlap function $\Omega(x)$ and its appearance at different resolutions (The shaded region indicates the region needs to be resolved over).

Multi-resolution analysis provides a hierarchical framework for interpreting the overlap function. It is natural and more efficient to first analyze details at a coarse resolution and then increase the resolution for only the region of interest. Given a sequence of increasing resolutions, at each resolution, MRI picks the crest from the overlap function, and resolve only the crest in the next finer resolution level. *Crest* is a region in the overlap function with the highest peak and the widest spread. The algorithm is optimal, since the overall time required is $O(n \log n)$, which is the time required to maintain $\Omega(x)$. This algorithm is also robust, satisfies a Lipschitz condition [11], which ensures that minor changes in the input intervals cause only minor changes in the integrated result. Figure 5 illustrates the multi-resolution analysis procedure.

Cho et. al. [3] improves the Ω function to only return the interval with the overlap function ranges $[n - f, n]$. It also satisfies Lipschitz condition. The biggest advantage of this function is that it is able to reduce the width of the output interval in most cases and produce a narrower output interval when the number of sensors involved is

large, which is the case for distributed sensor network in general.

4.2 Decentralized MRI algorithm

In a distributed sensor network (DSN), all readouts from the sensor nodes are sent to their corresponding processing elements, where the overlap function at the *finest* resolution is first generated, and the multi-resolution analysis procedure is then applied to find the crest at the *desired* resolution.

In a mobile-agent-based DSN (MADSN), the mobile agents migrate among the sensor nodes and collect readouts. Therefore, each mobile agent always carries a *partially* integrated overlap function which is accumulated into a final version at the processing element after all the mobile agents return. During this process, if MADSN applies MRI in the same way as DSN does, that is, letting mobile agents carry the partially integrated overlap function in its finest resolution and then use MRI to find the crest at desired resolution at the processing center, the advantages of mobile agents will be nullified because of heavy data migration.

We modified the basic MRI algorithm and presented a more efficient implementation for MADSN [18]. The key concept underlying the modified algorithm is that MRI is applied *before* accumulating the overlap function. A 1-D array, ω_x, can serve as an appropriate data structure to represent the partially-integrated overlap function carried by a mobile agent. The size of the array is dependent upon the resolution requirement. The coarser the resolution, the smaller the data buffer. The modified algorithm also provides *progressive accuracy*. When the accuracy requirement has been reached, the mobile agent can return to the processing center immediately without finishing the scheduled route. The MADSN implementation of MRI achieves the same integration result as original MRI but is more flexible, and is able to carry out the integration distributively. We use target classification as an example to show the advantages of MADSN.

5 MADSN for target classification

Target detection, classification, localization, and target tracking are all typical applications in DSNs. By integrating observations from different sensors at different locations, the system will automatically send alarms whenever a certain target is detected. Our discussion in this section concentrates on ground vehicles classification using MADSN formed by unattended ground sensors. Both seismic and acoustic sensors are typically used in battlefield surveillance and our example uses only acoustic signals.

Acoustic signal is strongly non-stationary. It can be interfered by many factors such as the speed of the target, Doppler effects, noise from various moving parts and frictions, environmental effects, etc. On top of that, the deployed sensor can be faulty or even dead. Therefore, sensor integration is crucial here in order to tolerate the faults, uncertainties in the sensor readouts and obtain accurate classification results.

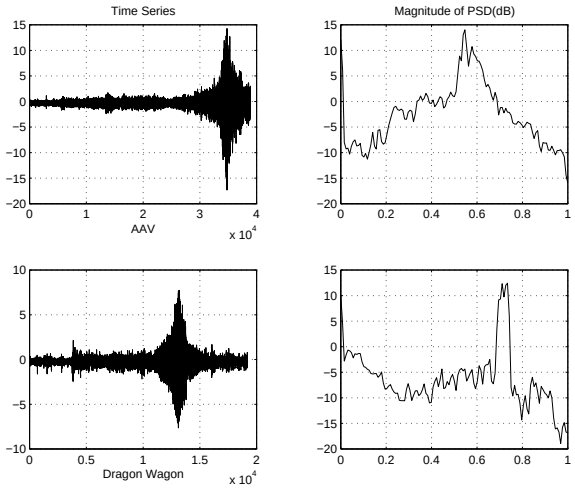

Figure 6: Time series and PSD of acoustic signals from AAV and DW.

5.1 Parameter selection

Before we can use MRI algorithm for distributed integration, we must first identify the parameters to be observed. Different parameters can be chosen dependent upon different signal processing techniques applied. Acoustic signals can be analyzed in time domain, frequency domain, or time-frequency domain. Here we choose to analyze the signal under frequency domain. By observing the power spectral density (PSD) of the time series acoustic signal, we found that the *dominant frequency range* in the PSD has a good indication of different vehicles. Figure 6 shows a segment of time series acoustic signal from AAV and Dragon Wagon (DW) with the corresponding PSD as well. We can see that the location of the peak and the width of the peak differ from vehicle to vehicle. Here, we define the dominant frequency range as the range between the closest inflection points at the left and right sides of the maximum. It is, however, difficult to find the maximum from the original PSD. Therefore, we first use Bézier spline [7] to smooth the original PSD into an $(n-1)th$ polynomial, where n is the number of samples used to represent the original PSD. The more samples we use, the closer the polynomial to the original PSD, the longer the computation time. Once we obtained the smoothed PSD, we can easily pick the maximum and the dominant frequency range as defined above. MRI can then be applied to integrate the dominant frequency range as the mobile agent migrates from node to node. A block

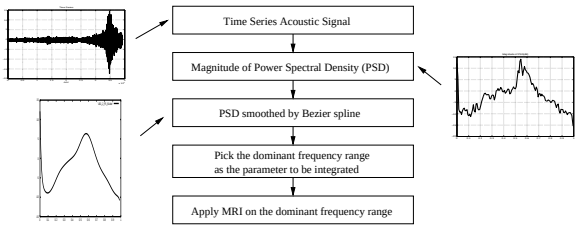

Figure 7: Block diagram on using MRI to integrate the dominant frequency range obtained from PSD.

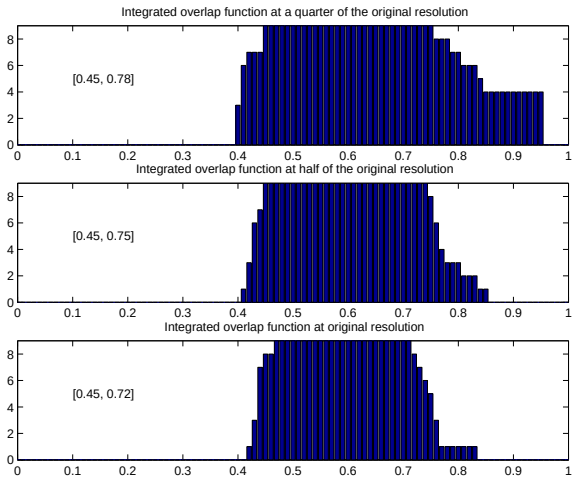

Figure 8: Integration result at different resolutions from nine sensors with one of them being faulty. (The target is AAV)

diagram of the procedure is illustrated in Fig. 7.

5.2 Integration results using MRI

We conducted two experiments to show the results of using MRI in MADSN. In the first experiment, we choose the readouts from nine sensor nodes with one of them being faulty. The target in this experiment is AAV. In the second experiment, we also choose the readouts from nine sensor nodes but with two of them being faulty. Dragon Wagon (DW) is the target in the second experiment.

Figure 8 is the result of mobile agent implementation of MRI at different resolutions for AAV. From the plots, we can see the effect of multi-resolution analysis: the finer the resolution, the narrower the dominant frequency range, the more accurate the estimation. For AAV, the dominant frequency range at the finest resolution is $[0.45, 0.72]$ with the location of peak at around 0.58. Figure 9 shows another example with DW as the target. The dominant frequency range at the finest resolution is $[0.61, 0.65]$ with the location of peak at around 0.63.

Figure 10 shows the effect of progressive accuracy achieved in the first experiment. The overlap functions

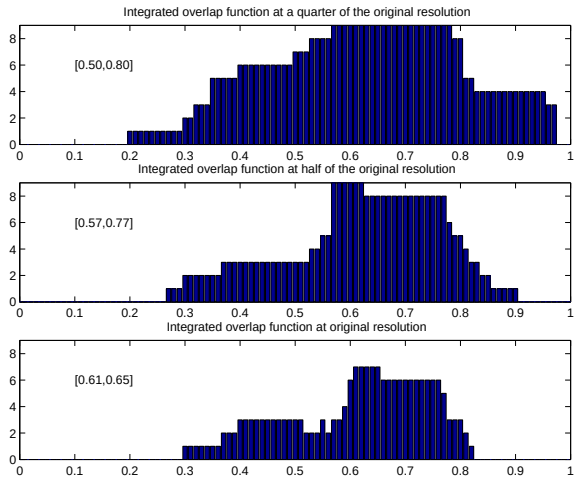

Figure 9: integration results at different resolutions from nine sensors with two of them being faulty. (The target is DW)

are all at their finest resolution. From the eight partially integrated results, we can see that after having migrated around five sensors, the integration result goes stable. The observations from the remaining four sensors do not affect the result at all. Therefore, the mobile agent can return to its dispatcher without finishing up its scheduled itinerary, but still achieve required accuracy.

6 Summary

This paper describes the design and implementation of mobile-agent-based distributed sensor networks. Mobile agent has been a promising solution to high performance distributed computing. Here, we use mobile agent to carry out distributed sensor integration tasks. We take an example of target classification to illustrate the efficiency of mobile agent computing model.

References

[1] Aglets. http://www.trl.ibm.co.jp/aglets.

[2] W. Caripe, G. Cybenko, K. Moizumi, and R. Gray. Network awareness and mobile agent systems. *IEEE Communications Magazine*, pages 44–49, July 1998.

[3] E. Cho, S. S. Iyengar, K. Chakrabarty, and H. Qi. A new fault tolerant sensor integration function satisfying local lipschitz condition. *Submitted for publication*, 2000.

[4] P. Dasgupta, N. Narasimhan, L. E. Moser, and P. M. Melliar-Smith. Magnet: mobile agents for networked electronic trading. *IEEE Transactions on*

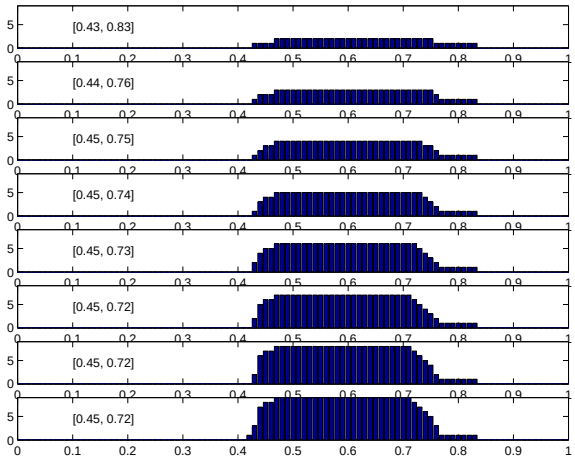

Figure 10: Progressive accuracy achieved by MADSN. (All the overlap functions are at their finest resolution. The target is AAV)

Knowledge and Data Engineering, 11(4):509–525, July/August 1999.

[5] C. G. Harrison, D. M. Chess, and A. Kershenbaum. Mobile agents: are they a good idea? Technical Report RC 19887, IBM Thomas J. Watson Research Center, March 1995. http://www.research.ibm.com/massive/mobag.ps.

[6] M. Hattori, N. Kase, A. Ohsuga, and S. Honiden. Agentb-based driver's information assistance system. *New Generation Computing*, 17(4):359–367, 1999.

[7] D. Hearn and M.P. Baker. *Computer Graphics, C Version*. Prentice Hall, Upper Saddle River, New Jersey, 2nd edition, 1997.

[8] D. N. Jayasimha, S. S. Iyengar, and R. L. Kashyap. Information integration and synchronization in distributed sensor networks. *IEEE Trans. Syst., Man, Cybern.*, SMC-21(21):1032–1043, Sept./Oct. 1991.

[9] J. Kay, J. Etzl, G. Rao, and J. Thies. Atl postmaster: a system for agent collaboration and information dissemination. In *Proceedings of the 2nd International Conference on Autonomous Agents*, pages 338–342, Minneapolis, MN, 1998. ACM.

[10] A. Knoll and J. Meinkoehn. Data fusion using large multi-agent networks: an analysis of network structure and performance. In *Proceedings of the International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 113–120, Las Vegas, NV, Oct. 2-5 1994. IEEE.

[11] L. Lamport. Synchronizing time servers. Technical Report Technical Report 18, Digital System Research Center, 1987.

[12] D. B. Lange and M. Oshima. Seven good reasons for mobile agents. *Communications of the ACM*, 42(3):88–89, March 1999.

[13] M. Lutz and D. Ascher. *Learning Python*. O'Reilly, April 1999.

[14] D. Milojicic. Mobile agent applications. *IEEE Concurrency*, pages 80–90, July-September 1999.

[15] T. Oates, M. V. N. Prasad, and V. R. Lesser. Cooperative information-gathering: a distributed problem-solving approach. *IEE Proceedings - Software Engineering*, 144(1):72–88, February 1997.

[16] L. Prasad, S. S. Iyengar, R. L. Kashyap, and R. N. Madan. Functional characterization of sensor integration in distributed sensor networks. *IEEE Trans. Syst., Man, Cybern.*, SMC-21, Sept./Oct. 1991.

[17] L. Prasad, S. S. Iyengar, and R. L. Rao. Fault-tolerant sensor integration using multiresolution decomposition. *Physical Review E*, 49(4):3452–3461, April 1994.

[18] H. Qi, S. S. Iyengar, and K. Chakrabarty. Multiresolution data integration using mobile agents in distributed sensor networks. *Accepted by IEEE Transactions on SMC: C*, 2000.

[19] K. N. Ross, R. D. Chaney, G. V. Cybenko, D. J. Burroughs, and A. S. Willsky. Mobile agents in adaptive hierarchical bayesian networks for global awareness. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pages 2207–2212. IEEE, 1998.

[20] Agent TCL. http://agent.cs.dartmouth.edu/.

[21] Telescript. http://www.generalmagic.com.

[22] J. S. Wong and A. R. Mikler. Intelligent mobile agents in large distributed autonomous cooperative systems. *Journal of Systems and Software*, 47(2):75–87, 1999.