Cooperative Large Scale Mobile Agent Systems

Osher Yadgar[1], Charles L. Ortiz, Jr.[3], Sarit Kraus[1,2], Evyatar Sharabi[1]

[1] Dept. of Computer Science,
Bar Ilan University,
Ramat Gan, 52900 Israel
{yadgar,sarit} @macs.biu.ac.il

[2] Dept. of Computer Science,
Bar Ilan University,
Ramat Gan, 52900 Israel
sarit@macs.biu.ac.il

[3] Artificial Intelligence Center,
SRI International,
Menlo Park, CA 84025 USA
ortiz@ai.sri.com

***Abstract*—We consider environments that consist of a large set of relatively cheap and simple agents and a large set of objects. The agents should track the objects and identify how the objects' states change over time. Since the agents are simple, it is very difficult for an individual agent to track an object by itself. Thus, the agents must cooperate and exchange information in order to succeed in their tasks. The problem becomes even more difficult when agents can collect only partial information about nearby tasks obtained by possibly noisy sensors.**

We propose the Distributed Dispatcher Manager (DDM) system that organizes teams hierarchically to solve such large-scale problems. We show that using large numbers of simple agents may be more beneficial than small numbers of sophisticated ones.

DDM addresses three important issues: (i) how the agents should be distributed over an area, (ii) how agents should process local information, derived from possible noisy sensors, to provide a partial solution to nearby tasks, and (iii) how partial solutions should be integrated into a global solution. DDM's contributions include: (1) realtime processes for combining partial results to form an accurate global solution, (2) increased system fault tolerance, and (3) scalability for very large task and agent problem domains•.

***Index Terms*— Large-scale agent systems, scalability and complexity issues.**

I. Introduction

AGENT collaboration for performing tasks in very large-scale multi-agent environments is an important and difficult problem. The problem becomes even more challenging if local noisy data collected by several agents must first be combined as a prerequisite for task execution. A further complication arises if tasks must be performed in real-time. In this paper we focus on environments that consist of a large set of relatively cheap and simple agents and a large set of objects. The agents are tasked with tracking objects and in real time identifying how the objects' states change over time. Since the agents are simple, it is very difficult for one agent to track an object by itself. Thus, the agents must cooperate and exchange information in order to succeed. A typical domain that we have in mind is one involving sensor webs that must be jointly tasked for surveillance: sensors correspond to agents and targets that appear in the environment correspond to objects. The agents should track the targets and identify their paths through the environment in real time. Such a problem is very common: airports rely on sets of radars to track aircraft, a military defense system may use sensors to track movements and urban traffic controls may study drivers' behaviors.

The issues that need to be addressed in order to solve such problems involving object-tracking in large-scale environments can be characterized as follows:

- **Individual agents:** (i) how should an agent act in order to collect data? (ii) how can an agent extend local, partial information to arrive at a better local assessment of the states of objects?
- **Large-scale multi-agent interaction:** (i) how should the agents be organized to cooperate given that there is a very large number of agents and tasks? (ii) how can local assessments from teams of many agents be used to form a global assessment of the objects' states, given the large environment and the fast respond need?

In [13,14] Yadgar *et al* presented the DDM framework to solve each of these problems. DDM organizes teams hierarchically. Each agent can act autonomously within its environment while processing local data. In DDM agents are grouped into teams, each with a distinguished team leader; teams might be assigned to specific geographic sectors of interest. Teams are themselves grouped into larger teams. Communication is restricted to flow only between an agent (or team) and its team leader. Each team leader is provided with a novel algorithm to integrate information obtained from its team members. Yadgar *et al* presented preliminary experiments that demonstrated the advantage of the DDM model in environments with dozens of agents and objects and the advantage of mobile agents over static ones. These experiments showed the ability of the DDM algorithms to successfully address, in real-time, the problems of local processing and information integration mentioned above. In this paper, we present results from experiments that involved one thousand agents and more than a thousand objects. These results support our hypothesis that DDM is successful in large-scale environments.

Our results show that as the number of levels of the hierarchy increases the quality of the results slightly decrease. However, the time complexity of the system decreases

• This work was supported by DARPA contract F30602-99-C-0169 and NSF grant IIS9820657. The second author is also affiliated with UMIACS.

exponentially. Consequently, we found that using too few levels may not suffice to solve the global problem. We studied the problem of balancing hierarchy height with the desired accuracy of tracking and discuss this issue in this paper.

Previous systems that were developed to solve similar problems were tested only in small-scale environments (i.e., tens of targets and agents). It appears that there is no easy way to apply these systems for solving problems involving thousand of tasks because they require close cooperation and coordination between the agents. For example, in [11,12,6] three agents must take measurements simultaneously in order to identify a target. Such cooperation is achieved in various sophisticated ways. For example, in [12] partial global planning is used while in [5] auction mechanisms are used and in [11] case-based reasoning is used. The ability of DDM to support large-scale environments is primarily due to the following properties. The large-scale problem is hierarchically divided into small segments to be partially solved by agents. Agents in DDM act autonomously and do not need to coordinate their activities with other agents; they need only interact with their team leaders, exchanging, from time to time, the partial solutions they have so far constructed. The partial solutions correspond to that information which the agent was able to process together with any other data that it was not able to process. Thus, the amount of unprocessed data that each agent needs to process does not increase as one move up in the hierarchy. In addition, during such interactions, there is no need for synchronization. Concurrently processing information by a vast number of agents in such large-scale environments leads to a fast solution; an alternative consisting of a purely centralized architecture may lack sufficient computation resources.

Others have also investigated the application of hierarchical models to multi-agent cooperation (e.g., [4]). However, those applications have not considered restricting the source of an agent's information to strictly local sources, nor the effects of such a restriction on the accuracy of a system's results. Corkill and Lesser [2] investigated various team organizations in terms of *interest areas* which partitioned problem solving nodes according to roles and communication, but they were not initially hierarchically organized [3,7].

Another approach to large-scale system management is presented by Silva et al [8]. In their work, they introduce the reflective blackboard architectural pattern for large-scale systems. That architecture combines two other well-known architectural patterns: the blackboard and the reflection patterns. The architecture separates control strategies from logic and data. In contrast to to their work, we use independent agents that act autonomously. Such a loose coupling is more beneficial in terms of simplicity, scalability, robustness and fault tolerance.

In the next section we describe the general problem. We then present a real world example. In Section 4 we sketch the DDM architecture and in Section 5 we detail a comprehensive study we conducted to evaluate the hierarchical approach. We conclude by discussing the major contributions of our solution to large-scale agent system challenges in terms of capability, accuracy, efficiency, cost-effectiveness, robustness and fault tolerance.

II. Problem Description

The general problem that is considered in this paper can be modeled in the following way. We define an *object*-tracking problem*, OTP,* as a tuple,

$OTP=\langle Z,O,S,T,A,Sa,\sigma,ResBy\rangle$ such that

- Z is a virtual area (*the controlled zone)*
- O is a set of tasks or objects existing in Z;
- S is a set of object/tasks' states;
- T is a set of integer times;
- A is a set of agents existing in Z;
- Sa is a set of agent states;
- $\sigma : O \times T \rightarrow S$ associates with each object and a time point the states of the object at that time;
- A relation, $ResBy \subseteq S \times T \times S \times T$, constrains the evolution of object states in terms of contiguous, legitimate paths, such that *ResBy(s1,t1,s2,t2)* iff *s2* at *t2* could follow *s1* at time *t1*.

We introduce the notion of an object state function $f_o : T \rightarrow S$ that associates with an object o its state change over time. If f_o is the actual path function then for any t, $f_o(t) = \sigma(o,t)$. We define an *information map,* as a set of object state functions. An information map, *infoMap*, is an accurate solution to a given *OTP,* iff each object in O is captured in an actual path function in the *infoMap*. There may be situations (e.g., not enough agents) where the agents will not be able to construct an accurate information map but rather will construct a partially accurate information map. We will assert that as the number of objects in O that are captured in an actual path function in *infoMap* increases, the accuracy of the *infoMap* increases too.

We assume that in order to solve an *OTP* the agents of A are able to make observations of the objects. Whether or not an object can be sensed by an agent depends on the state of that agent and that object. We assume that local observations obtained by an agent are incomplete and uncertain and, regardless of the agent's state, one observation is not enough to identify the exact state of an observed object. To form a correct f_o several observations, preferably by different agents, are needed. Thus, the agents should be provided with strategies to decide how to change their own state over time, when to make observations and with which agents to cooperate. In addition, they need to be given algorithms to locally process their observations and algorithms to integrate data of various agents. The processing and integration of agents' information must be accomplished in real time.

To capture the uncertainty associated with observed

information, we assume that each sampled object is associated with several possible *states*. We introduced the notion of a capsule that represents a few possible states of an object at some time as derived from agent observations at a given time, in a given state.

Definition 1: A *capsule* is a triple of a time point, an agent state and a sequence of object-states, i.e., $c =< t, sa, \{s_1,...,s_l\} >$ where $t \in T, sa \in Sa, s_i \in S$.

The formal definitions of the elements of an *OTP* are presented in [13,14]. In the next section, we demonstrate the use of *OTP* elements within a domain involving mobile Doppler sensors.

III. The Mobile Doppler Domain

We demonstrate the *OTP* problem in a mobile Doppler domain and also explain how capsules are produced in that domain. We built a simulation environment within that domain to test the DDM model in an environment consisting of a large number of agents and objects. In the simulation a suite of Doppler sensors are used to form a global information map of targets moving in a steady velocity. The problem was devised as a challenge problem by the DARPA Autonomous Negotiating Teams (ANTS) program to explore real time distributed resource allocation algorithms in a two dimensional geographic environment [1]. A Doppler sensor is a radar, based on the Doppler effect [6]. It has a wide beam of 120 degrees. A Doppler sensor can only provide information about an arc on which a detected target may be located as well as the velocity towards that sensor, that is, the *radial velocity*. Given a single measurement, one cannot establish the exact location of a target nor its exact velocity.

Formalizing the Dopplers problem as an OTP problem can be accomplished in the following way. In the Dopplers domain, objects correspond to targets. The target state structure is $s =< \bar{r}, \bar{v} >$. $\bar{r}$ is the target location vector and $\bar{v}$ is the velocity vector. If a target state s_2 at t_2 resulted from target state s_1 at t_1 and the velocity of the target remained constant during the period $t_1.t_2$, then $\bar{r}_2 = \bar{r}_1 + \bar{v}_1 \cdot (t_2 - t_1)$. Thus, in the Dopplers domain where, $s_i =< \bar{r}_i, \bar{v}_i >$, $ResBy(< t_1, s_1 >, < t_2, s_2 >)$ is true iff (i) r_2 can be derived from r_1 using a motion equation with $\bar{v}_1$ during the period $t_2 - t_1$ and (ii) $\bar{v}_1 = \bar{v}_2$.

We hypothesize that the following criteria should be considered when determining the sampler agents' movement: (i) the union of all sensed area at time *t* should be maximized and (ii) the intersection of the sensed areas at time *t* and at *t*+1 should be minimized. One of the ways to achieve this is to move in a *Patrol movement pattern* (Fig. 1). We compared the patrol movement with a steady random movement that was used in [13,14]. A steady random movement is defined as a movement in a random direction and velocity. Upon reaching the end of the controlled zone, the velocity and the direction are changed and direct into the zone. In [13,14] Yadgar *et al* showed that the steady random movement is significantly better than static agents.

IV. Constructing an information map

The goal of the DDM is to construct an information map that is as accurate as possible. Intuitively, *infoMap* represents an estimate of the state changes of objects over time. Because each agent has only partial and uncertain information of its local surroundings, there is a need to integrate information from several agents. In some domains, as in the Doppler domain, the most accurate *infoMap* can be constructed when the information obtained from all the agents is integrated. However, the complexity of such integration is usually very large and thus it cannot be done in real time. Instead we have adopted an iterative method in which integration is performed in stages. However, constructing the *infoMap* in stages decreases its accuracy because each stage involves only partial information. Thus, the challenge is to balance the accuracy of the constructed *infoMap* and the number of stages.

In a large-scale environment many capsules may need to be examined. Applying the *ResBy* relation many times to consider all possible pairs between states in capsules can be time consuming. However, there is a low probability that capsules created based on observations taken far away from one another will be related. Therefore, it makes sense to distribute the solution. The DDM hierarchical structure guides a distributed construction of the global *infoMap*. The base level of the hierarchy consists of the sampling agents that produce the capsules, which are grouped according to their associated area. Each group has a leader. Thus, the first level of the hierarchy consists of sampler group leaders. Sampler group leaders are also grouped according to their associated area. Each group of sampler leaders is associated with a zone group leader. Thus, the second level of the hierarchy consists of zone group leaders, which in turn, are also grouped according to their associated area, with a zone group leader, and so on. Leader agents are responsible for retrieving and combining information from their group of agents.

To avoid transferring all the capsules, we introduced an additional data structure that we refer to as *localInfo.*

A localInfo is a pair of a partially accurate information map, *PinfoMap,* and a set of unused capsules *unusedCapsules*. At each stage of the tracking process, some capsules can be used to form object state functions that have a high probability of representing objects. These functions are recorded in *PinfoMap.* The remaining capsules are maintained in the *unusedCapsules* set and are used to identify state functions at a later state. *localInfo's* are moved from one level of the hierarchy to the other. In summary, the formation of a global information map integrates the following processes: (i) Each sampling agent gathers raw sensed data and generates capsules; (ii) Every predefined dT seconds each sampler

group leader obtains capsules from all of its sampling agents and integrates them into its *localInfo* and (iii) Every dT seconds each zone group leader obtains *localInfo* from all of its subordinate group leaders and integrates it into its own *localInfo*.

We have developed several algorithms to implement each of the three processes and they are presented in [13,14].

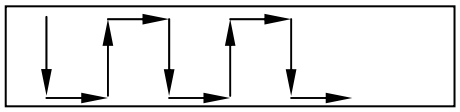

Fig. 1: Patrol movement pattern

V. Experiments and Results

A. Simulation environment

We developed a simulation of the Doppler domain to study the problems associated with the application of the DDM in a large-scale environment. The simulation consists of an area of fixed size in which Dopplers attempt to extract the object state functions of moving targets. Each target has an initial random location along the border of the area and an initial random velocity of up to 50 km. per hour in a direction that leads inwards. Targets leave the area when reaching the boundaries of the zone. Each target that leaves the area causes a new target to appear at a random location along the border and with a random velocity in a direction that leads inwards. Therefore, each target may remain in the area for a random period of time. Running the simulation for a long time while keeping a constant load of randomly generated targets is equivalent to many shorter runs. We ran the simulation each time for 3 days to simulate one hour. We preferred simulating one hour in one run instead of many shorter runs to avoid edge influences and to save setup time.

The averages reported in the graphs below correspond to one simulated hour. The *target tracking percentage time* was calculated by dividing the number of randomly generated targets that the agents succeeded in tracking by the actual number of targets. We considered only targets that exited the controlled zone. The *tracking time* was defined as the time that the agents needed to find the object state function of the target from the time the target entered the simulation. Tracking average time was calculated by dividing the sum of tracking time of the tracked targets by the number of tracked targets.

Basic Settings. The *basic setting* for the environment corresponded to an area of 10,000 by 10,000 meters. In each experiment, we varied one of the parameters of the environment, keeping the other values of the environment parameters as in the basic settings.

Dopplers: The Dopplers were mobile and executed either the random or the patrol patterns. Each Doppler moved one second and stopped for 5 seconds to take 5 measurements. The maximum detection range of a Doppler in the basic setting was 100 meters; the number of Dopplers was 1,000. For the patrol pattern simulations the controlled area was divided into 1,000 equal rectangles, each 400x250 square meters. Each patrolling Doppler was assigned to such an area and executed the patrol movement pattern. 1,000 Dopplers using the patrol pattern with a detection range of 100 meters yielded a coverage of 8% of the controlled area. Using random patterns only assures that this maximum coverage is an upper bound, due to a possible overlapping of sensed areas.

Targets: The number of targets at a given time was 1,500. In total, during one hour 4,200 targets passed and exited the controlled area.

Hierarchy: In the basic settings we used a hierarchy of 4 levels: three levels of zone group leaders and one of sampler group leaders. Each of the zone group leaders divided its zone into 4 quarters and put a sub-leader to lead each one of them. Therefore there was one leader at the top level, 4 at the second level, 16 at the third and 64 at the fourth. Each sensor communicated with one of the fourth-level leaders.

VI. Results

We conducted three sets of tests: (i) evaluating the basic settings, (ii) investigating the influences of the number of levels in the hierarchy, and (iii) studying the tolerance towards faulty sensing agents, leaders and sensing noises. At this stage of our research, samplers and leaders do not react to changes in the functioning agent community.

In the following figures, we present a detailed comparison between the patrol and random patterns. In the figures, the darker bar represents the patrol pattern while the brighter bar represents the random pattern.

Patrol vs. Random. Our hypothesis was that by applying the patrol pattern we would gain better results than when using the random Doppler movement. A better coverage of the controlled zone is achieved when using the patrol method and therefore the system should, in that case, track more targets. A better coverage should also lead to a faster tracking of targets and extend the period that each target is tracked. We ran the simulation using the basic settings and compared both methods and found out that with the patrol method the system tracked 83% of the 4,200 randomly generated targets while it tracked only 78% with the random method (this was achieved with Dopplers covering only 8% of the area). Not only did the patrol method track significantly more targets, it also tracked them significantly faster. More than 50% of the targets that stayed in the controlled zone less than 360 seconds were tracked when using the patrol pattern while less than 40% were tracked when using the random method. Most of the targets passing through the simulated area remained in the area less than 720 seconds. We discovered that both patrol and random methods achieve most of their tracking in less than 2 minutes from the time of a target's entrance into the zone: while the patrol tracks 71% of its tracked targets in this period the random tracks only 63%. We found that the patrol pattern also tracked targets for significantly longer time than the random method. As expected, for the basic settings the patrol

pattern performed significantly better than the random method. It performed better in three ways: (i) it tracked significantly more targets, (ii) it tracked them significantly faster and (ii) it tracked them for a significantly longer time.

Level comparison. We investigated the influence of the number of levels in the hierarchy. Our hypothesis was that too few levels would overload the leaders so they would not have enough time to process the information. We also anticipated that, as more leader agents were involved in generating the global solution, the less accurate the solution obtained.

Fig. 2a presents the tracking performances of both methods as a function of the number of levels in the hierarchy. As we hypothesized, both methods tracked fewer targets as the number of levels increased. This can be explained by a greater fragmentation of the zone, i.e. 4 quarters in 2 levels versus 64 in 4 levels. The figure shows that the decrement is narrow. Comparing the tracking percentage of the methods proves, once again, that the patrol movement tracked significantly more targets regardless of the number of levels.

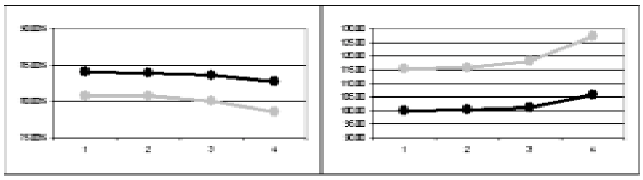

Fig. 2: Left(a) - Tracked target percentage as a function of the number of levels. Right(b) Tracking time (Sec.) as a function of the number of levels

As shown in Fig. 2b, the patrol movement method also tracked the targets significantly faster than the random method regardless of the number of levels. Moreover, as we anticipated, as the number of levels decreased the system tracked targets faster.

In Fig. 3a we present the maximum time an agent needed to perform its task using our computer capabilities as detailed above. The maximum time is very close to the average time; therefore we do not present the latter here. As we predicted, using one level only is not sufficient to form a global solution in real time. It took the patrol and the random methods about 4 hours to reach a solution for one simulated hour. Using 2 levels enabled the system to solve the problem in only 35 minutes. Using 4 levels decreased the maximum time that an agent needed to perform its task to 10 minutes.

We assume that the patrol pattern needed more time than the random because it tracked more targets. When we divided the maximum agent process time presented here by the number of tracked targets, the patrol and the random graphs almost overlapped.

Dysfunctional sampling agents. To investigate the fault tolerance property of the DDM in a large-scale environment we had the sampling agents stop functioning. We increased the number of damaged sampling agents from 0% as in the basic settings to 90%, leaving only 10% active agents. We hypothesized that by increasing the number of faulty sampling agents the system would not perform as well as in the basic settings. However, the system would be able to perform well for a certain number of disabled sampling agents. Our goal was to place a bound on the number of dysfunctional sampling agents that the system could tolerate while still performing well.

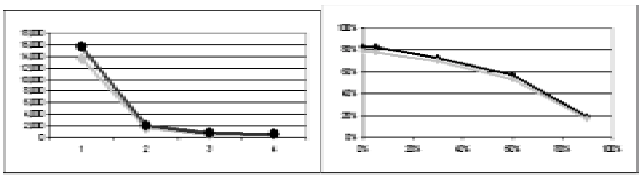

Fig. 3: Left(a) - Maximum agent process time (Sec.) as a function of the number of levels. Right(b) - Tracked target percentage as a function of dysfunctional samplers.

Fig. 3b presents tracked targets percentage as a function of the number of samplers that stopped functioning. Once again, we can see that the difference between the two methods is very small even though the patrol method consistently tracks more targets. As we stated in Section 5, we hypothesized that as the number of dysfunctional sampling agents increases the difference between the performances of the patrol and the random patterns will increase in favor of the random method since no re-organization is made after some of the patrol agents stop functioning. Fig. 3b does not indicate any such anticipated difference between the patrol and the random performances. That can be explained by the fact that the patrol pattern still has a better coverage of the controlled area. We also found out that increasing the number of disabled stopped sampling agents also increases the time it takes to track targets. By increasing the number of disabled sampling agents by 5% the average time it takes to track a target increased by 6%.

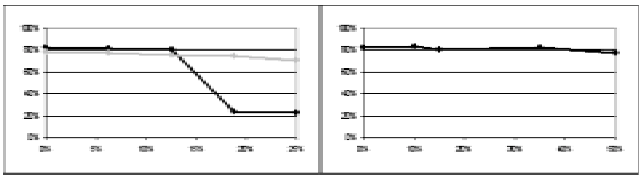

Fig. 4: Left(a) - Tracked target percentage as a function of dysfunctional first level leaders. Right(b) - Tracked target percentage of patrol as a function of lost communication messages between samplers and leaders

Dysfunctional leader agents. A second aspect of the system's fault tolerance is the tolerance towards dysfunctional leaders. In contrast to samplers, a dysfunctional leader will result in a difference in the coverage of the patrol and the random patterns. For example, consider a case in which a leader responsible for half of the controlled area stops functioning. Using the patrol pattern will result in a loss of information from half of the samplers. However, using the random method the system will only loose information from half of the samplers, on average. At certain points it will use more samples and at other times it will use less. We expected this behavior to benefit when implementing the random pattern. To validate this hypothesis we conducted simulations in which we varied the number of dysfunctional sampler leaders.

Fig. 4a confirms our hypothesis. It shows that the patrol pattern could tolerate reduction of up to 13% in the number of functioning sampler leaders. A reduction of 18% or more, results in a very low performance level. The random pattern, on the other hand, was much more tolerant to the same reduction. However, despite the fact that the patrol pattern resulted in a poor tracking percentage for high-rate dysfunctional sampler leaders, we found out that it tracked targets faster.

Noisy communication. As we stated, we would like to show that using simple and cheap sensors may be beneficial even if they tend to malfunction or if communication with their leaders degrades. We conducted a simulation testing the patrol model while using faulty communication between samplers and leaders. We predicted that the system would be tolerant towards such noise.

We found that even if 50% of the messages sent by the Dopplers did not reach their destination (either because of faulty communication or faulty samplers), the system still performed well. Losing 50% of the messages resulted in a reduction of only 5% of the tracked targets and increased the tracking time by 20 seconds.

VII. Conclusions

We have introduced a solution to a set of large-scale system problems. We have shown that such problems involving hundreds and thousands of agents and objects cannot be solved in the traditional flat architecture. Distributing the solution into smaller fragments of problems that can be solved partially by simple agents is the approach we presented. Using many simple and cheap agents instead of a much smaller number of sophisticated and expensive ones may also be cost-effective: it is often more affordable to replace and maintain many simple agents than to depend on a few sophisticated ones. We also suggested ways to combine partial solutions to form a global solution. We established an autonomous movement algorithm to be implemented by each sampling agent and showed in the Doppler domain how it out-performs the random movement method. In that domain, we have also shown that the capabilities of the hierarchical model are greater than that of the flat one. In particular, the flat model could not solve the problem we addressed at all.

We have shown that the number of levels in the hierarchy influences the accuracy of results. As the number of levels increases the number of tracked targets drops, even though the drop is moderate. However, as the number of levels increased the time every agent needed to complete its mission dropped exponentially. By combining these two results we can balance the two properties to achieve the desired behavior. Choosing the right number of levels should also take into consideration the time it takes to track targets. As we have shown, it takes more time to track targets as the number of levels in the hierarchy increases.

We have presented two autonomous search patterns that samplers may adopt. We have established our hypothesis that, as long as leader agents are functioning, the patrol pattern will perform better than the random method. However, the performances of the random pattern are very close. On the other hand, the random pattern is much more tolerant towards disfunctioning leader agents.

To conclude, we have shown that in the Doppler domain large-scale agent systems can perform well even if agents are very simple and inaccurate. We have shown how partial information can be combined and how the existence of dysfunctional participants can be overcome. In short, we have presented a method to “transform quantity into quality” by applying a cooperative mechanism.

References

[1] ANTS Program Design Document, unpublished. 2000.

[2] D. Corkill and V. Lesser “The use of meta-level control for coordination in a distributed problem solving network,” in proceedings of the Inter. Joint Conference on Artificial Intelligence, 1983, pages 748–756.

[3] T. Ishida, L. Gasser, and M. Yokoo, “Organization self design of production systems,” in *IEEE Transactions on Knowledge and Data Engineering,* 4(2):123-134, 1992.

[4] V. R. Lesser, D. D. Corkill and E. H. Durfee, “An update on the Distributed Vehicle Monitoring Testbed”, CS Technical Report, University of Massachusetts, Amherst,1987, 87-111.

[5] C. L. Ortiz et al., “Incremental negotiation and coalition formation for resource-bounded agents”. AAAI Fall Symposium on Negotiation Methods for Autonomous Cooperative Systems, 2001.

[6] Pöss, Christian Doppler in Banska Stiavnica, in *The Phenomenon of Doppler* (Prague, 1992)

[7] W. Richard Scott, *Organizations: Rational, Natural and Ope*n, Prentice-Hall, 1992.

[8] Silva, O.; Garcia, A; Lucena, C. J. “The Reflective Blackboard Architectural Pattern for Developing Large Scale MA Systems”. Proc. of the 1st Inter. Workshop on Software Engineering for Large-Scale MA Systems at ICSE 2002, Orlando, USA, May 2002.

[9] Y. So and E. Durfee, “Designing Tree-Structured Organizations for Computational Agents,” *Computational and Mathematical Organization Theor*y, 2(3), 1996, 219-246.

[10] Y. So and E. H. Durfee, “Distributed Problem-Solving Infrastructure for Computer Network Management.” *International Journal of Intelligent and Cooperative Information System*s, 1(2):363-392,1992.

[11] Leen-Kiat Soh, C. Tsatsoulis, Reflective Negotiating Agents for Real-Time Multisensor Target Tracking. IJCAI 2001, 1121-1127.

[12] R. Vincent et al, Implementing Soft Real-Time Agent Control. In Autonomous Agent. 355-362. June, 2001.

[13] O. Yadgar, S. Kraus and C. L. Ortiz, “Hierarchical organizations for real-time large-scale task and team environments.”, in Proc. of AAMAS, 2002.

[14] O. Yadgar, S. Kraus and C. L. Ortiz, “Information Integration Approach for Large-Scale Multiagent R.M.”, in *Communication in Multiagent Systems* (to appear), 2003.