

Hierarchical organizations for real-time large-scale task and team environments

Osher Yadgar
Dept. of Math and CS,
Bar Ilan University,
Ramat Gan, 52900 Israel
yadgar@macs.ac.il

Sarit Kraus
Dept. of Math and CS,
Bar Ilan University,
Ramat Gan, 52900 Israel
sarit@macs.biu.ac.il

Charles L. Ortiz, Jr.
Artificial Intelligence Center,
SRI International,
Menlo Park, CA 84025 USA
ortiz@ai.sri.com

ABSTRACT

In this paper, we describe the Distributed Dispatcher Manager (DDM), a system for managing large collections of dynamically changing tasks. We assume that tasks are distributed over large geographic areas and that teams consist of very large groups of mobile and cooperative agents which have direct access to only local information about their immediate environment. DDM organizes teams hierarchically and addresses three important issues that are prerequisites for success in such domains: (i) how the agents should be distributed over the area, (ii) how agents should process local information, derived from possibly 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 to very large task and agent problem domains. We describe extensive experimentation in a simulated domain.*

Content areas: Coordinating multi-agents and activities, group and organizational dynamics, scalability and complexity issues.

1. Introduction

This paper considers the problem of managing large collections of dynamically changing tasks, distributed over large geographic areas, by very large teams of mobile and cooperative agents which have direct access to only local information about their immediate environment. We are particularly interested in problem domains in which scalability, fault tolerance and real-time behavior are a central concern. There are several domains where such problems arise: satellites that are tasked to form a general picture of a large area, satellites that form weather maps; agents that control air pollution or ocean pollution; sensor webs that monitor geographic areas for passing aircraft; and unmanned air and ground vehicles that must be jointly tasked on surveillance missions. In such domains, there are three important issues that represent prerequisites for success: (i) how the agents should be distributed over the area, (ii) how agents should process local information, derived from possibly noisy sensors, to provide a partial solution to nearby tasks, and (iii) how partial solutions should be integrated into a global solution.

We describe the Distributed Dispatcher Model (DDM), an agent based computational model of semi-centralized task allocation. DDM is designed for efficient coordinated task allocation in systems consisting of hundreds of agents (resources); the model makes use of hierarchical coalition formation to restrict the degree of communication between agents necessary for coordination. Our main contribution is threefold. First, the hierarchical team organization supports processes for very quickly combining partial results to form an accurate global solution. Each level narrows the uncertainty about the solution based on the data obtained from lower levels. A second contribution of our work is the use of hierarchical team organization to increase system fault tolerance. We provide a mechanism through which disabled agents can be replaced efficiently by other agents in the hierarchy. A final benefit of the hierarchical team organization is scalability to very large task and agent problem domains.

To motivate and test the model, we draw on a challenge problem taken from the DARPA Autonomous Negotiating Teams (ANTS) program. The ANTS challenge problem is a command and control problem in which sensing agents must be allocated to moving targets (tasks) in a distributed and real time fashion. Targets must be located reliably and resource allocations must achieve good coverage of the controlled area. In this domain, tasks vary with time; it is therefore important for the agent system to acquire accurate task information over time, a process which we refer to as forming an *information net*. With respect to the issues mentioned above, we will focus on the second and the third issues. For reasons of space we do not describe the algorithm we employ for dynamic task allocation. We present a domain specific solution that facilitates the processing of local and noisy information available to each sensing agent. Our methods for the integration of the agents' partial solutions are general and can be applied in other domains.

2. The Ants Challenge Problem

The ANTS problem makes use of Doppler sensors to form an information net of targets in a controlled area as a function of time; each Doppler sensor has a wide beam 120 degrees. A Doppler sensor is a radar, based on the Doppler effect. Due to its nature Doppler sensors may provide information only about either an arc that the detected target may be located on or the velocity towards the sensor, that is, the *radial velocity*. Given a single Doppler measurement, one cannot establish the exact location of a target and its exact velocity.

* This work was supported by DARPA contract F30602-99-C-0169 and NSF grant 9820657. The second author is also affiliated with UMIACS.

One way to establish the exact location and velocity of a sensed target using Doppler sensors is to use three Doppler sensors at the same time and intersect the resulting arcs of each ([15]). The intersection point represents the location of the target while the velocity may be derived from a geometric vector analysis of the three radial velocity vectors. The results of this method are accurate and fast. However, alongside these benefits are some drawbacks. The intersection method depends on the coordinated action of three Doppler sensors to simultaneously sample the target. Such coordination requires good synchronization of the clocks of the sensors and therefore communication among the Doppler agents to gain that synchronization.

We have developed another method involving sequential measurement sampling by a single Doppler radar to compute the location and velocity of targets. The single Doppler retrieves several pairs of location and velocity measurements. This method requires less frequent synchronization among sensors and fewer Doppler sensors to acquire an information map. This method also has a desirable fault tolerant property: it does not rely on a specific combination of sensors to determine the target's state. Since sensors can estimate location and velocity measurements on their own, this method allows agents to be mobile, thereby increasing the coverage of a controlled area.

3. The DDM model

In DDM, centers are organized hierarchically and each is associated with a particular geographic area. We refer to each center as a coalition leader. Mobile sensors report to an immediate coalition leader that passes the information to its leader. Each coalition leader passes down instructions to direct sensors. We consider several DDM entities.

The first is a target which has the properties of location and velocity. We assume that targets generally move at a steady speed, however they may change their velocity from time to time.

The second is a sensing agent which also has location and velocity properties. A fixed number of Doppler sensing agents are available to track targets in the controlled area. The agents use an internal clock to tag each measurement with a time. The clocks in the system need only be synchronized occasionally; although this can result in small differences between the clocks; the system can deal adequately with such variations.

Doppler radars are based on the Doppler phenomenon: the relative motion of a source and a detector affects the observed frequency of light or sound coming from the source [10], [9], [7], [3]. A Doppler sensor measures the amplitude and relative velocity of a target as a result of the frequency change. The field of detection depends on the characteristics of the particular sensor. In our environment, given an amplitude for a target the target can be located along the line represented by [1]:

$$(1) \quad R_i^2 = \frac{k \cdot e^{-\frac{(\theta_i - \alpha)^2}{\sigma}}}{A_i} \quad \text{where, for each sensed}$$

target, i , R_i is the distance between the sensor and i ; θ_i is the angle between the sensor and i ; A_i is the measured amplitude

of i ; α is the sensor beam angle; and k and σ are characteristics of the sensors and influence the shape of the sensor detecting area.

Each sensing agent's Doppler radar is divided into three 120° sectors. Each agent is mobile and capable of rotating along its main pivot. We use the sensing agent to receive information about targets in the form of raw sensed data.

Definition 1: *Raw sensed data* is represented as triple, $\langle t, A, V_r \rangle$ where A is the sensed amplitude, V_r is the velocity towards the sensor, and t is the time the measurement was taken. [1]

The third participating entity in the DDM model is the sampler agent. To every sensing agent we attach a sampler agent. The sampler agent requests raw sensed data from the sensing agent. The sampler agent then computes the possible states for each sensed target according to the raw sensed data. Target states are produced using N consecutive measurements.

Definition 2 The *task state* is a set of domain oriented characteristics representing the state of a task at time T . In our domain, we will associate a task state with the state of a target. The pair $\langle \vec{D}, \vec{V} \rangle$ represents the *target state*, where \vec{D} and \vec{V} are vectors representing possible locations and velocities of a target at a particular time.

Definition 3: The *sensing agent state* represents the state of the sensor at the time the raw sensed data is gathered. In our domain, the sensing agent state is represented by the triple $\langle t, \vec{D}, O \rangle$ where t is the sensing time, \vec{D} is the location of the sensor and O is the orientation of the sensor.

Every sensed task may be associated with several possible *task states* derived from the same raw sensed data. All possible task states are combined with the sensor to form a capsule.

Definition 4: A *capsule*, $\langle \text{Sensing agent state}, \{\text{Task state}_1, \dots, \text{Task state}_N\} \rangle$, is a set of task states corresponding to a given sensing agent state. A capsule represents a few possible states of a task at time t as measured by the sensor in a given state.

The fourth entity is the coalition leader agent. The coalition leader agent controls a certain area. Each higher level controls a larger area. We will refer the controlled area as a *zone*.

We distinguish between two types of coalition leader agents: a *zone coalition leader* agent and a *sampler coalition leader* agent. While the zone coalition leader controls other coalition leaders the sampler coalition leader controls the behavior of a set of sampler agents in its controlled zone. The main goal of both types is to obtain information about tasks in its controlled area.

Another important role of the zone coalition leader is to balance the number of sensing agents in its area. The main difference between a zone coalition leader agent and a sampler coalition leader agent is that the first is responsible for an area combined out of different zones while the latter is directly responsible for the behavior of the sensing agents in a specific zone. Therefore, many of the zone coalition leaders may be in charge of other zone coalition leaders at a lower level. The lower zone coalition leaders are in turn in charge of sampler coalition leaders.

We do not distinguish between the two types of coalition leaders when we apply the algorithm of information net formation in the top-level coalition leader. However, the control algorithms for zone coalition leaders and sampler coalition are different. Whereas the zone coalition leader should balance the number of sensing agents between zones and should decide how many should pass from one zone to another, the sampler coalition leader should follow orders from its superior zone coalition leader and decide which sensing agent to pass and how to do this. Sampler coalition leader should also direct the sensing agents in the zone that it is in charge of. We will not discuss the latter sampler role further in this paper.

The DDM design has important fault tolerant properties. A sampler agent can calculate possible states of a task without the need of another agent. A sampler agent may also acquire information about the same task from a different point of view and establish the right path for the task. As we use more sampler agents we gain better performances. However, for a large number of sampler agents the performance of the system is not significantly affected if some of the samplers are not operational. In addition if one of the leaders is disabled, it is replaced by one of the other agents below it by its leader. If the top-level leader stops functioning, then the leaders at the second level will distributively choose a top-level leader replacement.

4. Algorithm descriptions

We have developed several algorithms that can be activated by the agents acting in our environment. The first algorithm describes the method for constructing a capsule from raw sensed data. This algorithm is activated by each sampler agent and uses consecutive raw sensed data. A sampler agent executes this algorithm using data gathered by a sensing agent. The complexity of each capsule generation is $O(1)$. The second and the third algorithms describe the way in which a top-level coalition leader process the capsules generated by sampler agents. The aim of this process is to form a net of task states that represents possible tasks and their location as a function of time.

As we have stated, we assume that a target is moving in an almost steady velocity, i.e. the speed and the direction of a target changes infrequently. We use this assumption to calculate the location and velocity of a target by using four situational measurements of one Doppler and matching those measurements with prior target information. Given a very small period of time between two consecutive measurements, a target is unlikely to accelerate drastically during it. As a result, the velocity of a target can be assumed to remain constant during the four consecutive measurements. For cases in which this assumption is an over-simplification, the computed location may be inaccurate and later measurements may have to be used. The second assumption is that each Doppler may sense all the targets in its range of detection and obtain the amplitudes and radial velocity in one measurement.

4.1 Target location and velocity algorithm

We use one Doppler agent and one sampler agent to deduce a set of possible target states at a given time. The Doppler agent senses the data while the sampler agent processes the data. It is

known that if the location of a body at time 0 is D_0 and its velocity is V than the next location, D_1 , at time 1 of a body ([6], [5]) is given by:

$$(2) \quad D_1 = D_0 + \int_{t_0}^{t_1} V dt$$

where D_t is the displacement of a body in time t. If we consider the distance from the center of the Doppler we may claim that

$$(3) \quad R_1 = R_0 + \int_{t_0}^{t_1} V_r dt$$

where R_t is the displacement from the center of the sensor at time t and V_r is the relative velocity between the Doppler and the target in the direction of the Doppler's center. We will refer to this velocity as the *radial velocity* of the target. We assume that the acceleration of a target in a short time period is zero. The next target location after a very short period of time is then:

$$(4) \quad R_1 = R_0 + V_r \cdot (t_1 - t_0)$$

We denote $(t_1 - t_0)$ by $t_{1,0}$. Knowing the relation between R , θ and A we can find the next angle as a function of the former:

$$\theta_1 = \alpha_1 + \sqrt{-\sigma \cdot \ln\left(\frac{A_1}{k} \cdot (R_0 + V_{r,0} \cdot t_{1,0})^2\right)} \quad \text{and} \quad R_0 = \sqrt{\frac{k \cdot e^{-\frac{-(\theta_0 - \alpha_0)^2}{\sigma}}}{A_0}}$$

Where $R_0, \theta_0, A_0, V_{r,0}$ and α_0 represent values of the target at time $t = 0$ and θ_1, A_1 and α_1 represent values of the target in time $t = 1$. The same holds for the next angle, θ_2 .

We use the relationship between θ_0, θ_1 and θ_2 to find θ_1 and θ_2 from θ_0 . We can then find the exact location of a target in time 0,1 and 2 as a function of θ_0 : $Loc_0(\theta_0)$, $Loc_1(\theta_0)$ and $Loc_2(\theta_0)$. We assumed that during the short period $t = 0..2$ the target kept an almost steady velocity therefore: $V_{2,1}^{\omega} = V_{1,0}^{\omega} \rightarrow \frac{dLoc_2(\theta_0)}{dt} = \frac{dLoc_1(\theta_0)}{dt} \rightarrow \frac{Loc_2(\theta_0) - Loc_1(\theta_0)}{t_{2,1}} = \frac{Loc_1(\theta_0) - Loc_0(\theta_0)}{t_{1,0}}$

These equations cannot be solved symbolically therefore the sampler agent uses computational methods. The sampler agent scans the range of θ_0 and looks for suitable locations corresponding to θ_0 . Only certain angles will fit the above equation. To be more accurate, the sampler agent uses one more sample and applies the same mechanism to θ_1, θ_2 and θ_3 . Comparing the results from both cases assure accurate results. The calculated angles will be used to form a set of possible pairs of location and velocity of a target (i.e., a capsule). Only one of the target states in a capsule is the correct one. The next section shows how to identify the correct state.

Theorem 1 The time complexity of the target location and velocity algorithm is $O(1)$.¹

¹ For reasons of space, proofs of theorems are omitted.

4.2 Forming the task states net algorithm

The task states net formation algorithm makes use of the hierarchical agent organization to achieve global mapping of the controlled area. Each coalition leader forms a map of its controlled area, whereas, the top-level coalition leader (level 0) forms a global map of the entire area.

Definition 5: A *path* is a linked list of task states representing a possible task and its location as a function of time. We denote the first task state of the path by *start task state* and the last task state of the path by *end task state*.

Definition 6: A *task states net* is a directed acyclic graph of *task states*: represents possible tasks and their location as a function of time. The task states net uses two kinds of vertices and two kinds of edges. A vertex may be either a task state or a sensor state. An edge may be either a path edge or a capsule edge. A path edge links two task states to form a path list as defined in Definition 5. A capsule edge links a sensing agent state to N task states to form a capsule as defined in Definition 4. Note that only one task state in a capsule can be valid, therefore, only one path is valid out of those who share the same capsule.

Let us consider a case of two targets and one sensor with one sector. At time 0 one of the targets is located at point (100,100) with velocity of (5,5) while the other target is located at (50,50) with velocity of (2,3). At time 0 and 1 the sensor may sense these two targets, however it may not know exactly where they are. It may compute several possibilities for a location and velocity for each target in a form of a capsule. In our example suppose that the sensor computed only two possible target states for each target (i.e., each capsule consists of two target states).

Cap.	Sensing Agent State			Target State A		Target State B	
	Time	Loc.	Ori.	Loc.	Vel.	Loc.	Vel.
0	0	0,0	0	100,100	5,5	60,60	3,-1
1	0	0,0	0	50,50	2,3	30,40	-2,-2
2	1	0,0	0	105,105	5,5	63,59	3,-1
3	1	0,0	0	70,80	2,3	52,53	2,3
4	2	0,0	0	66,58	3,-1	110,110	5,5
5	3	10,10	0	54,56	2,3	30,30	3,3

According to equation (2) if target state X resulted from target state Y then $R_X = R_Y + \int_{t_Y}^{t_X} V dt$

If the velocity of a target remains constant during the period $t_Y.t_X$, then $R_X = R_Y + V_Y \cdot (t_X - t_Y)$. We assume that no target is likely to appear exactly as another target. Therefore, two samples that fit the behavior of one target in a steady velocity are expected to belong to the same target. The following captures the dependency of one task state on another.

Definition 7: A target state S_2 *physically results from* target state S_1 iff: (i) the location of target state S_2 may be derived from the location and velocity of target state S_1 using equation (2), and (ii) the velocity of both target states is the same.

We can now build a directed graph connecting only the feasible target states. Every node in the graph represents different possible target states. Each path may represent a feasible target

history. Some of the paths have a larger probability of representing the right path of the target. The Coalition leader algorithm that forms a target states net will specify how to form the target states net demonstrated by the example. We will now show how to select the most suitable paths.

The two factors that define the probability that a path represents the true path of a target are the number of target states that make up a path and the variety of sampling sources made use of in determining the target states of the path. As the number of target states increases the probability that the path represents the true path of a target increases. A path consisting of target states computed based on distinct agent states (e.g., from different locations) will represent a better estimate of the true path.

A measure of the variety of the sampling sources of a path can be computed by summing the standard deviation, stDev, of the agent states associated with the path (Definition 3): x and y coordinates of D and the orientation, o .

$$(5) \text{ grade} = \text{stDev}^2 \{x \mid (t, (x,y), o) \in \text{path}\} +$$

$$\text{stDev}^2 \{y \mid (t, (x,y), o) \in \text{path}\} + \text{stDev}^2 \{o \mid (t, (x,y), o) \in \text{path}\}$$

We will take the path that has the higher value as the best estimate of the target path. If there are several candidate paths then the probability is lowered. In the previous example there were three possible paths, listed below.

	Path 1			Path 2			Path 3		
Grade	0			0			66.67		
Length	2			2			3		
Velocity	5,5			3,-1			2,3		
	C	T	Loc.	C	T	Loc.	C	T	Loc.
	0	0	100,100	0	0	60,60	1	0	50,50
	2	1	105,105	2	1	63,59	3	1	52,53
	4	2	110,110	4	2	66,58	5	3	56,59

Path 3 involves three capsules; it is the only possible path involving those three capsules. The grade of this path is 66.69; this means that it emerged from more than one point of view. As the measurements' source varies the grade will be greater. For this reason, path 3 represents a good candidate target path.

Path 1 and 2 are not using any of the capsules of path 3: they must represent another target. However, the same three capsules and the same point of view formed Path 1 and 2. They both received the grade of 0 and have equal length. We can assume that one of these paths represents a target, however we cannot determine the path with certainty. We can conclude that the algorithm found two targets. We know for certain that the path of the first target, path 3, represents the true path of the target. The second target followed either path 1 or path 2. Therefore, each path has a 50 % probability of representing a target.

4.2.1 Coalition leader target states net formation

To form a target states net (Definition 6) the coalition leader uses the previous net and updates it with new capsules. At the beginning of the process the net is empty. Each coalition leader implements the following generic algorithm every dT seconds (note that in our domain, the tasks are the tracked targets).

Task states net formation algorithm

Input: Network of target states (*old_net*)

Output: The new network of task states (*new_net*)

A set of higher probability task paths (*g_paths*)

A set of mediocre probability task paths (*m_paths*)

Create empty capsules set, *caps*.

For each subjugated coalition leader or sampler, S

 Ask S for capsules and add them to *caps*

new_net = **Capsules combination algorithm** (*old_net*, *caps*)

path_list = all start points of paths in *new_net*

Grade each path in *path_list*

Sort *path_list* by increasing order of the grade

Mark all capsules in *new_net* as not used

Create empty *g_paths*

Create empty *m_paths*

For each path *P* in *path_list* with grade > 0

 If all capsules participating in *P* are marked as not used

 Add *P* to *g_paths*

 Mark all the capsules participating in *P* as used

For each path *P* in *path_list* with grade == 0

 If all capsules participating in *P* are marked as not used and

P.length() > threshold

 Add *P* to *m_paths*

Return *new_net*, *g_path* and *m_path*

4.2.2 Capsules combination algorithm

Every dT seconds new capsules are computed by the sampler agents. The capsules combination algorithm returns a target states net. The capsules are sorted by their time stamp and the earliest capsule is generated after the oldest capsule in the net.

Capsules combination algorithm

Input: Network of task states (*old_net*) Set of capsules (*caps*)

Output: The new network of task states (*new_net*)

new_net = *old_net*

end_list = all end of paths in *new_net*

For each capsule, *cap*, in *caps*

 For each task state, *TS*, in *cap*

 For each task state, *ETS*, in *end_list*

 If physically resulted (*ETS*, *TS*)

 Link *TS* to *ETS*

 Remove *ETS* from *end_list*

 Add all task states of *caps* to *end_list*

Return *new_net*

Theorem 2 Let T stand for the length of the time interval associated with the nodes of the task and G for the maximum number of tasks at a given time point. The complexity of the task states net formation algorithm is $O(T * G^2 + T * G \log(T * G))$.

4.2.3 Capsules discarding algorithm

With time the size of the target states net increases. As the net reflects possible target paths as a function of time some of the

earlier data may be redundant. As we saw in the former algorithms the order of the calculations depends on the number of target states and paths. To be able to calculate in real time we discard old and redundant target states and decrease the size of the net. We discard old capsules and capsules that after some time did not take part in any targets' path. Every dT seconds a coalition leader who holds a target states net will perform the following algorithm:

Capsules discarding algorithm

Input: Network of task states (*old_net*)

Output: The new network of task states (*new_net*)

new_net = *old_net*

scan_list = all start of paths in *new_net*

For each task state, *TS*, in *scan_list*

C = the capsule of *TS*

 If the time stamp of *C* < *threshold_T1* and *TS* is linked forward

 Add the next task state element to the *scan_list*

 Remove *TS* from the net

 If the time stamp of *C* < *threshold_T2* and *TS* is not linked forward

 Remove *TS* from the net

Discard every capsule that all its task states were removed from the net.

Return *new_net*

Theorem 3 The order of the capsules discarding algorithm is $O(T * G)$, where T and G are as in Theorem 2.

5. Simulation, experiments and results

We developed a simulation to test the model. The simulation describes an area of 1200 over 900 meters. In our experiments, at any given time, there were up to 30 targets in the area. Each target had an initial random location and an initial random velocity, with a speed limit of 50 kilometers per hour (13.9 meters per second). Targets leave the area when reaching the boundaries of the zone. Each target that leaves the area causes a new target to appear at the same location with the same velocity in a direction that leads it into the area. Therefore, each target may remain in the area a random time. In particular, 29% of targets in our experiments remained in the area less than 60 seconds. Each Doppler sensor has initial random location and velocity that is up to 50 kilometers per hour. Every sector of each Doppler was able to detect targets in the range of up to 200 meters according to the sector's formula as presented above. When a mobile Doppler gets to the border of the controlled area it bounce back with the same velocity. We run the simulation for one hour of simulated time. The criteria for fitness of target to a calculated path were: (1) the distance between the calculated Location(t) and the real Location(t) is less than 1 meter, and (2) the difference between the calculated V(t) vector and the real V(t) vector is less than 0.1 meter per second.

At the end of each simulation the estimated paths identified by the top-level coalition leader were examined. The identified paths were divided into two categories: in one only a single path was associated with a particular target: those paths are assigned a 100% probability of corresponding to the actual path. In the

second, two possible paths are associated with a target and each assigned a 50% probability of corresponding to the actual path.

We began with a *basic setting* of the simulation environment consisting of the hierarchy model with mobile Dopplers and we varied the active sensor sector every time they reached a sampling stage. The maximum detection range in the basic setting was 200 meters. In that setting, the *number of Dopplers* was 20 and the *number of targets* was 30. We varied the methods employed by the system - demonstrating the benefits of the hierarchy model over a flat model - as well as other characteristics of the basic setting. For example, we tested the influences of the number of targets and Dopplers as well as the coverage area of the Dopplers on performance. To compare the performances of the various scenarios, we define the following notion of *tracking time*. The tracking time is important as a measure of the speed of target tracking in the basic setting in comparison with other settings and the characteristics that influence it.

Definition 8: *Tracking time* is the time that the system needed to find the representing path, either *100% path* or *50% path*.

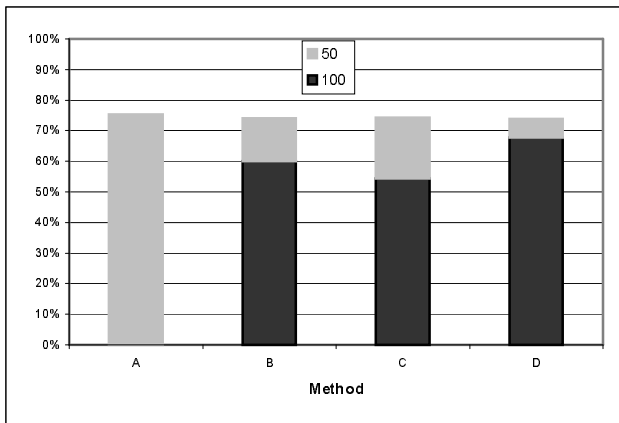


Figure 1: Target tracking percentage versus method used.

5.1 Method comparison

We started by comparing several methods to test the hierarchy, taking into account the mobility of the Dopplers and the variation in choice of active sector. We examined the characteristics of 4 different methods; D was the basic setting.

method	A	B	C	D
mobile Doppler	no	no	yes	yes
varying sector	no	no	yes	yes
hierarchy	no	yes	no	yes

Figure 1 shows that changing the method does not affect the number of targets that will not be detected. That may be explained by the fact that all the methods covered an equal area over time. The difference between the methods is presented by the division of the detected target between accurate tracking and the mediocre tracking. Method D performed significantly better than other methods. This supports our claim that a hierarchical organization leads to better performance. Further support for the benefit of the hierarchical structure comes from method B being significantly better than method C even though method B uses

more primitive Dopplers than method C. The only advantage of method B is the use of a hierarchical organization.

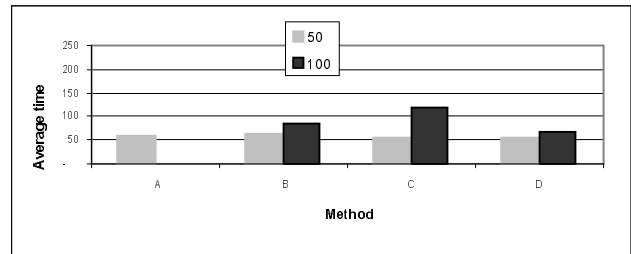


Figure 2: Average target tracking time versus method used.

Another aspect of the performances of the models is the average tracking time as shown in Figure 2. While method A could only find 50% paths, methods B, C and D could track targets accurately. Method D could track more targets accurately; it also tracked them faster than the other methods. Once again, one can see that the hierarchical designs lead to better results.

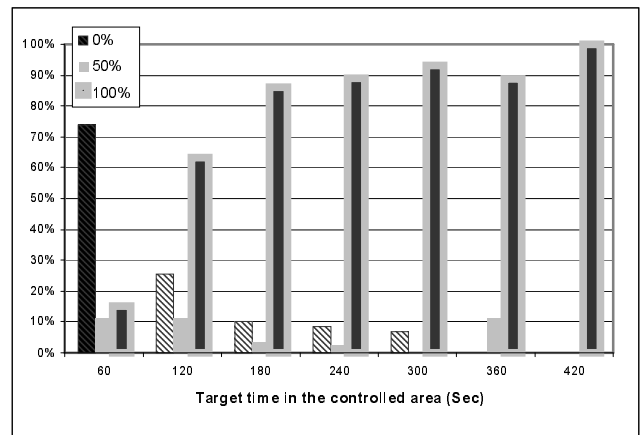


Figure 3: Target time in controlled area

Each column in Figure

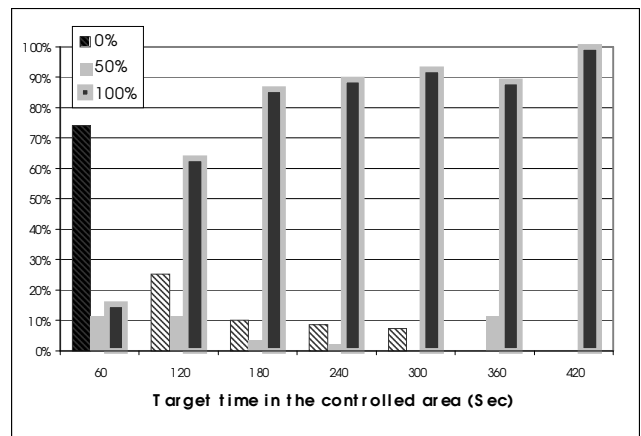


Figure 3, each column represents 100% of the targets that stayed in the zone the indicated period of time. For example, 86% of the targets that remained 120-180 seconds in the controlled area were accurately tracked.

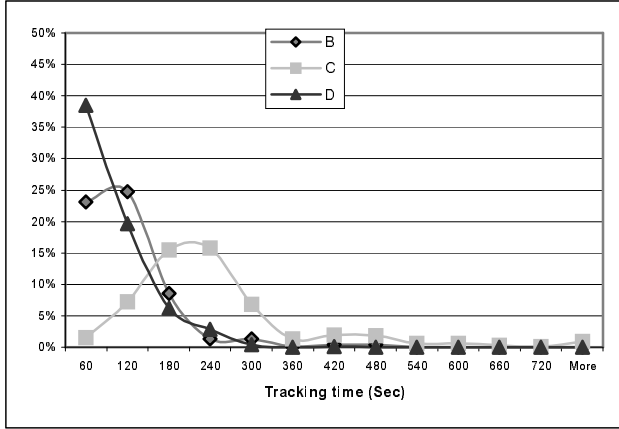


Figure 4: Accurate tracking percentage of each method as a function of the tracking time

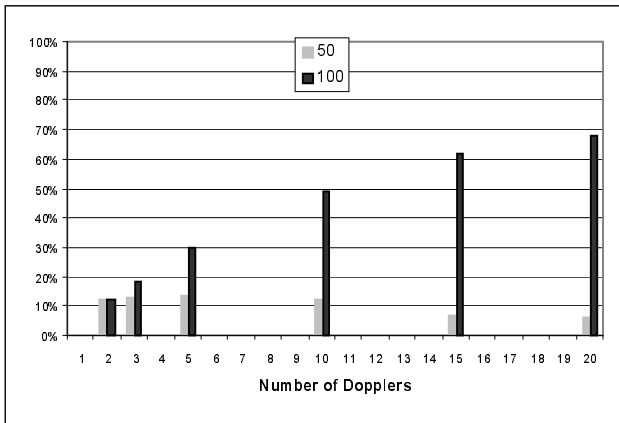


Figure 5: Tracking percentage as a function of the number of Dopplers

Finally, Figure 3 adds further support to our claim that method D demonstrates the best performance by virtue of its ability to identify targets very quickly.

5.2 Number of Dopplers comparison

We examined the effect of the number Dopplers on performance. Figure 5 compares tracking percentage as a function of the number of Doppler sensors in the environment. As the number of Doppler sensors increases the percentage of the 100% paths increases while the percentage of targets that the top-level coalition leader could not track decreases. This is significant in that it demonstrates that the system can make good use of additional resources that it might be given.

We can also see that as the number of Doppler sensors increases the 50% probability paths decrease. That may be explained by the fact that 100% paths result from taking into consideration more than one point of view of samples. Using more than one sensor or using the same sensor from different points of view may improve this. Having more radars increases the probability of having more than one Doppler detect the same target.

Figure 6 illustrates that average tracking time decreases as the number of Dopplers increases.

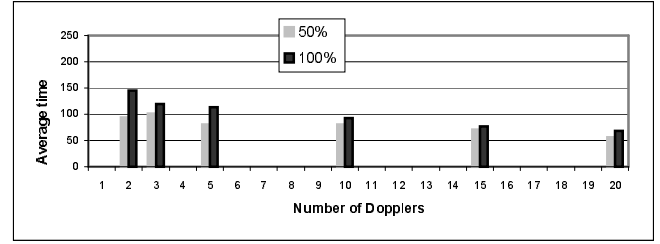


Figure 6: Average tracking time versus number of Dopplers

5.3 Number of targets comparison

The number of targets passing through the controlled area in each simulation was kept at some constant number, N ; we ran additional experiments in which N varied in each simulation. We found that increasing the number of targets did not influence the system's performance. We believe this is because an active sector can distinguish more than one target in a sector.

5.4 Maximum detection range comparison

We tested the influence of the detecting sector area on performance. Given equation (1) we can represent the sensed area by the minimum amplitude. The minimum amplitude may be derived from the maximum detecting range therefore the detection area is proportional to the maximum detection range.

$$A_{Min} = \frac{k}{R_{Max}^2}, Area = \int_{-\pi}^{\pi} \sqrt{\frac{k \cdot e^{-\frac{-(\theta-\alpha)^2}{\sigma}}}{A_{min}}} = R_{max} \cdot \int_{-\pi}^{\pi} \sqrt{e^{-\frac{-(\theta-\alpha)^2}{\sigma}}}$$

The basic setting uses Dopplers with detection range of 200 meters. We compared the basic setting to similar settings with Doppler's detection range of 50, 100 and 150 meter. Figure 7 indicates that as the maximum range increases the tracking percentage increases polynomially. One can also see that the detection area is an influencing factor. Using Dopplers with detection radius of 50 meters results in a tracking accuracy of 5% of the targets. Increasing the detection area four times results in 68% of the targets to be detected accurately.

The average tracking time is also influenced by the change in the maximum detection range. Figure 8 depicts the change of the average detection time as a function of the changes in the detection area. As the maximum radius of detection increases the tracking average time decreases polynomially. While 40% of the Dopplers with 200 meters of maximum detection range, only 22% of the targets are accurately detected when using Dopplers with maximum detection range of 150 meters.

Figure 9 shows that, for Dopplers with smaller area, the productive period of time shifts from up to 60 seconds to the period of 60 to 120 seconds in the case of maximum detection range of 100 meters and 120 to 180 seconds for 50 meters.

6. Related work and summary

Agent and organizational designs typically vary along a number of dimensions. Agent architectures can vary from very simple designs, usually deployed in large numbers and crafted so that some set of desirable global properties might emerge [12], to more complex designs usually based on some variation of the BDI agent model in which agents behaviors are connected to team roles and commitments.

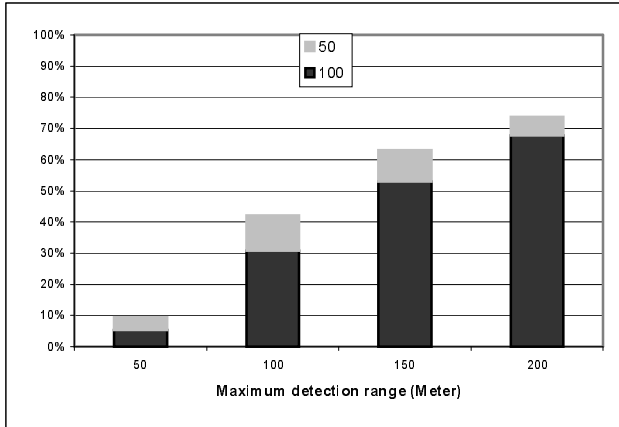


Figure 7: Target tracking percentage as a function of the maximum detection range

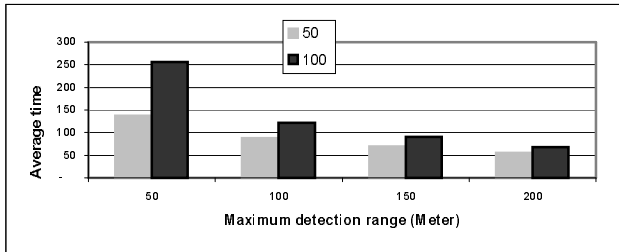


Figure 8: Average target tracking time as a function of the maximum detection range.

On the organizational side, societal structures are usually either prescriptive in design - in the sense that a particular structure is assigned to a team - or represent the structural outcome of agent interactions. In the latter case, organizations can be identified as consequences of role persistence in team interactions [4]. Prescriptive structures might derive from task decomposition structures or segmentations based on capabilities, commitments, spatio-temporal divisions, or *a priori* authority structures.

The benefits of hierarchical organizations have been argued by many. So and Durfee draw on contingency theory [11] to examine the benefits of a variety of hierarchical organizations; they describe a hierarchically organized network monitoring system for task decomposition [14,13]. Corkill and Lesser investigate various team organizations in terms of capabilities which agents within certain *interest areas* were well-equipped to handle [2]. Wagner and Lesser consider the role that knowledge of organizational structure can play in control decisions [16].

DDM falls in the simple agent and prescriptive organizational design category. DDM differs from the above efforts in its use of a hierarchy as: (1) an efficient way of reducing the uncertainty associated with solutions to information collection tasks based strictly on local information by instead combining partial local solutions into a global solution, and (2) a structure for building more fault tolerant systems consisting of large teams of simple agents. We demonstrated the performance advantage of hierarchical design through extensive experimentation.

In future work, we hope to extend DDM with structures that can adapt [8] and consider more complex hierarchies [14].

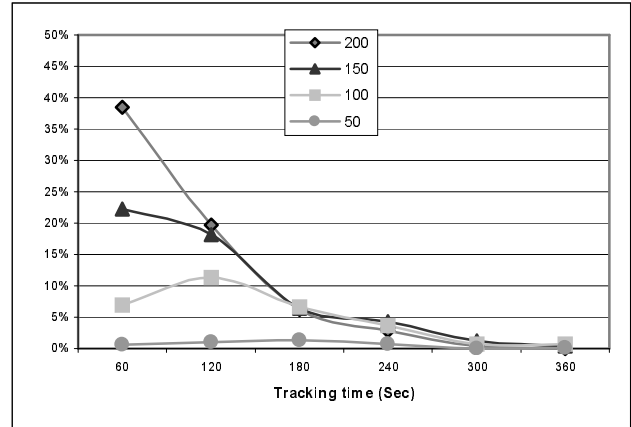


Figure 9: Accurate tracking percentage as function of tracking time. Maximum detection range comparison.

7. References

- [1] ANTS Program Design Document, unpublished.
- [2] D. Corkill and V. Lesser, "The use of meta-level control for coordination in a distributed problem solving network," IJCAI 1983.
- [3] Richard J. Doviak and Dusan S. Zrnic. Doppler radar and weather observations. Orlando, Academic Press, 1984.
- [4] E. Durfee, V. Lesser, and D. Corkill, "Coherent cooperation among communicating problem solvers," *Readings in DAI*, 262-284, 1987.
- [5] R.P. Feynman. The Feynman Lectures on Physics. Addison-Wesley Publishing Company, chapters 12-14, Bombay, India, 1963.
- [6] Fishbane, Paul M. Physics for scientists and engineers, Prentice Hall Upper Saddle River, New Jersey, pages 144-234, 433-462, 1996.
- [7] Gill, Thomas P., The Doppler effect: an introduction to the theory of the effect, London: Logos, 1965.
- [8] T. Ishida, L. Gasser, and M. Yokoo, "Organization self design of production systems," *IEEE Transactions on Knowledge and Data Engineering*, 4(2):123-134, 1992.
- [9] Kerut, Edmund Kenneth, Handbook of echo-doppler interpretation, Amrmonk, N.Y.: Futura Pub. Co., 1996.
- [10] Pöss, Christian Doppler in Banska Stiavnica, in *The Phenomenon of Doppler* (Prague, 1992), 55-62.
- [11] R. Scott, "Organizations: Rational, Natural and Open," Prentice-Hall, 1992.
- [12] O. Shehory, S. Kraus and O. Yadgar, "Emergent cooperative goal-satisfaction in large scale automated-agent systems," *Artificial Intelligence Journal*, 110(1), pages 1-55, 1999.
- [13] Young-pa So and Edmund Durfee, "A distributed problem-solving infrastructure for computer network management, International Journal of Intelligent and Cooperative Information Systems, 1992.
- [14] So, Y., and Durfee, E.H., "Designing Tree-Structured Organizations for Computational Agents," *Computational and Mathematical Organization Theory*, 2(3), pages 219-246, 1996.
- [15] Leen-Kiat Soh and Costas Tsatsoulis. Reflective Negotiation Agents for Real-Time Multisensor Target Tracking, IJCAI-01, Vol. 2, 2001.
- [16] T. Wagner and V. Lesser, "Relating Quantified Motivations for Organizationally Situated Agents," *ATAL 2000*.