

# Poster Abstract: Sensor Network as a Distributed Manager for Multi-Robot Task Allocation

Maxim A. Batalin and Gaurav S. Sukhatme  
Robotic Embedded Systems Laboratory  
Center for Robotics and Embedded Systems  
Computer Science Department  
University of Southern California  
Los Angeles, CA 90089, USA  
maxim@robotics.usc.edu, gaurav@usc.edu

## Categories and Subject Descriptors

I.2.9 [Artificial Intelligence]: Robotics—*autonomous vehicles, sensors*; C.3 [Computer Systems Organization]: Special-Purpose and Application-Based Systems—*real time and embedded systems*; C.2.4 [Computer Communication Networks]: Distributed Systems—*distributed applications*; J.7 [Computer Applications]: Computers In Other Systems—*command and control, process control, real time*

## General Terms

Algorithms, Design, Experimentation, Management, Measurement, Performance, Reliability

## Keywords

mobile robotics, sensor network, task allocation, distributed

We study Multi-Robot Task Allocation (MRTA) where robots have to be allocated dynamically to tasks. The task schedule is unknown in advance. We approach the problem by using mobile robots and sensor network collaboratively. Sensor network computes task assignments and then guides the robots to tasks efficiently. Moreover, we suggest a solution to the MRTA problem which requires groups of robots to be formed. In addition, we discuss practical aspects in implementing algorithms on physical system.

We study a particular experimental scenario, emergency handling, as an experimental substrate. In our experimental scenario, events in the environment trigger alarms. An alarm is spatially focused, but has temporal extent (i.e. it remains on until it is turned off by a robot). Alarms are detected by nodes in the static network. The task of the team of robots is to turn off the alarms by notionally responding to the emergency signaled by each alarm. This is done by a robot navigating to the location of the alarm which causes

the alarm to shut off. The goal is to minimize the cumulative alarm *OnTime* across all alarms, over the duration of the entire experiment. Each Alarm's *OnTime* is computed as the difference between the time the alarm was turned off by a robot and the time the alarm was detected by one of the nodes of the network.

In our earlier work we worked on Implicit and Explicit Task Allocation. We call Implicit Task Allocation - Distributed In-Network Task Allocation (DINTA). The basic idea of DINTA is that given a set of alarm weight pairs detected by the network, every node in the network computes a suggested direction that a robot should take if in the vicinity of a node. This computation results in a direction which maximizes the net utility of the robot. The concept of weight is abstracted in our work to a single variable and can include a mixture of several parameters like priority, magnitude, time (older alarms should be served first), etc. The ensemble of suggested directions computed at each node is called a Navigation Field. Hence, a single Navigation Field is being computed and the tasks assigned to robots implicitly. It is important to note that above approach makes implicit assignments of tasks to robots, which may result in suboptimal behavior both in terms of time and wasted resources (several robots might pursue the same task). Consider the case when the robots are cluttered in one region (one subfield) and therefore, can all be attracted towards the same alarm or simply ignore other alarms, depending on the implementation. To address these issues we currently experimenting with an Explicit approach - Multi Field Distributed In-Network Task Allocation (DINTA-MF). The approach is based on maintaining multiple navigational fields, one for every alarm (task) at the same time and assigning those fields to different robots using greedy policy. In other words, every node in the environment computes optimal direction that the robot should follow for every alarm in the environment. The tasks are then assigned explicitly to robots. Both approaches rely on the concept of Distributed Value Iteration. In order to compute a suggested direction of maximum utility, every node in the network updates its utility locally, queering neighbors for their corresponding utilities. The update equations are:

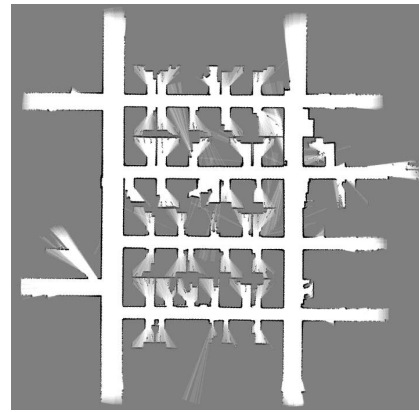
$$V_{i+1}(s) = C(s, a) + \max_{a \in A(s)} \sum_{s' \in S-s} P(s'|s, a) \times V_i(s') \quad (1)$$

$$\pi(s) = \arg \max_{a \in A(s)} \sum_{s' \in S-s} P(s'|s, a) \times V(s') \quad (2)$$

where  $V$  is the utility value,  $C(s, a)$  is the cost associated with an action,  $P(s'|s, a)$  is the transition probability of arriving to node  $s'$  if an action  $a$  was taken at node  $s$ ,  $\pi(s)$  is the policy, or direction that the node  $s$  is going to suggest for the robot in the vicinity. The idea is that every node in the network updates its value and computes the optimal task assignment (navigation action) for a robot in its vicinity on its own. Note that the values of neighboring nodes are needed as well, hence, the node queries its neighbors for corresponding values. Our current work is focusing around improving and generalizing Task Allocation algorithms presented above and implementing them not only in simulation but also in physical system. Consider scenario when the system is posed with several problems (tasks) that require several resources (homogeneous or heterogeneous) to be allocated for a single task. DINTA and DINTA-MF solve the task allocation problem when only one resource is required.

Currently we developing system which is able to group robots together and assign groups to tasks rather than individual robots. In this case general problems involving task allocation can be solved. We name this system - Distributed Manager (DM). The general idea of DM is to use a static network and mobile robots cooperatively. The network provides a sensor that is stretched over the environment and thus widens the range of applications for groups of robots that do not cover the whole environment - can't be everywhere at the same time. Thus, an alarm can be detected even though no robot is within sensor range. In addition, mobile robots can communicate through the static network even if they are not within communication range of each other. The other benefit of using the network is distributed computation. First, there is no redundant computation (on each separate robot). Second, since every node of the network updates its state based only on the state of its neighbors and robots in the vicinity, the system is scalable. Third, utilities are computed in the network distributively and propagated from the goal state. Another benefit is that the robots used can be very simple since they do not need to localize and map the environment - they navigate by listening to the suggestions from the sensor network. Tasks in the system are represented as  $(nodeID, weight, hopcount, (R1, R2, \dots, Rn))$ , where  $nodeID$  is the node posted the task,  $weight$  is relative importance,  $hopcount$  is the distance from current node to the node that posted the task and set  $(R1, R2, \dots, Rn)$  is the set of resources that are required by a posted task. The task allocation process occurs in time intervals called *decision epochs*. Only tasks arrived during current *decision epoch* are considered for assignment at the same time. Once the resources were allocated, they are committed to the task until it is over. These rules insure optimal performance and avoid deadlocks (avoid resource starvation).

In general the structure of the proposed DM system allows the sensor network to serve as a more general, multi purpose infrastructure. This could enable, for example, solutions to problems requiring heterogeneous groups of robots. Imagine a scenario on a construction site which requires cooperation of two groups of robots - transporters and builders. Transporters concentrate on delivering the materials to several piles while builders choose the type of material they need



**Figure 1: Map of the environment.**

at the moment from a corresponding pile and continue construction. Thus, a transporter robot might use the network to find the shortest path towards the material storage or towards the pile that requires certain material the most (using proposed Distributed Value Iteration algorithm for example). While the builder robot would be directed towards a pile with required material or towards another builder needing assistance.

The system is currently under development and testing, but preliminary results show that it works reliably with good performance. Concurrently we are working on physical implementation of above mentioned algorithms at Intel Research facilities in Hillsboro, Oregon. Our physical platforms are Pioneer 2DX mobile robot equipped with 180 laser range finder, Amigobot mobile robot equipped with ring of 8 sonars and camera, and a set of 20 Mica2 motes. The sensor network is embedded into the environment of Figure 1. Every node of the sensor network is monitoring its light sensor. If high light intensity is detected by a node, this node sends out a packet containing its id, the importance of the event and has a hop count field so that on reception other nodes would determine the relative distance to the goal node appropriately. Since the message protocol and radio itself sometimes drop packets on Mica2 motes, the algorithm for computing the optimal direction was adapted. Hence, instead of querying the neighbor for data in Distributed Value Iteration algorithm, every node sends out its data to the neighbors as it becomes available. When the assignment field (navigation field) is computed, every node starts to emit suggested direction (assignment direction) for the robot to follow and the id of the next node that the robot would have to switch to and start listen to that nodes suggestions. Since the experiments are taking place indoors, in the cubicle space the compass is useless and hence we adapted our navigation algorithm for robots as well. Hence, upon receiving the navigation assignment message the robot moves in suggested direction and switches to a different node based on its id, observed landmarks (intersections, corners, etc.) and signal strength threshold. These three parameters in combination allow the robot to navigate and switch to a correct node at correct time reliably. Further experiments are currently conducted which would allow us to test reliability and resiliency of the approach to changing conditions of the environment.