In *IEEE International Conference on Robotics and Automation*
pp. 158-164, New Orleans, LA, April 26 - May 1, 2004

Using a Sensor Network for Distributed 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
Email: maxim@robotics.usc.edu, gaurav@usc.edu

***Abstract*— We present a Multi Field Distributed In-network Task Allocation (DINTA-MF) algorithm for online multi-robot task allocation (OMRTA) where tasks are allocated explicitly to robots by a pre-deployed, static sensor network. The idea of DINTA-MF is to compute several assignment fields in the sensor network and then distributively assign fields to different robots. Experimental results with a simulated alarm scenario show that our approach is able to compute solutions to the OMRTA problem in a distributed fashion and arguably in an optimal way. We compared DINTA-MF with a simpler implementation (DINTA) which uses one assignment field. The data show that DINTA-MF outperforms DINTA as the number of robots increases.**

I. Introduction

We are motivated by a particular application of wireless sensor networks, namely, multi-robot task allocation (MRTA). The MRTA problem has been well-studied in the robotics community [1], and is simply stated as the problem of allocating tasks to robots. Of particular interest is the online version of the problem (OMRTA), where tasks in the environment are geographically and temporally spread, and robots need to visit task locations to accomplish task completion. The problem is to assign tasks to robots optimally in an online fashion. Previous MRTA approaches in the robotics community have focused on performing the task allocation computation on the robots or at some centralized location external to the robots. All the sensing associated with tasks and robot localization is typically performed on the robots.

In recent work [2], we have proposed an alternative strategy based on the interaction between a sensor network and mobile robots. Tasks, upon arrival, are allocated implicitly to robots by a pre-deployed, static sensor network. In prior work [3], [4] we have developed an algorithm for the deployment, and maintenance of such a static network by robots. We have also developed algorithms for exploration and navigation [3], [4] where robots use the deployed network to efficiently explore an unknown environment and navigate to a designated goal. In [2], we assume the network is pre-deployed (through means outlined in [3], [4]), and robots have to perform spatially and temporally distributed tasks efficiently. Our solution [2] was to allow the process of task allocation to occur in the static network through distributed computation and implicit assignment of robots to tasks. We termed the approach described in [2] DINTA: Distributed In-Network Task Allocation. The basic idea of DINTA is that given a set of tasks detected by the network, every node k in the network computes a suggested motion direction for a robot if its in the vicinity of k. The ensemble of suggested directions computed over all nodes is called a Navigation Field. An adaptive distributed value iteration algorithm is used to compute the navigation field.

In this paper, we propose a variant of DINTA, where multiple navigational fields (one for every task) are maintained in the network at a given time. Fields are assigned to robots using a greedy policy. We call this approach DINTA-MF (multiple fields). The difference between this approach and the previous approach is that in DINTA-MF every network node computes the direction that the robot should follow in its locality for every task in the environment. Like our previous work on DINTA, we consider an online task assignment problem, in which tasks need to be assigned to robots in real time and the distribution of tasks' arrival is not known a priori.

Our work is broadly situated at the intersection of mobile robots and sensor networks. The underlying principle in interaction between the network and robots is: the network serves as the communication, sensing and computation medium for the robots, whereas the robots provide actuation (mobility), which is used, among other things, for network deployment, repair, and other tasks.

We study a particular experimental scenario, emergency handling, as an experimental substrate. In prior work [5], we have used a similar scenario to study the role of opportunism vs. commitment in MRTA. 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 sensor nodes. 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 *On Time* across all alarms, over the duration of the entire experiment. Each alarm's *On Time* 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.

We make the following assumptions:

1) The sensor network is predeployed in the environment (one could use the algorithm in [4] for automatic deployment using robots or rely on a manual deployment).
2) In addition to deploying the network nodes, the deployment algorithm also computes the distributions of transition probabilities $P(s'|s,a)$ from network node s to s', when the robot executes action a [3]. The appropriate distributions are stored on corresponding nodes.
3) An alarm requires at least one robot to service it. To turn off an alarm, a robot needs to appear in its vicinity. Thus, the handling of the alarm is purely notional since that is not our focus here.
4) The nodes of the sensor network are time synchronized (high precision is not required). One of several existing techniques may be used for this. As an example see [6].

For completeness, we enumerate what we do not assume:

1) The robots do not have a pre-decided environment map or access to GPS.
2) The environment is not required to be static.
3) The robots do not perform localization or mapping.

Our key result is that the multi-field approach (DINTA-MF) statistically outperforms our earlier implementation (DINTA [2]). Further, DINTA-MF arguably provides an optimal solution to an online assignment problem in terms of time and assigned resources.

There are many applications of MRTA including security, monitoring, and urban search and rescue (USAR) in the aftermath of a natural or man-made disaster (e.g. building rubble due to an earthquake or other causes). Further, the ability of a sensor network to assign tasks to different robots, thus serving as a multi-purpose infrastructure, enables solutions to problems requiring heterogeneous groups of robots. Imagine a scenario on a construction site which requires cooperation of two distinct groups of robots - transporters and builders. Transporters concentrate on delivering the materials to several piles while builders choose the type of material they need from a corresponding pile and continue construction. Thus, transporter robots would be guided (tasked) along a shortest path towards the material storage area or towards the pile that requires certain materials the most. While the builder robots would be directed towards a pile with required materials or towards another builder needing assistance. In other words, the network can be used as a distributed multi-functional manager. Note also that even though we study task assignment problem in the context of mobile robot task allocation, the proposed system can be applied for the general online task assignment problem where the resources are different from robots (e.g. people trying to get outside of the building would be guided (tasked) to the closest exits, etc.).

II. Related Work

Our work is closely related to the body of literature on using markers to aid mobile robot navigation. This idea has received attention in coverage and exploration [3], [4], [7], [8], [9], and navigation [3], [10], [11]. Ant-like trail laying algorithms [10], [9] consider a special case of the *marker* deployment approaches - when the distance between the two consecutive *markers* is small. Therefore a *trail* is formed that the robots can follow and cover the environment and/or navigate. In these cases, no inter-marker communication is necessary, indeed the markers are passive 'read-only' devices.

In [7], [8] the problem of graph coverage using a few *markers* is considered. In both cases the authors study the problem of *dynamic* single robot coverage of an environment consisting of nodes and edges (a graph). The key result was that the ability to tag a limited number of nodes (in some cases only one node) with unique *markers* dramatically improved the cover time.

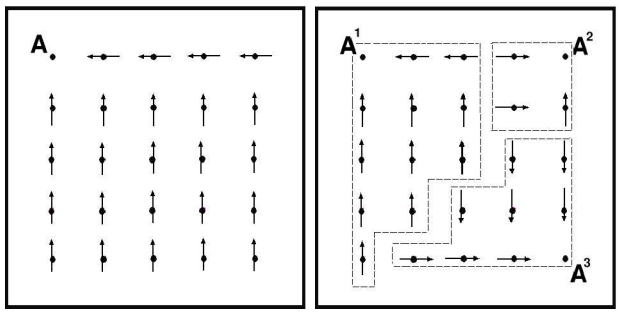
The problem of multi-robot task allocation (MRTA) has received considerable attention. For an overview and comparison of the key MRTA architectures see [1], which subdivides MRTA architectures into behavior-based and auction-based. For example, ALLIANCE [12] is a behavior-based architecture that considers all tasks for (re)assignment at every iteration based on robots' utility. Utility is computed by measures of acquiescence and impatience. Broadcast of Local Eligibility [13] is also a behavior-based approach, with fixed-priority tasks. For every task there exists a behavior capable of executing the task and estimating the utility of robot executing the task. Auction-based approaches include the M+ system [14] and Murdoch [15]. Both systems rely on the Contract Net Protocol (CNP) that makes tasks available for auction, and candidate robots make 'bids' that are their task-specific utility estimates. The highest bidder (i.e., the best-fit robot) wins a contract for the task and proceeds to execute it.

DINTA-MF (and DINTA), differs from the above MRTA approaches in the following ways:

1) DINTA-MF relies on a static *network*, and communication, sensing and computation are distributed.
2) The utilities of task assignments are propagated and computed by the network based on purely local communication between the network nodes.
3) The system does not require mobile robots to be within communication range of each other. The network is used for propagating messages between the robots.
4) The system does not place a limitation on the number of robots. There is no computation or communication overhead associated with increasing the number of robots.
5) The system does not require one robot to recognize another robot.

III. Task Allocation: Offline vs. Online

The Task Allocation (TA) problem has two major subdivisions: Offline and Online. Offline TA is the problem of assigning resources (robots) to different tasks (alarms) if the tasks' information such as arrival time distribution, tasks' weight or priority, etc. is known *a priori*. The assignment process is thus offline. An offline TA problem, in its most



(a) One alarm

(b) Three alarms; A^1, A^2 and A^3

Fig. 1. Examples of navigation field computed by DINTA.

general form, is equivalent to the NP-Complete conjunctive planning problem [16].

We focus on the other version of the problem - Online Task Allocation. In online TA, the information about the tasks becomes available only upon arrival and hence has to be computed in real time. Therefore, the task assignment occurs in *decision epochs*. A *decision epoch* is a period of time during which only the alarms which have arrived are considered for assignment. It is shown in literature [17] that the optimal solution to online TA assigns the tasks in a greedy fashion. Also [17] show that the greedy online TA solution is within a bounded limit of an optimal solution obtained by offline TA. In general, increasing the *decision epoch* to infinity turns the online TA into offline TA problem. We note that several real-life applications involving mobile robots in dynamic environments are naturally online problems.

IV. The Basic Approach: DINTA

The basic idea of DINTA (see [2] for details) is that given a set of alarm-weight pairs (a_i, w_i) detected by the sensor network, every node k in the network computes a suggested direction that a robot should take if in the vicinity of k. This computation results in a direction which maximizes the net utility of the robots. The weight w_i is an abstraction, which is a scalar representation for several parameters like priority, magnitude, time (older alarms should be served first), etc. The ensemble of suggested directions computed over all nodes is called a Navigation Field. An adaptive distributed value iteration algorithm is used to compute the navigation field. An example of a navigation field for one and three alarms is shown in Figure 1.

It may be noted that a parallel approach for the construction of a navigation field has been proposed in the sensor network literature [11]. Instead of value iteration [11] uses potential fields and the hop count to compute the magnitude of the directional vectors.

The DINTA approach has two subsystems - *Coverage/Exploration* and *Alarm Response*. If no alarms are detected, the system operates in *Coverage/Exploration* [3], [4]

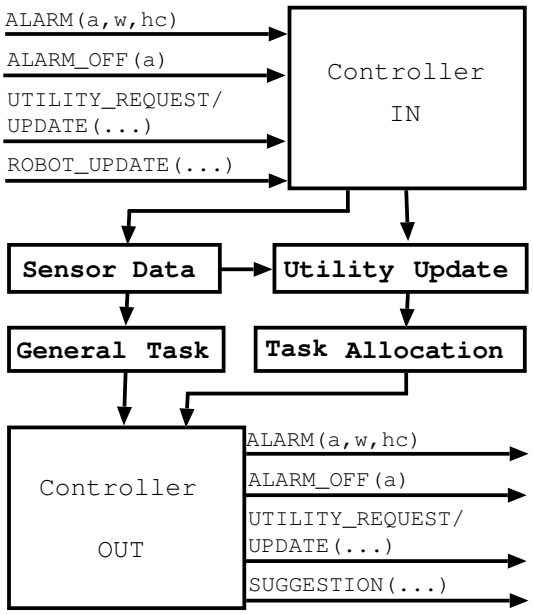


Fig. 2. Node Architecture in DINTA.

mode. In this mode, the navigation field computed by the network, causes the robots to patrol the environment. If, on the other hand, an alarm is detected, the system switches to the *Alarm Response* mode where the navigation field computed by the network guides the robots to turn off alarms, thereby implicitly solving the MRTA problem. The *Coverage/Exploration* mode is the same in DINTA and DINTA-MF.

Figure 2 shows the data flow on a network node. *Alarm Response* mode proceeds as follows. If a node receives an *ALARM* message with identification a of the node that detected the alarm, weight w (estimation of the alarm's importance) and *hop count* h (estimation of how far away node a is), the alarm is placed on the list L of currently active alarms according to its utility V (`Utility Update` block). We define the utility as the ratio $V = \frac{w}{h}$. Note that DINTA maintains only one navigation field . Thus the utility is used by each node to determine which alarm it should compute a direction for. Low hop counts equate to high utility so all things being equal, a node will maintain a direction command for a nearer alarm compared to one further away.

Every node maintains a *current alarm* variable, which is the element of L with largest utility. If the *current alarm* changes, the `Task Allocation` block computes a new task assignment for a robot (discussed next) and reroutes the alarm message with incremented hop count to neighboring nodes. In the global perspective, prioritizing between the tasks according to their utility value results in creation of multiple superimposed navigation fields (for example three alarms case of Figure 1). Note that if `Sensor Data` block indicates that an alarm is detected by the node itself, then the node initiates a message *ALARM(thisNodeID, w, 0)*. The `General Task` block represents the *Coverage/Exploration* subsystem, in case L is empty.

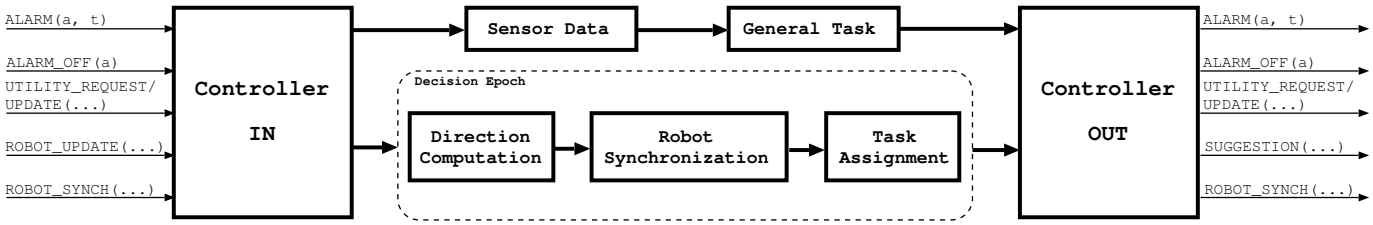


Fig. 3. Node Architecture in DINTA-MF.

The direction computation proceeds according to equations 1 and 2 and follows a distributed value iteration Algorithm. A thorough discussion of the approach can be found in [2].

It is important to note that DINTA does not make explicit assignments of tasks to robot or specific robot subgroups, 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.

V. The Multi-Field Approach: DINTA-MF

DINTA-MF is based on maintaining multiple navigational fields, one for every alarm at the same time and assigning those fields to different robots using a greedy policy. In other words, every node in the environment computes the 'optimal' direction that the robot should follow (when in the vicinity of that node) for every alarm in the environment. Figure 3 shows the data flow on a network node.

A. Philosophy

DINTA-MF uses 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 alarm (the goal node). 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.

The DINTA-MF approach, like DINTA, has two subsystems - *Coverage/Exploration* and *Alarm Response*. If no alarms are detected, the system operates in *Coverage/Exploration* mode. In this mode, the navigation field computed by the network, causes the robots to patrol the environment. For details on how that is enabled see [3], [4]. If, on the other hand, an alarm is detected, the system switches to the *Alarm Response* mode where navigation fields are computed by the network, which guide the robots to turn off alarms, thereby implicitly solving the MRTA problem.

B. Alarm Response

The following describes behavior of a node in the sensor network. If a node receives an *ALARM* message with identification a of the node that detected the alarm, the system time t when the alarm was detected by node a and hop count h from this node to the node detected the alarm (node a), the alarm is placed on the list L_{UA} of currently active unassigned alarms according to its time t.

This portion of the system executes in *decision epochs*. As described above, a *decision epoch* is a period of time during which only recently arrived alarms are considered for assignment. For purpose of clarity and ease of explanation we consider a *decision epoch* to be equal to one alarm, but easily can be generalized to n alarms or time units. Hence, in the experiments described here, we assign one alarm at a time. For a given *decision epoch*, if the list L_{UA} is not empty (i.e. it contains active unassigned alarms) and there are unassigned robots in the vicinity, the `Direction Computation` block starts to compute the assignment direction for an alarm of smallest time t in the list. List L_{CD} contains direction assignments computed for alarms of list L_{UA}.

A node decides which robot should be assigned to a particular alarm based on where robots are relative to the alarms. In a given *decision epoch* every node that has robot in its vicinity sends out a ROBOT-SYNCH-MESSAGE that contains robot's id, alarm id, and the hop count to the alarm. A node processes such responses in `Robot Synchronization` and passes this information to the `Task Assignment` block. Given shared information about the alarms and robots relative positions to these alarms the `Task Assignment` block of every node assigns the robot with shortest distance. Ties are broken in favor of robots with smaller IDs. Note also that in order to guarantee that the system is in the same *decision epoch* (i.e. computes assignments for the same tasks) a time-synchronization mechanism is needed. The `General Task` block implements the coverage/exploration algorithm described in [3], [4] in case L_{UA} is empty.

The task allocation problem for *emergency handling* can be formulated as guiding robots towards a specific goal node

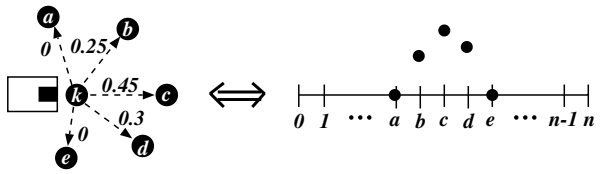


Fig. 4. An example of a discrete probability distribution of node k for direction (action) "East"(i.e. right).

(closest to alarm). Hence, the problem can be considered as the problem of navigation. A node computes the direction assignments in the `Direction Computation` block as follows. We assume that the network is deployed and every node has a discrete probability distribution of the transitional probability $P(s'|s_C, a)$ (probability of arriving at node s' given that the robot started at node s_C and commanded an action a). The reader is referred to [3] for detailed discussion on how such distributions can be obtained. Figure 4 shows a typical discrete probability distribution for a node per action (motion direction). Note that in practice the probability mass is distributed around neighboring nodes and zero otherwise.

Note that the node the robot transitions to depends only on the node closest to the robot and the action the robot takes. Thus the navigation problem is modeled as a Markov Decision Process [18]. To compute the best action at a given node, value iteration [19] is used on the set of nodes $S - s_g$, where s_g is the goal node. The general idea behind value iteration is to compute the values (or utilities) for every node and then pick the actions that yield a path towards the goal with maximum expected value. The value is incrementally computed:

$$V_{t+1}(s) = C(s,a) + \max_{a \in A(s)} \sum_{s' \in S-s} P(s'|s,a) \times V_t(s') \quad (1)$$

where $C(s,a)$ is the cost associated with moving to the next node. Usually the cost is chosen to be a negative number which is smaller than $\frac{-(minimalreward)}{k}$, where k is the number of nodes. The rationale is that the robot should 'pay' for taking an action (otherwise any path that the robot might take would have the same value), however, the cost should not be too big (otherwise the robot might prefer to stay at the same node). Initially the value of being at the goal node is set to a large number and of the other nodes to 0. Given the values, an *action policy* is computed for every node s as follows:

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

The standard value iteration algorithm assumes central computation. However its not difficult to implement a distributed version. 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. Once the *current alarm* has been changed, every node starts the computation of the optimal task assignment by updating values according to Equation 1. Note that the values of neighboring nodes are needed as well, hence, the node queries its neighbors for corresponding values. Note that value iteration can be considered as a form of Dynamic Programming. [20] showed how to compute general Dynamic Programming problem distributively.

After the values are computed, every node computes an optimal policy for itself according to Equation 2. Neighboring nodes are queried once again for the final value. The computed optimal action is stored at each marker and is sent as a *SUGGESTION* message, to any robots in the vicinity.

The action policy computation is done only once per alarm, and does not need to be recomputed. The value update equations have to be executed until the desired accuracy is achieved. For practical reasons the accuracy in our algorithm is set to 10^{-3}, which requires a reasonable number of executions of the value update equation per state (approx. 20) and thus, the list of values that every node needs to store is small (20). Since the computation and memory requirements are small it is possible to implement this approach on the real nodes that we plan to use (the Mote [21]).

Note that if neighbors of all nodes are known exactly (per every direction node has at most one neighbor), then $P(s'|s,a) = 1$. Hence, equations 1 and 2 reduce to maximization of utilities of neighboring nodes only. In this case the system would converge after a single iteration.

VI. Simulation Experiments

In our experiments we used the Player/Stage [22], [23] simulation engine populated with simulated Pioneer 2DX mobile robots equipped with 180° field-of-view planar laser range finders (used for obstacle avoidance), wireless communication and a mote base station (to communicate with the motes, used as network nodes). A network of 25 Motes was predeployed in a test environment. The communication range of motes and robots was set to approximately 4 meters. The task of the team of robots is to serve alarms by navigating towards the point of alarm and minimize the cumulative alarm *On Time*. Each alarm's *On Time* is computed as the difference between the time the alarm was served by a robot and the time the alarm was detected by one of the nodes of the sensor network. We conducted experiments in an environment of $576m^2$ with robot group sizes varying from 1 to 4, 10 trials per group. For experiments the schedule of 10 alarms was drawn (time-wise) from a Poisson distribution, with uniformly distributed nodes that detected the alarm. The parameter of Poisson distribution was set to $\lambda = \frac{1}{60}$, which means that the expected number of alarms is 10 in 600 seconds.

The implementation of the proposed approach in simulation proceeds as follows. If there are no alarms detected in the environment, then the robots execute exploration algorithm of [3], [4]. If an alarm is detected, the network computes task assignments (navigation fields in our case). Once the tasks are computed at every node, the robots change from *EXPLORATION* to *ALARM* mode, and traverse the directions suggested by the network. When a robot reaches an alarm

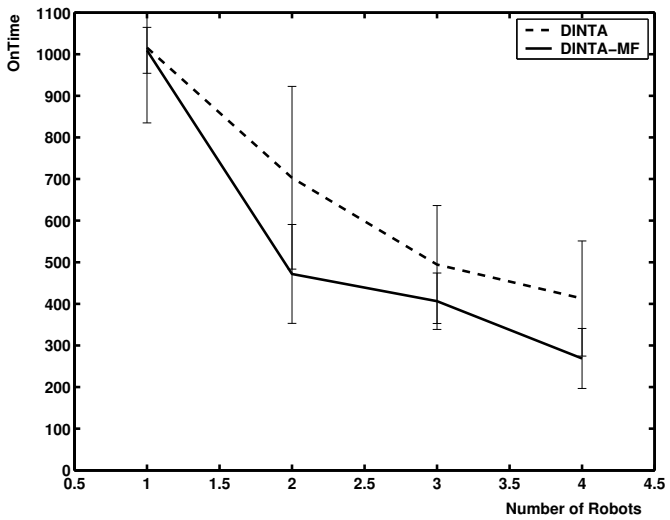


Fig. 5. Experimental comparison between DINTA-MF and DINTA

Fig. 6. Mobile robot and a Mote in experimental setting.

node, the robot injects an ALARM-OFF message into the network. Note that if there are multiple alarms active at the same time, several navigation fields will be produced. Figure 5 shows the *On Time* comparison for DINTA and DINTA-MF. In order to test the similarity between the pairs of datasets for DINTA and DINTA-MF, we ran the T-Test. The T-Test produced the following p values for correxponding data points: 0.9139 for the first pair of points which means that they are similar, 0.0081, 0.0897, 0.091 for the last three pairs of points, which means that they are different. Note that with one robot the performance of both algorithms is about the same, which is due to the fact that both approaches reduce to the same solution. Although, despite the fact that environment becomes saturated with robots, the proposed approach outperforms its predecessor as the number of robots in the team increases. The reason behind this is that every node in the network computes alarm assignment for every alarm and hence all unassigned robots are assigned an alarm, which is not necessarily the case with one assignment field. Note also that DINTA-MF does not waste resources (robots), whereas in DINTA, several robots can pursue the same alarm.

Note that space and time requirements for DINTA-MF are linear in the number of alarms, which makes it realistic for implementation on our target node platform (the mote). In addition we plan to extend the proposed approach to allow online grouping of robots for more complex tasks.

VII. Conclusion and Future Work

In this paper we introduced DINTA-MF: Multi Field Distributed In-Network Task Allocation for solving the Online MRTA (Multi Robot Task Allocation) problem. DINTA-MF allows us to combine the benefits of a sensor network with mobility and functionality of robots. The system computes task assignments distributively in-network while, at the same time, providing a virtual sensor and communication device that 'extends' throughout the whole environment and has obvious benefits over traditional OMRTA approaches. The fundamental assumption, though, is the existence of the sensor network on which robots can rely.

We compared DINTA-MF with an earlier implementation, which relies on maintaining a single navigation field consisting of several subfields. The experimental data show that DINTA-MF outperforms DINTA. The difference in the *On Time* metric is not large though. The reason is that the experimental environment is not large and the occurrence of alarms is rather infrequent. Hence the environment and conditions of the experiment are better suited for DINTA. Another advantage of using DINTA-MF is that it does not waste resources, whereas in DINTA, several robots can pursue the same alarm. In general, DINTA-MF proposes a solution which can handle alarms (tasks) of high frequencies, represents a multi purpose distributed manager that can solve a large variety of problems

There are several advantages in using DINTA-MF as opposed to other MRTA approaches. The sensor network allows robot to detect a goal (alarm) even though the alarm is not in robot's sensor range. In addition, mobile robots can use sensor network to relay messages if they are not in the communication range of each other. One of the other benefits of using DINTA-MF is distributed in-network computation, which 1. avoids redundant computation by updating the state of a node based only on the state of its neighbors and robots in the vicinity (scalability), 2. computes utilities in the network distributively and propagates from the goal state (alarm). Another benefit is the ease of determining relative distance to the goal (for determining utilities) by considering hop counts from the goal node. Note also that robots implementing DINTA-MF can be quite simple - they do not need to localize and map the environment - they can navigate by listening to the suggestions from the sensor network.

In future work we plan to extend the current implementation of DINTA-MF to allow assignment of tasks requiring groups of, potentially heterogeneous, robots, which would allow more complex task assignments as well as group formations. We

also plan to conduct further experiments both in simulation and hardware in varying environments, with tasks of varying complexity, requiring different numbers of robots. The system would have to assign not only a task, but also combine robots in a group if a task requires participation of several robots. We have completed a set of initial real hardware experiments. Figure 6 shows Pioneer 2DX mobile robot and a mica 2 mote during one of the experimental trials. In our experiments a sensor network of mica 2 motes was deployed in a cubicle-like environment. The goal was to compute a navigation field and navigate the robot towards the node that detected the alarm. We conducted 50 experimental trials for 5 different goal nodes. Each trial was successful. Currently we are working on further hardware experiments with multiple goal nodes and distributed task assignment algorithm (DINTA-MF). Our preliminary results show that the system is compact enough to fit in our target sensor network node platform (the Mica2 mote). Further experiments are conducted to check the reliability of the system and possible implementation for real-world applications.

VIII. Acknowledgments

This work is supported in part by NSF grants ANI-0082498, IIS-0133947, and EIA-0121141.

References

[1] B. Gerkey and M. J. Mataric, "Multi-robot task allocation: Analyzing the complexity and optimality of key architectures," in *To appear in Proceedings of the IEEE International Conference on Robotics an Automation (ICRA03)*, Taipei, Taiwan, 2003.

[2] M. A. Batalin and G. S. Sukhatme, "Sensor network-based multi-robot task allocation," in *To appear in Proc. of IEEE/RSJ Intl. Conf. On Intelligent Robots and Systems (IROS'03)*, Las Vegas, Nevada, 2003, pp. 1939–1944.

[3] ——, "Coverage, exploration and deployment by a mobile robot and communication network," in *The 2nd International Workshop on Information Processing in Sensor Networks (IPSN '03)*, Palo Alto, 2003, pp. 376–391.

[4] ——, "Efficient exploration without localization," in *To appear in Proc. of IEEE International Conference on Robotics and Automation (ICRA'03)*, Taipei, Taiwan, 2003, pp. 2714–2719.

[5] E. H. Ostergard, M. J. Mataric, and G. S. Sukhatme, "Distributed multi-robot task allocation for emergency handling." in *In Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2001, pp. 821–826.

[6] J. Elson, "Time synchronization in wireless sensor networks," Ph.D. dissertation, University of California, Los Angeles, May 2003.

[7] M. A. Bender, A. Fernandez, D. Ron, A. Sahai, and S. Vadhan, "The power of a pebble: Exploring and mapping directed graphs," in *Annual ACM Symposium on Theory of Computing (STOC '98)*, 1998, pp. 269–278.

[8] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, "Robotic exploration as graph construction," in *IEEE Transactions on Robotics and Automation, 7-6*, 1991.

[9] J. Svennebring and S. Koenig, "Trail-laying robots for robust terrain coverage," in *To appear in Proc. of IEEE International Conference on Robotics and Automation (ICRA'03)*, Taipei, Taiwan, 2003.

[10] R. Vaughan, K. Stoy, G. S. Sukhatme, and M. Mataric, "Lost: Localization-space trails for robot teams," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 796–812, 2002.

[11] Q. Li, M. DeRosa, and D. Rus, "Distributed algorithms for guiding navigation across a sensor network," in *The 2nd International Workshop on Information Processing in Sensor Networks (IPSN '03)*, Palo Alto, 2003.

[12] L. E. Parker, "Alliance: An architecture for fault-tolerant multi-robot cooperation." in *IEEE Transactions on Robotics and Automation*, vol. 14, no. 2, 1998, p. 220240.

[13] B. B. Werger and M. J. Mataric, *Distributed Autonomous Robotic Systems 4*. Springer-Verlag, 2000, ch. Broadcast of Local Eligibility for Multi-Target Observation, p. 347356.

[14] S. Botelho and R. Alami, "M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement." in *Proc. of IEEE International Conferenceon Robotics and Automation (ICRA)*, 2000, p. 293298.

[15] B. P. Gerkey and M. J. Mataric, "Sold!: Auction methods for multi-robot coordination." in *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, 2002, p. 758768.

[16] D. Chapman, "Planning for conjunctive goals," *Artificial Intelligence*, vol. 32, pp. 333–377, 1987.

[17] B. Kalyanasundaram and K. Pruhs, "Online Weighted Matching," *J. of Algorithms*, vol. 14, pp. 478–488, 1993.

[18] D. J. White, *Markov Decision Process*. West Sussex, England: John Wiley & Sons, 1993.

[19] S. Koenig and R. G. Simmons, "Complexity analysis of real-time reinforcement learning applied to finding shortest paths in deterministic domains," Carnegie Mellon University, School of Computer Science, Carnegie Mellon University, Pittsburg, PA 15213, Tech. Rep. CMU-CS-93-106, December 1992.

[20] D. P. Bertsekas, "Distributed dynamic programming," *IEEE Trans. Automatic Control*, vol. AC-27, no. 3, pp. 610–616, 1982.

[21] K. S. J. Pister, J. M. Kahn, and B. E. Boser, "Smart dust: Wireless networks of millimeter-scale sensor nodes," *Electronics Research Laboratory Research Summary*, 1999.

[22] B. P. Gerkey, R. Vaughan, K. Stoy, A. Howard, G. Sukhatme, and M. Mataric, "Most valuable player: A robot device server for distributed control," in *IEEE/RSJ Intl. Conf. On Intelligent Robots and Systems (IROS)*, Wailea, Hawaii, 2001.

[23] R. Vaughan, "Stage: a multiple robot simulator," Institute for Robotics and Intelligent Systems, University of Southern California, Tech. Rep. IRIS-00-393, 2000.