In *The Third International Multi-Robot Systems Workshop*
pages 27-38, March 14-16, 2005, Naval Research Laboratory, Washington, DC

SENSOR NETWORK-MEDIATED 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

Abstract We address the *Online Multi-Robot Task Allocation (OMRTA)* problem. Our approach relies on a computational and sensing fabric of networked sensors embedded into the environment. This sensor network acts as a distributed sensor and computational platform which computes a solution to OMRTA and directs robots to the vicinity of tasks. We term this Distributed In-Network Task Allocation (DINTA). We describe DINTA, and show its application to multi-robot task allocation in simulation, laboratory, and field settings. We establish that such network-mediated task allocation scales well, and is especially amendable to simple, heterogeneous robots.

Keywords: Mobile robots, sensor networks, task allocation, distributed

1. Introduction

We focus on the intentional cooperation of robots toward a goal (Parker, 1998). Within such a setting, a natural question is the assignment of robots to sub-goals such that the ensemble of robots achieves the overall objective. Following (Gerkey and Mataric, 2004) we call such sub-goals, *tasks*, and their assignment to robots, the *Multi-Robot Task Allocation (MRTA)* problem. Simply stated, MRTA is a problem of assigning or allocating tasks to (intentionally cooperating) robots over time such that some measure of overall performance is maximized.

We focus on the online version of the problem (OMRTA), where 1. tasks are geographically and temporally spread, 2. a task schedule is not available in advance, and 3. robots need to physically visit task locations to accomplish task completion (*e.g.,* to push an object). Our approach to OMRTA relies on a computational and sensing fabric of networked sensors embedded into the environment. This sensor network acts as a distributed sensor and computational platform which computes a solution to OMRTA and directs robots to the vicinity of tasks. To make a loose analogy, robots are routed from source to destination locations in much the same way packets are routed in conventional networks. We term this, Distributed In-network Task Allocation (DINTA).

There are five advantages to doing the task allocation in this manner:

1 **Simplicity:** Since the task-allocation is done in the network, robots may be very simple, designed specifically for optimal task *execution* (*e.g.,* specialized end effectors) rather than computational sophistication. Further, robots do not need conventional localization or mapping support.

2 **Communication:** Robots are not required to be within communication range of each other. The network is used for propagating messages between the robots.

3 **Scaling:** There is no computation or communication overhead associated with increasing the number of robots.

4 **Identity:** Robots are not required to recognize each other.

5 **Heterogeneity:** Robots may be of different types, and need only a common interface to the sensor network.

In this paper we make the following contributions. We briefly review the details of DINTA[1], and demonstrate its application to a system for spatiotemporal monitoring of environmental variables in nature. We note that while we study the task allocation problem in the context of mobile robots, sensor network-mediated task allocation can also be used in other settings (*e.g.,* in an emergency people trying to leave a building would be guided (tasked) to the closest exits by the network).

2. Related Work

The problem of multi-robot task allocation (MRTA) has received considerable attention. For an overview and comparison of the key MRTA architectures see (Gerkey and Mataric, 2004), which subdivides MRTA architectures into behavior-based and auction-based. For example, ALLIANCE (Parker, 1998) 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 (Werger and Mataric, 2000) 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 (Botelho and Alami, 2000) and Murdoch (Gerkey and Mataric, 2004). 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. All 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 themselves. Our approach relies on a sensor network, which performs event detection and task-allocation computation, allowing robots to be simple and heterogeneous.

3. Distributed In-Network Task Allocation: DINTA

As an experimental substrate, we use a particular stylized monitoring scenario in which robots are tasked with 'attending' to the environment such that areas of the environment in which something significant happens, do not stay unattended for long. We model this using the notion of 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 embedded in the environment. For example in a natural setting, an alarm might be generated in case an abrupt change in temperature is detected requiring inspection of the area by the robot. The task of the team of robots is to turn off the alarms by responding to each alarm. This is done by a robot navigating to the location of the alarm. Once the robot arrives in the vicinity of the alarm, the alarm is deactivated. Thus the robot response is purely notional in that the task the robot performs is to arrive at the appropriate location only. The goal is to minimize the cumulative alarm *On Time* across all alarms, over the duration of the entire experimental trial. Each alarm's *On Time* is computed as the difference between the time the alarm was deactivated by a robot and the time the alarm was detected by one of the nodes of the network.

The basic idea of DINTA is that given a set of alarms (each corresponding to a task) detected by the network (*e.g.,* nodes detect motion, presence of dangerous chemicals, etc.), every node in the network computes a suggested 'best' motion direction for all robots in its vicinity. The ensemble of suggested directions computed over all nodes is called a

Algorithm 1 Adaptive Distributed Navigation Field Computation Algorithm (running on every node).

s - current node (or a state)
S - set of all nodes
$A(s)$ - set of all actions possible from node s
$C(s, a)$ - cost of taking an action a from node s
$P(s'|s, a)$ - probability of arriving at node s' given that the robot started at node s and commanded an action a, stored on node s
$\pi(s)$ - optimal direction that robot should take

Compute Direction(goal_node)
if $s ==$ goal_node **then**
  $V_0 =$ some big number
else
  $V_0 = 0$
while $V_t - V_{t-1} > \epsilon$ **do**
  Query neighbor nodes for their new values V_t
  if received new values V_t from all neighbor nodes s' **then**
    $V_{t+1}(s) = C(s, a) + \max_{a \in A(s)} \sum_{s' \in S-s} P(s'|s, a) \times V_t(s')$
    Update neighbor nodes with new value $V_{t+1}(s)$
Query neighbor nodes for their final values $V(s')$
$\pi(s) = \arg\max_{a \in A(s)} \sum_{s' \in S-s} P(s'|s, a) \times V(s')$

navigation field. In case multiple tasks arrive at the same time, multiple navigation fields (one for every task) are maintained in the network and explicitly assigned to robots. Navigation fields are assigned to robots using a greedy policy.

3.1 Computing Navigation Field

We assume that the network is deployed and every node stores a discrete probability distribution of the transition probability $P(s'|s_C, a)$ (probability of the robot arriving at node s' given that it started at node s_C and was told to execute action a). The reader is referred to (Batalin and Sukhatme, 2004a) for a detailed discussion on how such distributions can be obtained.

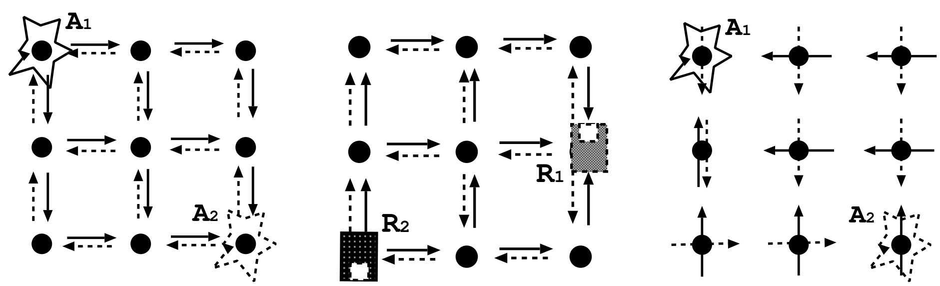
Algorithm 1 shows the pseudo code of the adaptive distributed navigation field computation algorithm, which runs on every network node. We use value iteration (Koenig and Simmons, 1992) to compute the best action at a given node. The general idea behind value iteration is to compute the values (or utilities) for every node and then pick the actions that yield a path towards the goal with maximum expected value. Expected values are initialized to 0. Since $C(s,a)$ is the cost associated with moving to the next node, it is chosen to be a negative number which is smaller than $\frac{-(minimal\_reward)}{k}$, where k is the number of nodes. The rationale is that the robot should pay for taking an action (otherwise any path the robot might take would have the same value), however, the cost should not be too large (otherwise the robot might prefer to stay at the same node).

Next, as shown in Algorithm 1, a node queries its neighbors for the latest utility values V. Once the values are obtained from all neighbors, a node updates its own utility. This process continues until the values do not change beyond an ϵ (set to 10^{-3} in our experiments). After the latest values from all neighbors are collected, a node can compute an action policy π (optimal direction) that a robot should take if it is in the node's vicinity.

In combination, the optimal directions computed by individual network nodes, constitute a global navigation field. Practical considerations for robot navigation using this approach are discussed in (Batalin et al., 2004b).

3.2 Task Allocation

DINTA assigns tasks in *decision epochs* - short intervals of time during which only the tasks that have arrived since the end of the previous epoch are considered for assignment. The following describes the behavior of DINTA in a particular epoch e. Let the network detect two alarms A_1 and A_2 (Figure 1a) by nodes a_1 and a_2 respectively in an epoch e. Both nodes a_1 and a_2 notify the entire network about the new alarms and start two navigation field computations (using Algorithm 1) - one for each goal node. Next consider nodes r_1 and r_2 that have unassigned robots R_1 and R_2 (Figure 1b) in their vicinity. r_1 and r_2 propagate the distances between the unassigned robots and the alarms A_1 and A_2. Four such distances are computed and distributed throughout the network. In the final stage, every node in the network has the same information about the location of alarms and available robots, and distances between the robots and each alarm. Each node in the network can now decide uniquely which navigation field to assign to which robot. Figure 1c shows two navigation fields (one for each robot) generated and assigned to the robots. A robot then simply follows the directions suggested by network nodes.



(a) Phase 1. (b) Phase 2. (c) Phase 3.

Figure 1. The three stages of DINTA in a decision epoch. a) The sensor network detects events (marked A_1 and A_2) and propagates event data throughout the network. b) Next, nodes that have unassigned robots in their vicinity propagate distances (in hop counts) from robots to each of the alarms. c) In the final stage, every node in the network has the same information about the location of events and available robots, and distances between robots and each event. Hence, a unique assignment of direction suggestion at every node can occur.

4. MRTA Experiments in Simulation

In the first set of experiments described here we used the Player/Stage (Gerkey et al., 2003) simulation engine populated with simulated Pioneer 2DX mobile robots. A network of 25 network nodes (simulated motes (Pister et al., 1999)) was pre-deployed in a test environment of size $576m^2$. The communication range of the nodes and robots was set to approximately 4 meters. Robots were required to navigate to the point of each 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. Every experiment was conducted in the same environment with robot group sizes varying from 1 to 4, 10 trials per group. The schedule of 10 alarms was drawn from a Poisson distribution ($\lambda = \frac{1}{60}$, roughly one alarm per minute), with uniformly distributed nodes that detected alarms.

We measured cumulative alarm *On Time* for network-mediated task allocation (*i.e.,* DINTA). As a base case we compared the results to the situation where the robots are programmed to explore the environment using directives from the sensor network designed only to optimize their environmental coverage (Batalin and Sukhatme, 2004a). The comparison highlights the benefits of purposeful task allocation. Fig-

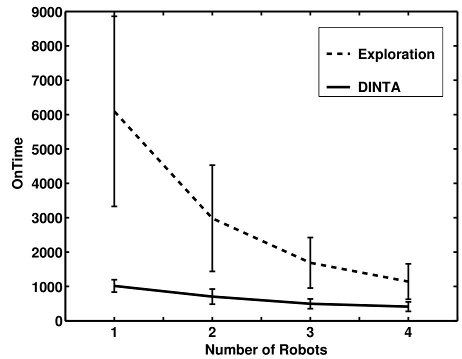


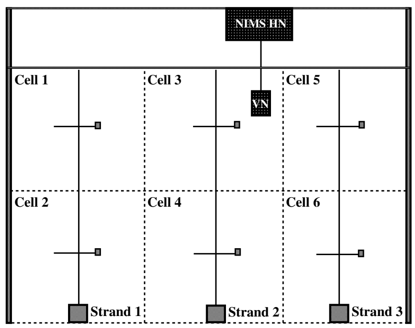
Figure 2. Comparison between implementation of DINTA and exploration-only.

ure 2 shows the *OnTime* comparison for DINTA and the exploration-only case. Clearly, DINTA outperforms the exploration-only algorithm even though as the environment becomes saturated with robots, the difference becomes smaller. The difference is statistically significant (the T-test p-value is less than 10^{-4} for every pair in the data set). Further, the performance of DINTA is stable (small and constant variance) whereas variances produced by the exploration-only mode change drastically and reduce as the environment becomes saturated with robots.

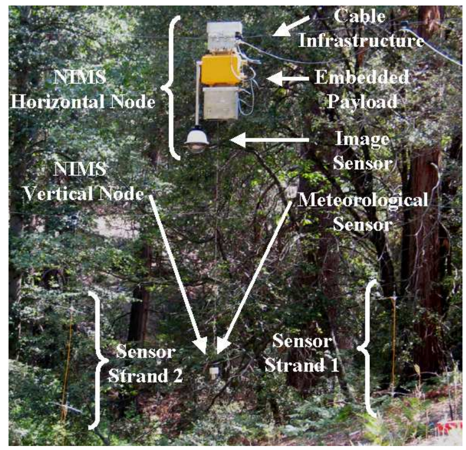
5. Laboratory Experiments with NIMS

The second set of experiments we discuss use a new testbed, currently under development - Networked Info-Mechanical System (NIMS, 2004). Figure 3 shows NIMS deployed in a forest reserve for continuous operation. The system includes supporting cable infrastructure, a horizontally moving mobile robot (the NIMS node) equipped with a camera, and a vertically mobile meteorological sensor system carrying water vapor, temperature, and photosynthetically active radiation (PAR) sensing capability. The purpose of NIMS is to enable the study of spatiotemporal phenomena (*e.g.,* humidity, carbon flux, *etc.*) in natural environments. Figure 3a schematically shows NIMS with deployed static sensor nodes (assembled in strands) in the volume surrounding the sensing transect. Wireless networking is incorporated to link the static sensor nodes with the NIMS node. The NIMS system is deployed in a transect of length 70m and average height of 15m with a total area of over 1,000 m^2.

The experimental NIMS system operates with a linear speed range for node motion of 0.1 to 1 m/second. Thus, the time required to map an entire 1,000 m^2 transect with 0.1 m^2 resolution will exceed 10^4 to



(a) NIMS horizontal (HN) and vertical (VN) nodes and static sensors (schematically)



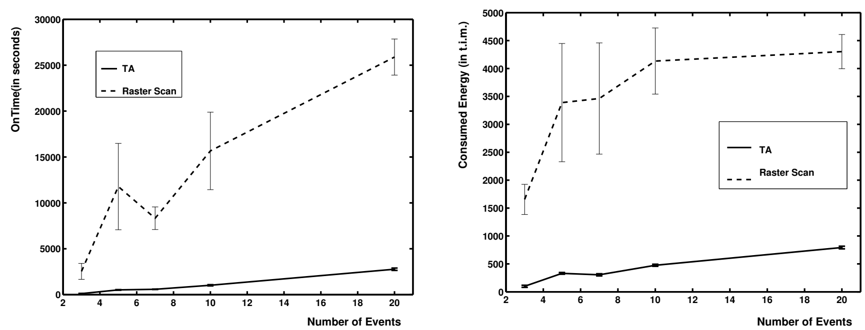
(b) NIMS deployed in a forest reserve

Figure 3. NIMS system deployed in the forest reserve for continuous operation.

10^5 seconds. Phenomena that vary at a characteristic rate exceeding this scanning rate may not be accurately represented. Hence task allocation is required to focus sampling in specific areas depending on their scientific value. The preliminary experiments using our in-network task allocation methodology show an order of magnitude improvement in the time it takes to complete sampling.

We conducted experiments on a smaller version of NIMS installed in the lab[2]. A network of 6 Mica2 motes was pre-deployed in the volume surrounding the NIMS transect (similar to Figure 3a) in a test environment. Experiments were conducted comparing a version of DINTA with a Raster Scan (RS) as a base case. RS is an algorithm of choice when there is no information about the phenomenon location (where the alarms are). RS scans every point of the transect with a specified resolution. When the Raster Scan reaches the location of an alarm, the alarm is considered to be turned off.

In our experiment, schedules of 3, 5, 7, 10 and 20 alarms (henceforth, events) were drawn from a uniform distribution to arrive within 10 minutes, with uniformly distributed nodes that detected the event. Note that for actual applications we do not expect to receive/process more



(a) Event OnTime.

(b) Energy consumption.

Figure 4. NIMS lab experiments: task allocation vs. a raster scan.

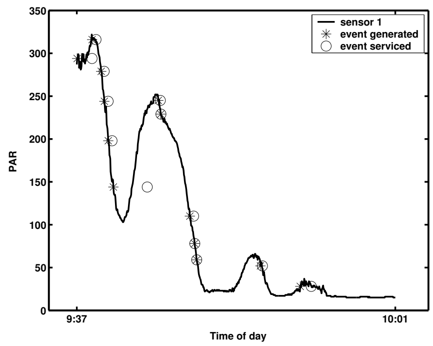
than 1 - 10 events in 10 minutes on average. Hence the case of 20 events shows the behavior of the system at the limit.

Figure 4 shows experimental results comparing *OnTime* performance of DINTA and RS. The number of events varies between 3 and 20. Both algorithms were evaluated from 3 different starting positions of the mobile node on the transect (drawn from a uniform distribution). The results were averaged. As can be seen from the graph, DINTA performs 9-22 times better on the entire interval of 3-20 events. Note also that DINTA is stable, as indicated by error bars, and hence is favored for use in this application since it provides reduced bounds on system run time over a simple Raster Scan method.

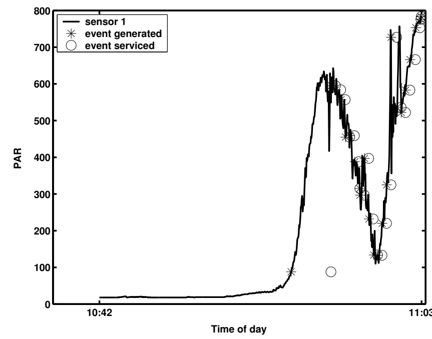
We also compared mobility requirements for DINTA and RS methods. Specifically, the use of mobility requires energy. A measure of energy for mobility is determined for the purposes of comparison by computing the total time of the robot motion. Figure 4 shows a comparison of energy consumption in units of time-in-motion. As expected, DINTA outperforms Raster Scan significantly. However as the number of events increases to infinity, DINTA will approach Raster Scan energy consumption. Also note, that on the interval [5,20] the slope of the Raster Scan curve is very small and the energy consumption is insensitive to event arrival rate.

6. Field Trials using NIMS

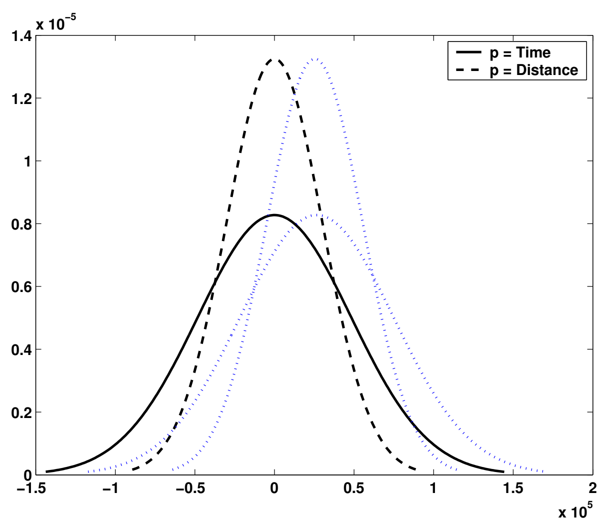
The third, and final, set of experiments discussed here were performed in field trials with the NIMS system. We used our task allocation system



(a) Time policy.



(b) Distance policy.



(c) Event OnTime.

Figure 5. NIMS field experiments for two policies. a,b) PAR data acquired by the first sensor during one of the field experiments. Events generated and serviced are shown for Time and Distance policies. Note that events are rendered time of occurrence vs. the PAR value of the event. c) Event OnTime in a form of a zero-mean Gaussian distributions for Time and Distance policies. The OnTime of events generated by all 6 sensors is considered. Dotted (blue or lighter) graphs show the distributions at original means.

and compared two policies - *Time* (tasks with smaller time stamp get priority) and *Distance* (tasks closer to the robot get priority). A set of experiments was conducted on a NIMS setup deployed in the James San Jacinto Mountain Reserve. Because of space limitations, only representative graphs are presented. Figure 5 shows the representative PAR data from sensor 1 collected during the operation of the *Time* policy (Figure 5a) and the *Distance* policy (Figure 5b). Figure 5 also shows points in time when events were generated and serviced by both policies for sensor 1. Note that events are generated in response to fluctuations in PAR. As shown on Figure 5, events are generated proportionally to the density of the 'spikes' in PAR data and cover all significant 'spikes' of PAR data.

Figure 5c shows the comparison between the cumulative event *OnTime* of the *Time* policy and the *Distance* policy. For visualization purposes, in Figure 5c event's *OnTime* is presented as a zero-mean Gaussian distribution. It follows that the *Distance* policy has smaller average *OnTime* with smaller deviation.

7. Summary

We presented a novel, sensor network-mediated, approach to multi robot task allocation. Our algorithm DINTA: Distributed In-Network Task Allocation solves the online multi robot task allocation problem. This approach allows us to combine the benefits of a sensor network with the 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. There are several advantages in using DINTA as opposed to traditional MRTA approaches. The sensor network allows a robot to detect a goal (alarm, event) even though the alarm is not in the robot's sensor range. In addition, robots can use the sensor network to relay messages if they are not within communication range of each other. Further, robots can be very simple and potentially heterogeneous. We also presented physical experimental results of using DINTA for field measurements in natural setting using a monitoring infrastructure composed of mobile robots on cables and network nodes in the vicinity of the cable transect.

Acknowledgments

This work is supported in part by the National Science Foundation (NSF) under grants IIS-0133947, EIA-0121141 and grants CCR-0120778, ANI-00331481 (via subcontract). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Notes

1. For implementation details of DINTA see (Batalin and Sukhatme, 2004b.)
2. For experimental and other details see (Batalin et al., 2004a).

References

Batalin, M., Rahimi, M., Yu, Y., Liu, S., Kansal, A., Sukhatme, G., Kaiser, W., Hansen, M., Pottie, G., Srivastava, M., and Estrin, D. (2004a). Call and Response: Experiments in Sampling the Environment. In *Proc. ACM SenSys*.

Batalin, M. and Sukhatme, G. (2004a). Coverage, Exploration and Deployment by a Mobile Robot and Communication Network. *Telecommunication Systems Journal, Special Issue on Wireless Sensor Networks*, 26(2):181–196.

Batalin, M. and Sukhatme, G. (2004b). Using a Sensor Network for Distributed Multi-Robot Task Allocation. In *Proc. IEEE International Conference on Robotics and Automation*, pages 158–164, New Orleans, Louisiana.

Batalin, M., Sukhatme, G., and Hattig, M. (2004b). Mobile Robot Navigation using a Sensor Network. In *Proc. IEEE International Conference on Robotics and Automation*, pages 636–642, New Orleans, Louisiana.

Botelho, S. and Alami, R. (2000). M+: A Scheme for Multi-Robot Cooperation through Negotiated Task Allocation and Achievement. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 293–298.

Gerkey, B., Vaughan, R., and Howard, A. (2003). The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems. In *Proc. International Conference on Advanced Robotics (ICAR 2003)*, pages 317–323, Coimbra, Portugal.

Gerkey, B. P. and Mataric, M. J. (2004). A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems. *Intl. Journal of Robotics Research*, 23(9):939–954.

Koenig, S. and Simmons, R. G. (1992). Complexity Analysis of Real-Time Reinforcement Learning Applied to Finding Shortest Paths in Deterministic Domains. Technical Report CMU-CS-93-106, Carnegie Mellon University, School of Computer Science, Carnegie Mellon University, Pittsburg, PA 15213.

NIMS (2004). http://www.cens.ucla.edu/portal/nims.html.

Parker, L. E. (1998). ALLIANCE: An Architecture for Fault-Tolerant Multi-Robot Cooperation. In *IEEE Transactions on Robotics and Automation*, volume 14, pages 220–240.

Pister, K. S. J., Kahn, J. M., and Boser, B. E. (1999). Smart Dust: Wireless Networks of Millimeter-Scale Sensor Nodes. *Electronics Research Laboratory Research Summary*.

Werger, B. B. and Mataric, M. J. (2000). *Distributed Autonomous Robotic Systems 4*, chapter Broadcast of Local Eligibility for Multi-Target Observation, pages 347–356. Springer-Verlag.