Mobile Robot Task Allocation in Hybrid Wireless Sensor Networks

Brian Coltin and Manuela Veloso

Abstract— **Hybrid sensor networks consisting of both inexpensive static wireless sensors and highly capable mobile robots have the potential to monitor large environments at a low cost. To do so, an algorithm is needed to assign tasks to mobile robots which minimizes communication among the static sensors in order to extend the lifetime of the network. We present three algorithms to solve this task allocation problem: a centralized algorithm, an auction-based algorithm, and a novel distributed algorithm utilizing a spanning tree over the static sensors to assign tasks. We compare the assignment quality and communication costs of these algorithms experimentally. Our experiments show that at a small cost in assignment quality, the distributed tree-based algorithm significantly extends the lifetime of the static sensor network.**

I. INTRODUCTION

Sensor networks have recently emerged as an effective tool for monitoring large-scale environments, and have been successfully deployed to solve problems as diverse as detecting floods, controlling the temperature in office buildings, and monitoring hospital patients [1]. In order to deploy sensor networks in such large environments, often with hundreds of nodes, wireless sensors must be low-cost and affordable. Hence, wireless sensors are typically highly limited in terms of sensing, computation, communication, battery life, and the actions they can perform.

These limitations can be addressed through the addition of more capable mobile sensors (either robotic or human) to form a hybrid wireless network. For example, in precision agriculture, static sensors may be deployed to monitor plants in a greenhouse. Higher-capability mobile robots may be dispatched to gather more accurate temperature or humidity readings, or to take soil samples which the static sensors are not equipped for. Although the static sensors are less capable than mobile robots, they are also much less expensive and can be deployed to cover a vast area at a low cost.

In general, static sensors detect *events* which must be handled by mobile robots. These events are associated with a point in space where a mobile robot is needed to perform a task, such as gathering a more accurate temperature reading, spraying pesticides, or recharing a static sensor's battery. The static sensors must assign these events to mobile robots while simultaneously minimizing the distance travelled by the robots and the communication among the static sensors to prolong their battery life.

We split this event assignment problem into two subproblems: the mobile to static (MtS) assignment problem, in which mobile robots are assigned to specific static sensors to handle the events surrounding them, and the mobile to event (MtE) local assignment problem, in which static sensors allocate events to their assigned mobile robots. We present three algorithms for solving the MtS problem: a centralized algorithm, an auction-based algorithm, and a novel tree-based greedy algorithm. The tree-based algorithm uses a spanning tree network topology and a greedy allocation policy to localize sensor communications and reduce communication, prolonging the lifetime of the static sensors. We show empirically that the centralized and auction approaches provide small improvements in assignment quality, while the centralized and tree-based greedy algorithms require much less communication. The tree-based algorithm balances the communication more evenly, so the sensors are expected to die less quickly than with a centralized approach. Furthermore, the centralized algorithm is more susceptible to the failure of a single sensor.

In the next section, we briefly discuss related work in both multi-robot task allocation and sensor networks. In section III we formally present the event assignment problem, and in section IV we present three algorithms to solve the MtS problem: a centralized algorithm, an auction algorithm, and a distributed greedy algorithm. Finally, in section V we compare these algorithms in terms of their assignment quality and communication cost.

II. RELATED WORK

The problem of assigning mobile robots to static sensors is an instance of the multi-robot task allocation problem, in which a group of robots is given a list of tasks to complete. The goal is to construct a schedule of tasks for each robot which minimizes the total cost to complete all the tasks. This problem is strongly $\mathcal{NP}$-hard, and so a common approach is to greedily assign tasks to robots as the robots become available [2]. Another popular approach is a free-market based auction system, where the mobile robots place bids on tasks based on the cost to accomplish them [3]. The task allocation problem we consider has the additional requirement of minimizing the communication between static sensors to prolong the network lifetime, and the additional networking structure provided by the static nodes.

Other research has focused on achieving high sensor coverage of an environment with a sensor network composed

This research was partially sponsored by the Office of Naval Research under grant number N00014-09-1-1031. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

B. Coltin is with The Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA, United States bcoltin@cs.cmu.edu

M. Veloso is with the Computer Science Department, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA, United States veloso@cs.cmu.edu

entirely of mobile sensors. Heuristic-based algorithms have been developed to deploy the mobile robots to an initial configuration with good coverage of the environment, where after deployment they act as static sensors [4] [5]. Wang, et al., have studied redeploying these mobile sensors in response to sensor failures and new events using a grid-based communication framework [6], and have also developed an auction algorithm for the coverage problem in hybrid networks of both static and mobile sensors, where the mobile agents bid on coverage holes [7]. Again, these algorithms are designed to achieve coverage of the environment with the mobile sensors, rather than handle specific events which is our goal.

A few real-world systems have utilized hybrid networks of both static and mobile sensors. In [8], Vasilescu, et al., deploy a team of ten static nodes and two mobile nodes to monitor underwater environments. The mobile nodes act as data ferries between the static sensors to address the difficulties of underwater radio communication. In [9], Sukhatme, et al., deploy ten static buoys and a single mobile boat to monitor microorganism levels in a lake. The mobile boat shares its sensor readings with the network of static buoys, which instruct the boat on what areas to observe next.

In [10], Wang, et al., present a distributed task allocation algorithm, GridSD, to allocate tasks observed by static sensors to mobile robots while minimizing communication costs. GridSD groups the static sensors into rectangular cells on a grid, each with a grid head which performs a centralized task allocation algorithm within the grid cell. The grid heads share the number of available mobile robots in its cell with other grid heads in the same column on the grid. When a grid head needs more mobile robots to perform tasks within its cell, it requests them from its column-wise neighbors, if available, and if not, forwards the request to its row-wise neighbors which perform the same search procedure. This algorithm reduces communication by routing messages along the grid structure. However, to form a grid, the sensors must be somewhat evenly distributed so that no cells are empty. The algorithms we present forgo this assumption.

III. EVENT ASSIGNMENT PROBLEM

The event assignment problem takes as input a tuple (S, M_t, E_t), where:

- S is the set of static sensors,
- M_t is the set of available mobile robots at time t, which are not currently assigned an event, and
- E_t is the set of events observed by the sensors in S at time t which do not have a mobile robot assigned to handle them. Events are not necessarily constrained to physical occurrences, but may represent anything a static sensor needs assistance with, such as watering a plant, improving sensing coverage, restoring or reinforcing network connectivity, or repairing static sensors.

The event assignment problem is, given (S, M_t, E_t), to choose an assignment $f : M_t \rightarrow E_t$ from each mobile robot $m \in M_t$ to an event $e \in E_t$, or to no event. In solving the event assignment problem we have two objectives:

- Minimize the *distance travelled* by the mobile robots to conserve fuel. This objective cannot be solved optimally due to the dynamic nature of the environment; static sensors detect new events over time, so the total distance travelled cannot be predicted a priori. When a robot is assigned an event, it proceeds to that event and handles it, and then re-enters M_t as an unassigned mobile robot and is assigned a new event. Once an assignment is made, the robot must complete it— no reassignments are permitted.
- Minimize *communication* among static sensors. The static sensors may run on limited battery power, and are difficult to recharge. So we must minimize the number of messages they send to lengthen the network lifetime (all messages are assumed to be short, of a fixed length, and both sensors and mobile robots have limited communication ranges). Furthermore, we wish to balance the communication load among the static sensors— if the load is uneven, the more heavily-trafficked nodes die more quickly.

The relative importance of these two objectives depends on the problem domain: with static sensors which are easy to recharge, minimizing the distance travelled should take precedence, but with highly capable mobile robots (or humans) and power-limited static sensors, minimizing communication should take priority.

We separate the event assignment problem into two sub-problems: the *local Mobile to Event (MtE) assignment problem* and the *Mobile to Static (MtS) assignment problem*. In this framework, mobile robots are assigned to static sensors (the MtS problem), and the static sensors then assign these robots to the events which they own (the MtE problem). Each event $e \in E_t$ is *owned* by a static sensor $s \in S$ if and only if s detects the event e and s is the nearest static sensor to e. The owner of an event takes responsibility for assigning that event to a mobile robot. Each static sensor $s \in S$ has a need $num\_events(s)$, which is the number of events in E_t (unassigned events) it owns. Static sensor s acquires up to $num\_events(s)$ mobile robots to handle these events. In the MtS problem, mobile robots are assigned to static sensors rather than events based on the need at each sensor. The mobile robots then proceed to their assigned static sensor and execute the MtE assignment algorithm, in which the static sensor assigns specific events to the mobile robots it owns. With this two-layered approach, the static sensors only share the number of mobile robots they need rather than information about each specific event, which reduces communication, since many events may be covered by the same static sensor.

To solve the MtE problem, we use a greedy algorithm. At time t, sensor $s \in S$ has a set $A_t \subseteq M_t$ of mobile robots assigned to it but not assigned events, and a set $U_t \subseteq E_t$ of unassigned events it must handle. Each sensor handles the events in its Voronoi cell, i.e., the events to which it is the nearest sensor. Due to the online nature of the problem, and the fact that the offline version is $\mathcal{NP}$-hard, we employ a greedy algorithm. While there are pairs of unassigned mobile

robots and unassigned events available, make the assignment of minimum cost (the cost is the distance the mobile robot must travel). When there are no more events to handle, the mobile robots are released from their assignment to the static sensor, and reassigned to new static sensors through the MtS assignment algorithm. We use this same local MtE assignment algorithm in conjunction with all of the MtS assignment algorithms for a fair comparison, since our focus in this paper is the MtS assignment problem rather than the MtE assignment problem.

Before discussing algorithms to solve the MtS assignment problem, we must consider the assumptions we make in terms of the capabilities of the static sensors and mobile robots. We assume that the mobile robots are able to localize in their environment, and that the static and mobile robots possess a map with the positions of the static sensors. Furthermore, the static sensors are able to form a connectivity graph C detailing which static sensors are able to communicate.

IV. MOBILE TO STATIC ASSIGNMENT ALGORITHMS

In this section, we present three algorithms to solve the MtS problem. The first is a centralized algorithm, where a single static sensor receives the requirements of the static sensors and the availability of the mobile robots, and provides assignments to the robots. Second, we present a distributed auction algorithm, where the static sensors bid for the services of mobile robots. Finally, we introduce a novel distributed algorithm based on a tree network structure to make assignments. This algorithm takes advantage of the topology of the network and a greedy assignment process to reduce communication costs.

A. Centralized Algorithm

In a centralized algorithm, a lead sensor is selected to perform all of the decision making. Since the sensors each have a map with the positions of all the other sensors, we select the sensor closest to the centroid of all the sensors as the leader, s_l.

Whenever a robot is not assigned to a static sensor, either because it has just been initialized or was released from its previous assignment by the local MtE assignment algorithm, it sends a request for assignment to the nearest static sensor. This request contains the mobile robots's ID, the ID of the nearest static sensor (so that the leader can send the mobile robots its assignment through a path in the connectivity graph C), and the current (x, y) position of the mobile robot. The static sensors forward the request along the shortest path in C to s_l. Similarly, when the number of robots needed by a static sensor s_i changes, either because robots successfully handled events or because more events occurred, the static sensor sends the new need $num\_events(s_i)$ to s_l through the shortest path in C.

The leader, s_l, accumulates a list of the number of mobile robots $num\_events(s_i)$ needed by each static sensor. s_l also gathers, for each unassigned mobile robot m, its position $pos(m)$, and the ID $closest(m)$ of the closest static sensor to that robot. Every time-step, s_l makes assignments using the greedy Algorithm 1. Algorithm 1 iteratively makes the assignment of lowest cost, in the same manner as the greedy algorithm used for the MtE problem. Again, we use a greedy algorithm since we are solving an online problem.

Algorithm 1 Assign unassigned mobile robots $m_i \in M$ to static sensors $s_i \in S$ with need greater than the number of currently assigned mobile robots. Returns a set of tuples $(m, s) \in R$ of assignments of mobile robots to static sensors.

```
R ← ∅
while M ≠ ∅ and S ≠ ∅ do
  (m, s) ← arg min_{m∈M, s∈S} ||pos(m) − pos(s)||
  R ← R ∪ {(m, s)}
  M ← M \ {m}
  num_events(s) ← num_events(s) − 1
  if num_events(s) = 0 then
    S ← S \ {s}
  end if
end while
```

Once an assignment has been made, i.e. m is assigned to static sensor s, the lead sensor sends the assignment along the shortest path in C to static sensor $closest(m)$, which forwards the message to m. The mobile robot then proceeds to sensor s, which it informs of its arrival and assignment. Then, s directs m to handle events with its MtE assignment algorithm, until there are no events left and m is released from its assignment. Then, m requests a new assignment from s and the process repeats.

B. Auction of Mobile Robots

In our second approach, the mobile robots hold auctions for their services among the static sensors, in a manner similar to [7]. When a mobile robot becomes available, either initially or because it was released from its previous assignment, it sends an announcement of an auction to the neighboring static sensors. This announcement contains the mobile robot's ID, position, and the ID of the nearest static sensor. The static sensors percolate the announcement through the entire network. Each of the static sensors forms a list a of the mobile robots holding auctions. When a static sensor decides it needs more mobile robots to aid it, it sends a bid to the nearest mobile robot in a with the position of the static sensor. A static sensor can place several bids at a time, up to the number of mobile robots it needs.

After a mobile robot m has called an auction, it forms a list b of the static sensors which have placed bids. The mobile robot then waits a time d_{auction} before it selects a winner. d_{auction} should be chosen based on the communication delay and the number of static sensors, so that all of the static sensors have time to place bids. If after d_{auction} time passes no bids have arrived, m continues waiting and accepts the first bid it receives (since calling a new auction incurs a high overhead). Otherwise, the nearest static sensor in b to m is greedily selected as the winner. m then sends a message

announcing the winner of the auction, which percolates to all the static sensors. The static sensors remove m from their list a of ongoing auctions, and the winner adds m to the list of robots working for it. m then proceeds to the winner's location, and executes the MtE assignment algorithm.

C. Tree-based Greedy Algorithm

In our third approach, we form a spanning tree T of the connectivity graph C. We use a minimum spanning tree where the metric is Euclidean distance between the static sensors. The sensors agree on the spanning tree because they have the same map of the environment. When a sensor dies, this information percolates through the network, and a new tree is constructed. If the graph is no longer connected, the algorithm gracefully degrades into using a forest of trees. A mobile sensor may be dispatched to restore network connectivity.

We take advantage of the tree network topology and a greedy algorithm to reduce the communication costs needed to make assignments. Unlike the auction algorithm, which is reactive in the sense that static sensors only respond to announcements from mobile robots, the tree-based algorithm is proactive in that the static sensors form a model of where the mobile robots are needed before any request from the mobile robots arrives. The imposition of a network structure to reduce communication is similar to GridSD, which uses grid cells, except a tree is more generally applicable.

Rather than modeling the number of robots needed at every static sensor, as in GridSD, we only store, for each static sensor s, the distance (in terms of the number of edges on T) to the nearest static sensor which needs mobile robots from each outgoing edge of s. Consider a static sensor s which has a list N of neighbors in T. This static sensor stores a list d of $|N|$ integers, one for each of s's $|N|$ neighbors. d_i is the distance from s to the nearest node in the branch of the tree containing n_i, where the edge from n_i to s is removed. In Fig. 1, the final values of d_i are shown after all the information has percolated through the network.

The d_i for each static sensor s are updated according to Algorithm 2. This algorithm only sends information when the d_i for the recipient would change based on the new information. This reduces the total number of messages sent, since a change in need at a sensor will not percolate through the entire network unless there are no other events. In a network where every sensor needs mobile robots, each node only sends a single message to form this model, and no information is forwarded.

With Algorithm 2, each of the static sensors stores the distance from each of its neighbors to the nearest sensor which needs mobile robots. Next, we describe how this information is used to make assignments. Similar to the centralized algorithm, when a mobile robot is unoccupied, it sends a request for an assignment to the nearest static sensor. If this static sensor has need of a mobile robot, it accepts the request. Otherwise, the static sensor forwards the request to its neighbor in T with the minimum d_i (the closest node in the tree which needs a mobile robot). If none of the neighbors

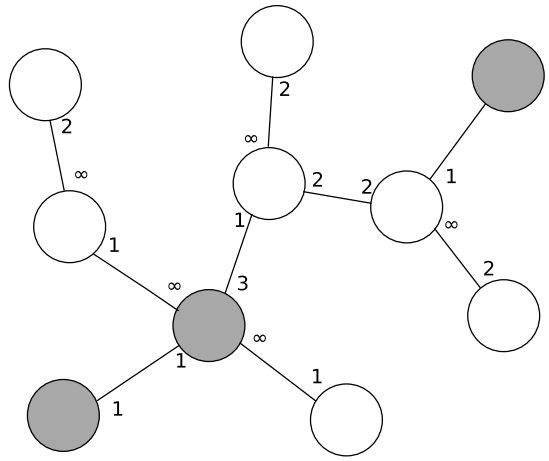


Fig. 1. The distances d_i to the nearest node which needs mobile robots in each branch of the tree after all nodes have sent updates to their neighbors. Grey sensors have a need for mobile robots, and the number next to each edge emanating from a sensor s to its neighbor n_i is the value of d_i for that sensor and neighbor pair, or the length of the shortest path from s to a node with nonzero need which passes through n_i.

Algorithm 2 Every time step, a static sensor s updates its own values d for the distances to the nearest location a mobile robot is needed at on each branch, and forwards any changes to its neighbors. o is the list of distances most recently sent to each neighbor, all initialized to ∞.

```
Update d based on received messages (initialized to ∞)
for all n_i ∈ n do
  if num_events(s) > 0 then
    dist ← 1
  else
    dist ← 1 + min d \ {d_i}
  end if
  if dist ≠ o_i then
    Send distance dist to sensor n_i
    o_i ← dist
  end if
end for
```

needs a mobile robot ($\min d = \infty$) the static sensor holds onto the request until a change in d occurs. In the case of a tie, a winner is chosen at random.

The request continues percolating through the tree until a static sensor is reached which accepts it. Then the acceptance is sent back to the mobile robot. Due to the dynamic nature of events and assignments, it is possible that while traversing the tree, an assignment request may arrive at a node which needs no mobile robot, and all of the edges emanating from that node (excluding the edge the request came from) also need no mobile robots with $d_i = \infty$. In this case, the request has failed, and this failure is sent back to the static sensor which initially received the request from the mobile robot. This node then repeats the process to receive an assignment. If the request is for the original static node, $\forall i \ d_i = \infty$, the sensor stores the request until a mobile robot is needed. When a mobile robot receives an assignment, it proceeds to

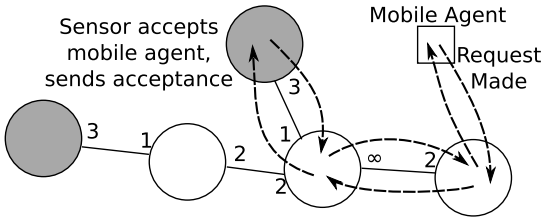


Fig. 2. A mobile robot is assigned to a static sensor through forwarding messages along the tree. Circles are static sensors, and filled circles represent sensors with nonzero need. The numbers next to edges represent the length of the shortest path to a sensor of nonzero need in that direction.

the designated static sensor and executes the MtE assignment algorithm. Fig. 2 illustrates how a mobile robot's request is sent through the network, taking the shortest path on the tree.

The advantage to the tree structure comes from how the requests of mobile robots are passed through the tree. Unlike the auction algorithm, the request is not percolated to the entire network, but to the nearest node with need on the tree, which greatly reduces bandwidth usage. This algorithm also has the advantages of being a distributive algorithm, and increased robustness over a centralized approach due to the lack of a single point of failure. However, this comes at a price in terms of reduced assignment quality, since distances on the tree do not directly correspond to Euclidean distance.

V. Experimental Results

To evaluate the three static sensor assignment algorithms, we ran experiments in simulation to compare the assignment quality and communication costs. The use of simulation allows us to run large scale experiments involving hundreds of mobile robots and static sensors. The task allocation problem is equally relevant to real-world deployments of hybrid networks. We compare the three algorithms to a modified version of GridSD [10], in which the grid heads use our local MtE assignment algorithm. We evenly distribute one hundred static sensors on a grid in a square environment to achieve full coverage. We test the algorithms in three different scenarios:

- an *offline* scenario in which events are spread uniformly at random throughout the environment, and are all detectable from the beginning of the simulation
- an *online* scenario, in which the events are scattered uniformly at random through space and a fixed time interval, and
- a scenario with *event-locality*, in which events occur at fixed intervals in time, and in a fixed pattern in space, occurring in the neighborhood of a point which moves up and down the diagonal of the square environment over time.

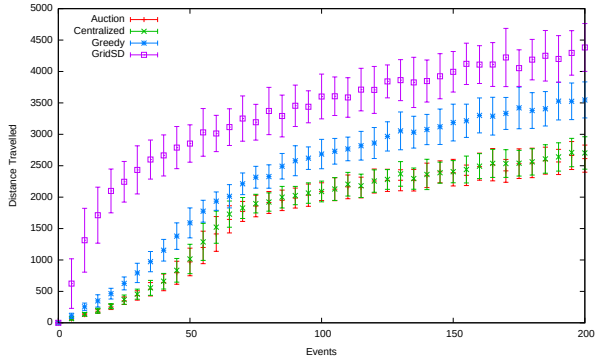
These scenarios were designed to show the performance of the four algorithms on both offline (where all events are known before hand) and online problems, and to demonstrate any benefits in terms of communication cost from the local communication of the tree-based algorithm and GridSD.

For each scenario, we run three tests with three different independent variables: the number of mobile robots m, the number of events e, and the number of static sensors s in addition to the one hundred static sensors deployed on the grid to obtain full coverage. We exclude GridSD from the experiment with variable static sensors since our local MtE assignment algorithm is incompatible with GridSD's centralized, cell-based assignment of mobile robots for cells with multiple static sensors. The mobile robots and additional static sensors are distributed uniformly at random throughout the environment. For every set of m, e and s, we run fifty trials on each of the algorithms in simulation, and measure the mean and standard deviation over the trials of the total distance travelled by the mobile robots, the total number of messages sent, and the maximum number of messages sent by a single robot. The maximum number of messages sent by a single robot is correlated with how quickly static sensors will begin to run out of power and die. For the trials in which m varies, $s = 0$ and $e = 100$, for those in which s varies, $m = 10$ and $e = 100$, and for those in which e varies, $s = 0$ and $m = 10$. Selected results are shown in Fig. 3.

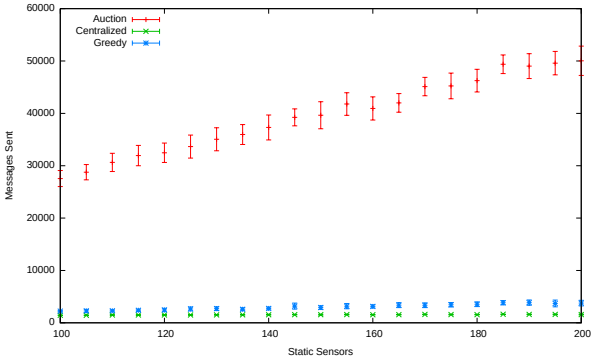
Fig. 3a shows the total cost in terms of distance travelled for each algorithm. The centralized and auction algorithms provide the best assignments because they select greedily over all available possibilities. The tree-based algorithm performs more poorly since the selection is based on the distance in the tree rather than the distance in space, and GridSD performs the poorest since assignments in the same column as the sensor are chosen before nearby columns are considered.

Fig. 3b shows how the communication costs vary with s. The messages sent in the auction algorithm increase rapidly, since every announcement and result of a mobile robot auction is sent to every static sensor. The cost of the centralized and tree-based algorithm both increase slowly with s. Fig. 3c shows how the total number of messages sent changes with the number of events e. The auction once again has the highest cost, followed by GridSD. The cost of GridSD is high since every time a mobile robot becomes available for an assignment, the entire column of static sensors is informed, which requires $\sqrt{s}$ messages. The tree-based algorithm begins with an equally rapid increase, but levels off as the effects of forwarding only the closest information kick in. The centralized communication cost continues to increase at a constant rate.

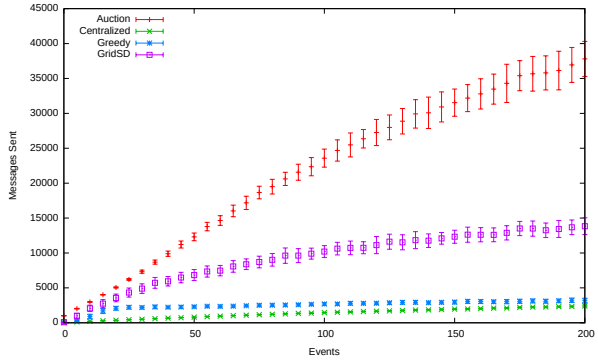
In Fig. 3d, the maximum number of messages sent by a single sensor is shown. The auction algorithm and GridSD have the highest values, since their total communication is higher. But even though the tree-based algorithm has higher total communication than the centralized algorithm, the maximum number of messages sent for a single robot is approximately half as much in the online scenario for two hundred events. This is because in the centralized algorithm, all the messages are routed through the leader. The tree-based algorithm, on the other hand, is distributed. This implies that even though the centralized algorithm communicates less in total, sensors are likely to begin to die more quickly when running the centralized algorithm.



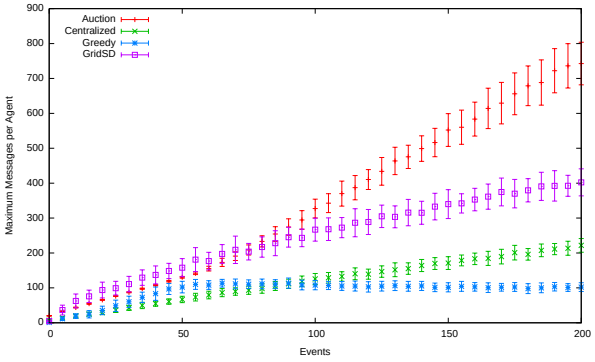
(a) Online Scenario, Distance Traveled when e Varies



(b) Offline Scenario, Communication Costs when s Varies



(c) Spatial-Locality Scenario, Net Communication when e Varies



(d) Online Scenario, Max Comm. per Agent when e Varies

Fig. 3. Selected results from the experiments. See the text for details.

One factor which is not shown in the experiments is that the auction and tree-based algorithms are decentralized, so if a node dies the network will continue its task. This is not the case for the centralized algorithm, where the death of the leader is problematic (and the leader is likely to die the most quickly). Similarly, in GridSD, if a grid grid cell has no active nodes, the algorithm will fail. The assignment algorithm to choose largely depends on the structure of the problem. If the highest quality assignment is essential, a centralized or auction-based approach would serve best, depending on the reliability of the sensors and how much of a concern power usage is. If a longer network lifetime is important, the tree-based algorithm may be the best option.

VI. CONCLUSION

In this paper, we presented the event assignment problem for assigning mobile robots to events in hybrid wireless sensor networks. We discussed three algorithms to solve this problem: a centralized algorithm, an auction-based approach, and a greedy approach which uses a spanning tree of the network. We evaluated the advantages and disadvantages of each in terms of assignment quality, total communication cost, and expected network lifetime. Our results showed a lower assignment quality for the tree-based algorithm, but with a low communication cost, a high expected network lifetime, and the advantages of a decentralized algorithm.

REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, 2002.

[2] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The Intl. J. of Robotics Research*, vol. 23, no. 9, 2004.

[3] R. M. Zlot, A. T. Stentz, M. B. Dias, and S. Thayer, "Multi-robot exploration controlled by a market economy," in *IEEE International Conference on Robotics and Automation*, 2002.

[4] A. Howard, M. J. Matarić, and G. S. Sukhatme, "An incremental self-deployment algorithm for mobile sensor networks," *Autonomous Robots*, vol. 13, 2007.

[5] B. Zhang and G. S. Sukhatme, "Controlling sensor density using mobility," in *EmNets '05: Proceedings of the 2nd IEEE workshop on Embedded Networked Sensors*, 2005.

[6] G. Wang, G. Cao, T. L. Porta, and W. Zhang, "Sensor relocation in mobile sensor networks," in *In Proc. of IEEE INFOCOM*, 2005.

[7] G. Wang, G. Cao, and T. La Porta, "A bidding protocol for deploying mobile sensors," in *ICNP '03: Proceedings of the 11th IEEE International Conference on Network Protocols*, 2003.

[8] I. Vasilescu, K. Kotay, D. Rus, M. Dunbabin, and P. Corke, "Data collection, storage, and retrieval with an underwater sensor network," in *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, 2005.

[9] G. S. Sukhatme, A. Dhariwal, B. Zhang, C. Oberg, B. Stauffer, and D. A. Caron, "The design and development of a wireless robotic networked aquatic microbial observing system," *Environmental Engineering Science*, vol. 24, no. 2, 2007.

[10] Y.-C. Wang, W.-C. Peng, M.-H. Chang, and Y.-C. Tseng, "Exploring load-balance to dispatch mobile sensors in wireless sensor networks," in *In Proc. of Int. Conf. on Computer Communications and Networks*, 2007.