Sensor-Mission Assignment in Wireless Sensor Networks

HOSAM ROWAIHY[1], MATTHEW P. JOHNSON[2], OU LIU[2], AMOTZ BAR-NOY[2], THEODORE BROWN[2] and THOMAS LA PORTA[1]

[1]Department of Computer Science and Engineering, Pennsylvania State University
[2]Department of Computer Science, The Graduate Center, City University of New York

When a sensor network is deployed, it is typically required to support multiple simultaneous missions. Schemes that assign sensing resources to missions thus become necessary. In this article, we formally define the sensor-mission assignment problem and discuss some of its variants. In its most general form, this problem is NP-hard. We propose algorithms for the different variants, some of which include approximation guarantees. We also propose distributed algorithms to assign sensors to missions which we adapt to include energy-awareness to extend network lifetime. Finally, we show comprehensive simulation results comparing these solutions to an upper bound on the optimal solution.

Categories and Subject Descriptors: []: —

General Terms: Wireless Sensor Networks, Resource Allocation, Mission Assignment

1. INTRODUCTION

In many sensor network applications, it is necessary to support multiple missions that may arrive over time and compete for the same sensing resources. For *isotropic* sensing devices recording ambient temperature, for example, the information provided by a single sensor can be used to support any and all missions, given that they lie within its sensing range. A directional or *anisotropic* sensor, such as a camera, however, must be directed to a certain location and hence can in general only support a single mission. Thus *choices must be made* as to which sensors will be assigned to which missions. Given information about available sensors and missions, the network must have a process to choose the "best" assignment of sensors to missions.

An intelligent sensor network should direct its resources to the most important feasible missions, reallocating resources appropriately as missions arrive or depart, while taking care not to waste resources on unsuccessful missions. If there are multiple sensors and multiple missions, it must choose the best matching of sensors to missions. A given sensor may offer different missions varying amounts of information (because of geometry, obstructions, or remaining battery level, for example), or none at all.

Missions may vary in both importance (*profit*) and difficulty (*demand*), and these properties need not be correlated. An ongoing surveillance mission may be expensive but of minor importance, whereas an urgent mission for information about one particular spot may be low-demand but very important. In many applications, partial satisfaction will be no better than zero satisfaction. If the goal of a given

mission is to reconstruct the 3D shape of an object, for example, then this may be accomplished with images from two cameras, but an image from just one camera will be useless. Indeed, accepting the single image could actually be harmful since the drain on the sensor's battery could preclude a future mission that might otherwise have been satisfiable. In our model we consider two profit functions: (1) we only receive profit from missions whose demands are fully met, and (2) we consider profits from ones that reached a preset *threshold*. Hence the problem is to choose the "best" assignment of sensors to missions, in the sense that profits from satisfied missions are maximized.

In some networks, there may simply be a static set of long-term missions, in which case the aspect of time may be eliminated. In other settings, mission arrivals and departures may be infrequent, so that for each block of time, sensor assignment can be solved as a static problem. Even in this static setting, our problem is computationally hard to solve optimally. Thus we use approximation algorithms and heuristics.

A centralized approach to sensor assignment will collect all the relevant information at a central location for decision-making and then distribute assignments. Such an approach can be expensive in terms of communication overhead, however. Another approach is to have nodes[1] make these assignment decisions locally, in a distributed manner, using mission information that is disseminated into the network. While this should decrease communication costs, a centralized algorithm may be able to guarantee better solution.

Since the problem we formulate is (in its most general form) NP-hard even to approximate, we investigate constrained versions for which approximation algorithms exist. First, we bound the number of sensors that may offer contributions to any single mission. This is a reasonable assumption in realistic settings in which sensors have a limited sensing range and the sensors are distributed in such a way as to limit sensing redundancy. Second, we consider a geometric constraint; only sensors within a bounded sensing range from a mission can be assigned to that mission. We also generalize the problem further by allowing a mission to be successful even if its demand is not fully met. We do this by setting a threshold which specifies the minimum fraction of the demand to be met for a mission to succeed. In this case, the mission will not be awarded the full profit but rather a fraction based on its satisfaction level.

Contributions. In this article, we consider the problem of assigning directional sensors to missions in wireless sensor networks. We show NP-hardness and give theoretical results for certain problem variants, including a Δ-approximation greedy algorithm and a shifting-based Polynomial-Time Approximation Scheme (PTAS). We present both centralized and distributed approaches to solving the dynamic problem. In the distributed setting, we give a novel multi-round proposal scheme, which we also adapt to a dynamic setting. We also provide an energy-aware extension to the distributed scheme that extends network lifetime.

Our simulations show that in spite of the theoretical difficulty of the general problem, efficient schemes can perform quite well. The greedy algorithm frequently

[1]In this article we use the terms *sensor* and *node* interchangeably. We use *element* to refer to either a sensor or a mission.

performs near-optimally, obviating the need for the more computationally expensive PTAS. Furthermore, distributed schemes are often competitive with the centralized greedy algorithm, especially in networks with high node density. In our static experiments, the difference in quality between the distributed and centralized solutions is less than 8%, and in some cases the same results are obtained. At the same time, the distributed approach saves as much as 50% of the number of messages. In the dynamic setting, the performance of our adapted distributed scheme closely matches that of the centralized scheme. We also find that adding energy-awareness to the distributed scheme can increase the network lifetime by up to 70%.

The rest of this article is organized as follows. Section 2 discusses some related work in sensor networks and in assignment problems. In Section 3 we state our network model and formally define the sensor assignment problem and study its computational complexity. In Section 4, we study variants of the problem and propose algorithms to solve them. In Section 5 we propose more practical distributed algorithms for the dynamic problem. In Section 6 we detail the network model used in our simulations and present those results. Finally, Section 7 concludes the article.

2. RELATED WORK

Sensor networks. The general problem of choosing sensors to achieve an objective has received sizable attention lately. Several selection objectives have been considered. In [Perillo and Heinzelman 2003; Shih et al. 2006], for example, the goal is to cover the region using few sensors in order to conserve energy. The techniques used range from dividing the sensors in the network into a number of sets and rotating them, activating one at a time [Perillo and Heinzelman 2003], to using Voronoi diagram properties [Aurenhammer 1991] to ensure that the area of a sensor's diagram is as close to its sensing area as possible [Shih et al. 2006]. Another technique used is delayed sensor activation [Lu et al.], in which sensors are initially inactive and are activated according to their coverage contribution. In that way the schemes greedily turn on the sensors with highest probability of successfully sensing the areas of interest. Our problem is similar if we consider that covering a certain area is a mission.

Another related problem is to efficiently locate and track targets such as the works in [Zhao et al. 2002] and [Kaplan 2006]. In [Zhao et al. 2002], for example, the most informative sensor for tracking a target is chosen based on the concept of information gain. This information is then passed on to the next active sensor, which is chosen by considering the target's expected path. Target localization using acoustic sensors is considered by [Kaplan 2006]. An initial active set of two sensors is chosen by exhaustive search, and then additional sensors are added to the active set. The goal there is to minimize the mean squared error of the target location as perceived by the active sensors. Our work, however, is motivated by contention between multiple missions with varying profit values, and therefore focuses on mission selection rather than sensor selection.

There has also been some work on frameworks for single and multiple mission assignment problems. For example, [Byers and Nasser 2000] defines a framework modeling the assignment problem with notions of utility and cost. The goal is to find

a solution that maximizes the utility while staying under a predefined budget. In [Mullen et al.], a market-based method is used in which sensors provide information or "goods" which can be purchased while observing certain budgets. The goal is to maximize the amount of goods delivered without exceeding the budget. A survey of sensor selection and assignment problems, including simple theoretical models of the problem, can be found in [Rowaihy et al. 2007].

The problem we consider here is different from previous work since it considers multiple missions with different priorities. These missions contend for the same set of sensors which calls for resolution mechanisms. We also consider adding energy awareness to our algorithms in order to prolong the network lifetime.

Energy awareness in sensor networks have also been studied in the literature. For example, the authors of [Heinzelman et al. 2000] propose LEACH which is an energy-efficient communication protocol that uses the concept of clusters. In their scheme they rotate the cluster heads in order to evenly distribute the load among them. This is similar to our energy-aware scheme in which we rotate the sensors assigned to a mission for the same purpose. A survey of energy-efficient scheduling mechanisms in sensor networks can be found in [Wang and Xiao 2006].

Algorithms. Although we use the terminology of sensors and missions for concreteness, the problem we are considering can be viewed as a more general problem of resource allocation. An alternative interpretation regards scheduling jobs on unrelated parallel machines. As in other (maximization) scheduling problems [Sung and Vlach 2005], the goal is a schedule that maximizes profit earned from jobs completed, subject to certain constraints. The twist is that each job specifies not the set of *machines* that can perform it, but the set of *families of machines* that can perform it. (A job may be too difficult to be performed by any single machine.) The feasibility constraint is that no machine can be assigned more than one "sub-job".

Our problem also relates to other optimization problems, such as the Bin *Covering* problem, in which the goal is to use a set of items to fill completely as many bins as possible. Sensor-Mission Assignment is a generalization of (weighted) Bin Covering in that an "item" may take up different amounts of space in different "bins". In this way, an analogy can be seen between Sensor-Mission Assignment and the Bin Covering on the one hand and the Multiple Knapsack and the Generalized Assignment (GAP) [Fleischer et al. 2006] problems on the other. While the problem we study in this article does not involve capacity constraints of GAP, in later work we use a GAP-inspired algorithm to solve a budgeted sensor assignment problem [Johnson et al. 2008].

Combinatorial Auctions. In contrast to conventional auctions, in combinatorial auctions players can bid of sets or *combinations* of items (which may entail exponentially many bids). Given a fixed supply of goods, the goal of the winner determination problem is to maximize revenue earned from the sale of disjoint item combinations. Since this is a difficult problem, much of the research focus has been on AI or algorithm-engineering approaches. (See [de Vries and Vohra 2003] for a survey.) Another way of understanding our problem is as a combinatorial auction in which bidders are the missions, the items are the sensors *and* the missions, and

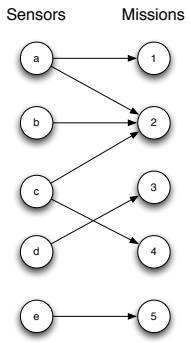


Fig. 1. Modeling the problem as a bipartite graph

each mission places a bid (equal to its own profit) for any set that can be constructed as follows: the set contains the mission itself and some subset of sensors that together satisfy the mission's demand. The language of profits, demands, and edge values, however, allows for a succinct representation of bids.

Many weaker and often fractional models of sensor-mission assignment can be reduced to maximum matching or network flow problems, and thus can be solved optimally in polynomial time [Ahuja et al. 1993].

3. OVERVIEW

In this section we discuss our network model. Then, we formally define the core problem of sensor-mission assignment.

3.1 Network Model

In our network model, we assume a set of static sensors pre-deployed in a field. Missions can arrive and depart over time. By a mission, we mean a primitive sensing task that requires information of a certain type, which may be contributed by one or more deployed sensors. Each mission is defined by a specific geographic location. An example of a mission is video monitoring an area of interest. General missions that cover large areas, such as perimeter monitoring, can be divided into multiple missions each having its own location. The deployed sensors are directional in nature and hence each of them can be assigned to a single mission (i.e. directed to one location). The direction of a sensor can be changed when the assignment is changed. A video camera is a good example of such sensors.

3.2 Core Problem Definition

The problem, which we call *Semi-Matching with Demand* (SMD), is modeled as a weighted bipartite graph whose vertex sets consist of sensors $S = \{S_1, ..., S_n\}$ and missions $\{M_1, ..., M_m\}$ (see Figure 1). A sensor S_i may be able, depending on its type and location, to provide mission M_j with some data. A positively weighted edge (S_i, M_j) means that S_i is applicable to M_j. The weight of the edge (e_{ij}) indicates the utility (or quality of information) that S_i could contribute to M_j if this pairing were chosen. The utility may vary depending on the sensor's type, location or other properties. Also given is a positive-valued demand d_j associated

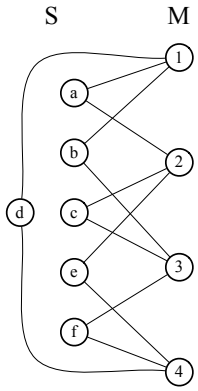


Fig. 2. Integrality gap instance

with each mission M_j, indicating the total utility the mission requires. To simplify the problem we assume that the utility amounts received by a mission are additive. That is the total utility received by a mission is equal to the sum of the utilities provided by sensors assigned to it. While this may be realistic in some settings such as sensing applications in which high-quality measurements can be obtained by e.g. either taking a single high-quality reading or averaging together several lower-quality readings, in others it is not; for our purpose of comparing the different algorithms this assumption is sufficient.

What we seek is a *semi-matching* of sensors to missions, so that (ideally) each mission demand is satisfied. That is, a sensor may be assigned to at most one of the missions to which it is applicable, but a mission can accept utility from multiple sensors. Of course, satisfying all missions may not be feasible; in general, the goal is to maximize a weighted sum of the *satisfied missions*. We assume there is a profit p_j associated with achieving mission M_j. We then seek to maximize the total satisfied profit.

We start by defining the strict version in which no profit is awarded for a partially satisfied mission. Later (in Section 4.3), we relax this requirement and consider a version of the problem in which missions can be partially successful if they reached a minimum threshold of utility.

Unless there is structure to the weights of the sensor-mission edges, for example if they relate to the geometry of node positions, we can assume without loss of generality that each demand is 1. For each mission M_j with demand d_j, simply divide edge value e_{ij} by d_j to obtain an instance with unit demands. Unless otherwise stated, we will assume this normalization henceforth, though it is sometimes convenient to allow for non-unit demands. With this in mind, we define the problem formally.

Instance:. A weighted bipartite graph $G = (S, M, P, E)$, where $S = \{S_1, ..., S_n\}$ is a collection of sensors, $M = \{M_1, ..., M_m\}$ is a collection of missions, $P = \{p_1, ..., p_m\}$ is a collection of positive mission profits, and E is a collection of non-negative weights for the edges $S \times M$.

Goal:. Find a semi-matching $F \subseteq E$ (no two chosen edges share the same *sensor*),

in which $\sum_{M_j \in A} p_j$ is maximized, where $A \subseteq M$ is the set of missions satisfied by F (i.e., $\sum_{(i,j)\in F} e_{ij} \geq 1$ for each $M_j \in A$).

The problem can be formulated as an Integer Program (IP). The IP below employs two sets of decision variables: u_j indicating whether mission M_j is satisfied with the received utility, and x_{ij} indicating whether sensor S_i is assigned to mission M_j. Finding a solution can be seen as a two-step process: decide which missions to satisfy, and then decide how to satisfy them. Each mission M_j has a constraint requiring that the sum of utility received by M_j be at least the value u_j, which is 0 or 1. When $u_j = 0$, this constraint is automatically satisfied. Here is the IP:

$$\begin{array}{ll} \textit{Maximize:} & \sum_j p_j u_j \\ \textit{Such that:} & \sum_{i=1}^{n} x_{ij} e_{ij} \geq u_j, \text{ for each mission } M_j, \\ & \sum_{j=1}^{m} x_{ij} \leq 1, \text{ for each sensor } S_i, \text{ and} \\ & x_{ij} \in \{0,1\}, \text{ for each variable } x_{ij} \text{ and } u_j \in \{0,1\}, \text{ for each variable } u_j \end{array}$$

Note that if we had not normalized to unit demands, the first constraint would be: $\sum_{i=1}^{n} x_{ij} e_{ij} \geq u_j d_j$.

The corresponding Linear Program (LP), relaxes variable constraints from $\{0,1\}$ to $[0,1]$. This allows for fractional profits to be awarded for partially satisfied missions *and* for sensors to be fractionally assigned to multiple missions. This fully fractional version can be solved optimally by Linear Programming. Such an *optimal fractional* solution provides an upper bound on the true optimal solution value.

DEFINITION 1. *Let $IP(I)$ indicate the optimal solution value for a given integer program and let $LP(I)$ indicate the optimal solution value of the LP relaxation, both on problem instance I. We will say that the formulation has integrality gap (at least) c for $c \geq 1$, if $c \leq \frac{|LP(I)|}{|IP(I)|}$ for some problem instance I.*

REMARK 2. *The IP above has unbounded integrality gap, since instances can be constructed in which the optimal solution of the corresponding LP $OPT_{LP} = m/2$ and the optimal solution of the IP $OPT_{IP} = 1$, where m is the number of missions. To create such an instance (see Figure 2), connect a separate sensor to each pair of missions, so that each mission has $m-1$ neighbors, and set all demands to $m-1$ and all profits to 1. Then setting all edge weights to 1/2 will clearly half-satisfy each mission, but only one can be satisfied integrally.*

DEFINITION 3. *Let $|OPT(I)|$ indicate the optimal solution value for a given problem instance(I), and let $|ALG(I)|$ indicate a corresponding approximate solution value. We will say that an algorithm is a* c-approximation *for $c \geq 1$ if $c \leq \frac{|OPT(I)|}{|ALG(I)|}$ for every problem instance I. We omit I below when it is clear from the context.*

PROPOSITION 4. *SMD is NP-hard and at least as hard to approximate as* MAXIMUM INDEPENDENT SET (MIS).

PROOF. Given an MIS graph $G = (V, E)$, an SMD instance is created with a mission M_v for each $v \in V$, with $d_v = deg(v)$ and $p_v = 1$, and a sensor $S_{u,v}$ for each edge $(u, v) \in E$, which offers utility 1 to missions M_u and M_v. Then

Algorithm 1 Δ-Approximation Greedy Algorithm

```
for each mission M_j in order of decreasing P_j
    for each still-available sensor S_i in order of decreasing e_ij
        assign S_i to M_j
        if M_j is satisfied then break
    if M_j is not satisfied then
       return any sensors assigned to it
```

the optimal SMD solution yields the optimal maximum independent set (and both share the same solution quality). □

Since MIS cannot be approximated to factor $|V|^{1-\epsilon}$ unless NP=ZPP [Håstad 1999], neither an optimal nor a nontrivial approximation algorithm for SMD is likely to be forthcoming. Therefore we turn in the next section to easier special cases.

4. PROBLEM VARIANTS AND ALGORITHMS

In this section we discuss a number of problem variants of SMD. First we discuss two special cases which admit approximation algorithms–a degree-bounded version and a geometric variant; then, we study two problem extensions, generalizing to allow a success threshold and generalizing from static to dynamic.

4.1 Degree-bounded Approximation Problem

Because of the difficulty of the approximation problem as defined in the previous section, we constrain it in order to render it more tractable. We assume that the problem instance has *bounded degree*, in the following sense. If a sensor S_i makes a non-zero offer to a mission M_j, then say that S_i is M_j's *neighbor*. Then the assumption is that no mission has more than Δ neighbors, for some small constant Δ. We call this problem Δ-SMD.

A simple greedy algorithm (see Algorithm 1) considers missions in decreasing order of profit. For each mission, the algorithm assigns it available sensors in decreasing order of offer utility, until the mission is satisfied. If the mission does not succeed, then all sensors are returned. The running time of this algorithm is $O(m \log m + mn \log n)$ where m is the number of mission and n is the number of sensors. This algorithm provides an approximation guarantee which we prove below.

We note that the algorithm can be given a distributed implementation, proceeding in rounds, as in the distributed greedy algorithm for the Dominating Set problem of [Jia et al. 2002]. In each round, each pending mission determines whether it is still satisfiable and if so broadcasts its profit value to all other missions within distance $2R_S$. Each pending, satisfiable mission is then assigned its chosen sensors iff it has the highest-profit among such missions in its $2R_S$ neighborhood. Since this is a faithful implementation of the algorithm, we note that the approximation guarantee given below applies to this case.

DEFINITION 5. *Let a* star *consist of a mission and a* minimally satisfying *set of sensors for it. The sensor set is minimal in the sense that no proper subset would*

completely satisfy the mission in question. (Notice that a given mission may in general be part of many stars.) Say that a mission is tight *if it has degree* Δ *and requires all* Δ *sensors in order to be satisfied. Two stars* overlap *if they share one or more sensors, if they share a mission, or both, including the case that the stars are identical.*

PROPOSITION 6. *Algorithm 1 produces a* Δ*-approximation for the* Δ*-SMD problem.*

PROOF. Let OPT be the set of missions satisfied in some optimal solution (with solution quality $|OPT|$), and let ALG be the missions satisfied by Algorithm 1 (with quality $|ALG|$). We want to show that $|OPT| \leq \Delta \cdot |ALG|$, i.e., that

$$\sum_{M_j \in OPT} p_j \leq \sum_{M_{j'} \in ALG} \Delta \cdot p_{j'} \tag{1}$$

To prove Ineq. 1, we account for each term p_j on the left-hand side with one of the terms $\Delta \cdot p_{j'}$ on the right-hand side. For each $M_j \in OPT$, say that M_j *charges to* the highest-profit mission $M_{j'} \in ALG$ whose star overlaps with M_j's star, and write $M_j \in ch(M_{j'})$. (There must be one such $M_{j'}$ with $p_j \leq p_{j'}$ since Algorithm 1 satisfies a maximal set of missions, selected in decreasing order of profit.) Then let $M_{j'}$ be an arbitrary mission in ALG. $M_{j'}$ is either tight or not. (Recall Definition 5.) Suppose tight, in which case that mission has only one star. If $M_{j'} \in OPT$, then only $M_{j'}$ itself charges to $M_{j'} \in ALG$; if $M_{j'} \notin OPT$, then at most Δ stars in OPT can charge to $M_{j'}$ (those that share at least one of its sensors). Now suppose $M_{j'}$ is not tight, so that it contains $\leq \Delta - 1$ sensors. Then at most Δ stars in OPT can charge to $M_{j'}$ (those that share at least one of its sensors, and possibly one that shares its mission). Thus we have

$$\sum_{M_j \in ch(M'_j)} p_j \leq \Delta \cdot p_{j'} \tag{2}$$

By summing Ineq. 2 over all missions in ALG, we obtain Ineq. 1. □

It is easy to construct an example with $\Delta + 1$ missions to show that the Δ bound is tight: let one mission have profit $1 + \epsilon$ and require all Δ sensors; let the rest of the missions have profit 1 and require one sensor each. We note that such examples can be given even for the geometric settings that we discuss below.

Interestingly, when the number of sensors neighboring a mission is less than or equal to two, then the problem can be solved in polynomial time.

PROPOSITION 7. 2-SMD *is in P.*

PROOF. We reduce to the (weighted) maximum matching problem (see Figure 3). The node set for the resulting graph will consist of the 2-SMD instance's sensors and missions. Whenever a mission M_j will be satisfied only by both its neighbors S_{i1}, S_{i2}, draw an edge (S_{i1}, S_{i2}) with weight equal to the mission profit; whenever a mission M_j will be satisfied by a single sensor S_i, draw an edge (S_i, M_j) with weight equal to the mission profit. Now find a maximum weighted matching in this (non-bipartite) graph in polynomial time. Each selected edge corresponds to

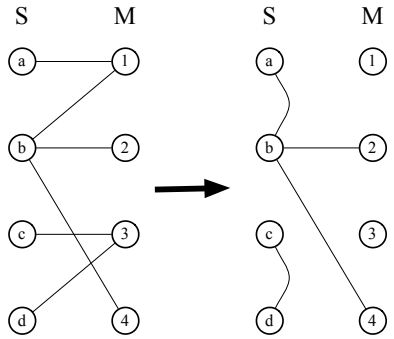


Fig. 3. Converting SMD to graph.

a satisfied mission. It is clear that no sensor or mission will be used more than once. The optimal solution values of the matching graph and the SMD are by construction the same. □

Since the graph of a 2-SMD instance is sparse, the maximum weighted matching can be found in time $O(m^2 \log m)$ [Galil et al. 1986], where m is the number of missions. The time for finding the maximum matching is the dominant component of the running time.

We now relate Δ-SMD and Δ-Set Packing. These problems turn out to be equivalent for small enough Δ. In the Set Packing problem, we are given a family of subsets of a universe of elements. Each subset has a positive weight. The goal is to choose a max-weight family of subsets without using any element more than once. Δ-Set Packing is the variant of Set Packing in which each set has at most Δ elements.[2]

Proposition 8. Δ-SMD *reduces to* Δ-Set Packing, *when* $\Delta = O(\log nm)$.

Proof. The idea of the reduction is that each star in our SMD instance will become a set in the Set Packing instance. (Since a given mission may have degree Δ, it can have $O(2^\Delta)$ many stars. Because of the bound on Δ, however, the resulting Set Packing instance will be at most polynomially larger than the initial SMD instance size.) Specifically, a star with $s < \Delta$ sensors will become an $s+1$-element set containing the star's sensors and mission; a star with Δ sensors will become a Δ-element set containing only the star's sensors. Since a mission can have only one tight star of size Δ, the mission need not be included in the resulting star. Choosing a max-weight family of disjoint sets will now be the same as choosing a max-weight set of disjoint stars. □

An existing local search algorithm from Berman [Berman] gives a $(\frac{\Delta+1}{2}+\epsilon)$-approximation for Δ+1-Claw-Free MIS, which Δ-Set Packing reduces to. (ϵ is a running-time parameter, specifically, a $\frac{k}{(k-1)}\frac{\Delta+1}{2}$ approximation can be found in time polynomial in $O(kn)$.) Hence there is a $\frac{\Delta+1}{2}$ approximation for Δ-SMD (for small Δ). It was recently shown [Hazan et al. 2006] that even for the cardinality

[2]The parameter used is typically k, but we are interested in the case $k = \Delta$.

version, approximating Δ-Set Packing within a factor better than $\frac{\Delta}{\ln \Delta}$ is NP-hard.

4.2 Geometric Problem

We now introduce GeomSMD, a variant in which geometrically inspired constraints are imposed. First, each sensor and mission now lie at a particular point in the plane. Second, we assume sensors have a bounded sensing range (R_S), i.e., e_{ij} can only be non-zero when the distance between S_i and M_j is less than this bound. Without loss of generality, let the sensing range be 1. In this case, every star will lie in a unit disk. We also assume a geometric analog to bounded degree, specifically an upper bound on the number of sensors or missions contained in any unit disk. This constraint will be satisfied automatically if the graph is *drawn in a civilized manner* [Hunt et al. 1998], i.e., so that any two nodes are separated by some minimum distance $\lambda > 0$. In this case, the problem instance will satisfy the Δ-SMD bounded degree requirement for some constant Δ.

The NP-hardness argument below involves the Unit-Disk MIS (UD-MIS) problem [Clark et al. 1990], which is an MIS variant in which the problem instance is the *intersection graph* for a set of unit disks lying in the plane. Equivalently, UD-MIS can be defined so that given a set of points in the plane, two points are connected by an edge iff their distance is strictly less than a global constant. We argue that the NP-hardness proof for UD-MIS also applies to a density-bounded UD-MIS.

Proposition 9. GeomSMD *is strongly NP-hard.*

Proof. We reduce from Planar 3SAT to UD-MIS to GeomSMD. Given the Planar 3SAT instance, we first apply the UD-MIS reduction of [Clark et al. 1990; Marathe et al. 1997], which results in a UD-MIS graph and a number k (there is an independent set of size k iff the formula was satisfiable). It can be verified by consulting the proof of [Clark et al. 1990] that the resulting UD-MIS instance can be drawn with at most O(1) disks per unit square resulting a density-bounded UD-MIS.

Now, we convert the UD-MIS decision-problem instance (G, k) into a GeomSMD decision-problem instance (G', k), by replacing each disk with a mission at the disk's center, and every maximal intersection of disks with a sensor needed by all of them. Since each mission needs all the sensors lying in its disk in order to be satisfied, k missions can be satisfied iff k independent disks can be chosen. Since in the UD-MIS construction each unit square contains at most $O(1)$ such intersections, in the resulting GeomSMD instance, each unit square will contain at most $O(1)$ missions and $O(1)$ sensors, and the sensing distance is respected by construction. Thus Planar 3SAT is reduced to GeomSMD. □

Since GeomSMD is strongly NP-hard, it follows that a *Fully Polynomial-Time Approximation Scheme (FPTAS)* is unlikely for it. A PTAS, however, is possible. To achieve this, we employ the shifting technique originally introduced in [Hochbaum and Maass 1985] which imposes a grid that partitions the region into cells. The portion of the problem instance lying within a given cell has bounded size, which allows the cell to be solved optimally (e.g., by brute force). The portion of the problem instance (in our problem, sensor-mission edges) lying on the

Algorithm 2 Shifting PTAS (error ϵ)

$k \leftarrow \lceil 2/\epsilon \rceil$; $S = \emptyset$
for each $(i, j) \in [0, k)^2$
  lay the mesh with offset (i, j); $S_{ij} \leftarrow \emptyset$
  for each cell C_t within the mesh
    $S_{ij} \leftarrow S_{ij} \cup opt(C_t)$
  if $val(S) < val(S_{ij})$ **then** $S \leftarrow S_{ij}$

boundary of a cell will be discarded, but for a large enough cell size, the perimeter of the cell will be small compared to its area. Although it could happen that much of the problem instance lies near to cell boundaries for a given positioning of the grid, many different grid positions are considered to avoid this problem.

We now give the shifting PTAS[3] (see Algorithm 2), which is similar to the UDMIS [Matsui 1998] PTAS (following the presentation in [Erlebach and Fiala 2001]). For now, assume for simplicity that all points are inside a square region I whose size is polynomial in the input size. Then for a desired error bound ϵ, we can choose parameter $c = \lceil 2/\epsilon \rceil$. Now, we lay a square grid on the plane, carving the region into square cells of size $c \times c$. Each integer pair $(i, j) \in [0, c)^2$ corresponds to a possible offset for the coarse-grain grid, i.e., one of the grid positions to be considered. For a given grid position, we eliminate all sensor-mission edges not fully contained within a single cell. Within any cell, there are $O(\Delta c^2)$ sensors and missions; therefore we can find the optimal assignment restricted to that cell by enumerating all $O((\Delta c^2)^{\Delta c^2})$ possible assignments. The solution for a given offset pair (i, j) is the union of the solutions for the individual cells. We compute the solution for each possible offset pair.

If the points lie in an extremely large region, then the method as stated may not run in polynomial time, since there may be exponentially many cells to check. This can be easily fixed. First, notice that there will be at most polynomially many non-empty cells, which can be found by iterating through the point coordinates. For each non-empty cell, we can "grow" it outward, to obtain a maximally non-empty region. Performing this action on every non-empty cell (i.e., Union-Find) produces a polynomial collection of independent regions. Now simply run the original algorithm on each independent region, rather than on the entire space.

PROPOSITION 10. *Algorithm 2 is a PTAS.*

PROOF. Consider the optimal star-set OPT with total profit P_{opt}. By the Shifting Lemma [Hochbaum and Maass 1985], there must be some vertical offset j that crosses a subset of OPT with total profit at most P_{opt}/c. Similarly, there must be some horizontal offset i that crosses a subset of OPT with total profit at most P_{opt}/c. Therefore the union of the cell-optimal solutions for this (i, j) will be within factor $(1 - 1/c)^2 \geq 1 - 2/c \geq 1 - \epsilon$ of the optimal. □

[3]Although we focus on the plane, it is easy to extend to a fixed higher dimension D.

4.3 Threshold Problem

We now generalize the core problem so that profit may be awarded fractionally, once a threshold is reached. Each mission M_j is still associated with a positive demand value d_j and a positive profit value p_j. The demand may now be interpreted as the total utility the mission desires. Profit for mission M_j indicates the importance of the mission and is awarded based on the percentage of satisfied demand, but only if this percentage reaches a satisfaction threshold T; p_j is the maximum profit receivable for mission M_j.

The goal is again to maximize total profits. We call this problem Threshold SMD. Threshold-1 SMD (i.e., $T = 1$) is simply the original problem. The problem instance and goal are defined as follows:

Instance: A global threshold $T \in [0, 1]$ and a weighted bipartite graph $G = (S, M, P, D, E)$, where $S = \{S_1, ..., S_n\}$ is a collection of sensors and $M = \{M_1, ..., M_m\}$ is a collection of missions; each mission M_j is associated with a profit $\{p_j\}$ and a demand $\{d_i\}$; each edge in $S \times M$ has an edge-weight e_{ij} indicating utility.

Goal: Find a semi-matching $F \subseteq S \times M$ (no two chosen edges share the same *sensor*), in which $\sum_j p_j(u_j)$ is maximized, where u_j is the total utility received by mission M_j divided by demand d_j. The profit functions are defined as follows:

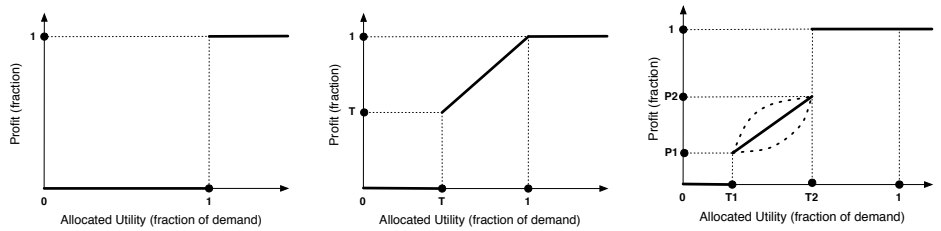
$$p_j(u_j) = \begin{cases} p_j, & \text{if } u_j \geq 1 \\ p_j \cdot u_j, & \text{if } T \leq u_j < 1 \\ 0, & \text{if } u_j < T \end{cases} \tag{3}$$

Finally, the IP formulation is the same as the original SMD, except for the objective function:

$$\textit{Maximize:} \quad \sum_j p_j(u_j)$$

Although the objective function is piecewise linear, it is not concave. In fact, it is neither concave nor convex. Intuitively, the profit function for a single mission is specified by the entire curve in Figure 4(b), as the allocated utility to mission M_j (u_j) (represented by the fraction of met demand) ranges from 0 to infinity. In fact, the objective function is only defined from 0 to 1, as a result of the range constraints on the u_j variables in the linear program. Since this is a maximization problem with a non-concave objective, standard LP and IP techniques do not directly apply. It is important to note, that if our profit function were concave on the region $[T, 1]$, it need not be concave overall.

Although we do not consider them in this article, there are many other profit functions that may warrant study. For the range lying between profit percentages of 0 and 1, natural options include: linear, full profit only for fully met demands, linear only after crossing a threshold, smooth convex (difficult to optimize) and smooth concave (easy to optimize). A more general family of profit functions depends on two thresholds (see Figure 4(c)): before threshold $T1$, there is no partial credit; after threshold $T2$, full credit is received; in between the two thresholds, the profit function could be piecewise convex, linear, or concave, as the application demands, and the profit values at the extremes of this middle range need not equal the corresponding threshold values. Note, however, that strictly speaking, profit



(a) Strict (b) Fractional with threshold (c) Other functions

Fig. 4. Modeling missions profits

functions that go to *full profit* after crossing a threshold can be ignored without loss of generality, since in this case the threshold is simply a lower demand.

The threshold-based problem generalizes both all-or-nothing profits (with T=1) and fully fractional profits (with T=0). When T=1, profit p_j is received only for fully satisfying mission M_j (see Figure 4(a)). In this case, the program reduces to an Integer Program (IP) with mission variable constraints $u_j \in \{0, 1\}$, which is the strict SMD problem. Treating T as part of the problem instance therefore means that this formulation can only be harder than the original strict version (which was shown to be NP-Complete in Section 3). Intuitively, lowering the threshold should make the problem easier. The problem, however, is NP-Complete even with threshold 0.

PROPOSITION 11. *Threshold-0 SMD is* strongly *NP-Complete.*

PROOF. We do a reduction from the 3-PARTITION problem [Garey and Johnson 1979]. Let $A = \{a_1, ..., a_{3m}\}$ be a multiset of positive integers with $\sum_{a \in A} a = mS$, satisfying $\frac{S}{4} < a < \frac{S}{2}$ for each $a \in A$. The goal in 3-PARTITION is to partition the set A into m triples so that each triple sums to S.

The resulting problem instance has m missions $M_1, ..., M_m$, with demands $d_1 = ... = d_m = S$, unit profits, and $3m$ sensors corresponding to the elements of A. Set all weights for edges (S_i, M_j), to a_i. Then the only way to meet all demands is, by construction, to meet them exactly. Thus the 3-Partition instance has an equal-sum m-partition iff in the resulting SMD instance all missions can be *fully* satisfied. Finally, it is clear that an assignment that satisfies all missions can be checked in polynomial time. □

We now discuss how the greedy algorithm given above in Algorithm 1 adapts to the threshold setting (see Algorithm 3). The algorithm will repeatedly satisfy the most *currently profitable* mission, i.e. the mission that can be satisfied with the greatest profit, using the currently available sensors. If $S' \subset S$ is the set of not-yet-assigned sensors (initially $S' = S$) and $u_j = \sum_{S_i \in S'} e_{ij}$, then the profit currently achievable by mission M_j is $p_j(u_j)$. (Of course, it may be that not all sensors are needed to achieve this profit; conversely, if the demand threshold is not met, this profit is 0.) The algorithm repeatedly select a mission M_j of maximum current profitability, and then satisfies it with available sensors (which are removed from S'), in order of decreasing contribution value e_{ij}, until either M_j is *fully* satisfied or

Algorithm 3 Greedy Algorithm for Threshold-SMD

```
while true
    for each available mission M_j
        u_j ← Σ_{S_i unused} e_ij
    j ← arg max_j p_j(u_j)
    if p_j(u_j) = 0 then break
    u_j ← 0
    for each unused S_i in decreasing order of e_ij
        if u_j ≥ d_j or e_ij = 0 then break
        assign S_i to M_j
        u_j = u_j + e_ij
```

all sensors with non-zero offers to M_j have been used. When there are no remaining missions with non-zero current profitability, the algorithm completes. The running time of the algorithm as written is $O(n(m + \log n))$ but it is easy to improve this to $O(mn \log n)$ by updating the u_j values over time rather than computing them from scratch. This greedy algorithm has the same Δ-approximation performance guarantee.

PROPOSITION 12. *Algorithm 3 produces a Δ-approximation for the problem.*

PROOF. The proof of Theorem 6 goes through essentially unchanged. □

When $T = 1$, this algorithm produces the same result as the simpler greedy algorithm of Algorithm 1. We note that this algorithm also can be given a distributed rounds-based implementation, similar to that for Algorithm 1. The only difference is that the value each mission broadcasts to its distance $2R_S$ neighbors is now its current profitability value, rather than its profit value.

For geographic settings with limited sensing range, we can also extend the approximation scheme (PTAS) of the previous subsection to the threshold problem (see [Rowaihy et al. 2007]). Although it is theoretically efficient, we found in our experiments that achieving performance competitive with the greedy algorithm's requires an unreasonable time, and so we do not include its results in this article.

4.4 Dynamic Problem Model

We now provide an orthogonal generalization of the original problem in terms of time to model more realistic scenarios in which missions arrive and depart over time. The problem statement is the same, except that now each mission is associated with a start time and an end time. A mission's demand and maximum profit are constant over time. Awarded profit for a mission is computed at each discrete timestep, based on the satisfaction level at that instant. Total profit for a mission is simply the sum of the instantaneous profits. We do not require that a mission's demand be met over its entire lifetime in order to receive profit. Our profit model is in this sense fractional *in terms of time*. The dynamic version is thus given by essentially the same mathematical program given above except that each variable now has an additional time index.

As a generalization of the static problem, previous hardness results apply also

to the dynamic version. Indeed, a natural strategy for the dynamic problem is to solve the static problem at each timestep.

5. DISTRIBUTED SOLUTIONS

Although centralized algorithms such as the greedy algorithm and the PTAS discussed in the previous section may provide better solutions to the matching problem due to their global view of the field, they can be expensive in terms of communication cost. Because a centralized algorithm requires global information about all sensors in the network such as their locations, utilities to the different missions and other stats, the number of messages required to be sent to the base station can become very large, especially in dense networks. This communication cost becomes even higher for dynamic environments in which missions arrive and depart at different points in time, requiring the base station to continually gather information about sensors in the field.

To avoid this cost, we consider distributed algorithms to solve the problem. In such an approach, a *mission leader* is selected for each mission. This should be a sensor that is close to the mission's location. Finding the leader can be done using a geographic-based routing techniques such as [Bose et al. 2001] or [Karp and Kung 2000]. (In our simulations, we simply choose the closest unused sensor to serve in this role.) The leaders are informed about the missions' demands, profits and locations by the base station. Then they run a local protocol to match nearby sensors to their respective missions. We assume that the contribution a sensor can provide to a mission is a function of the geographic distance between them and hence only nearby sensors are considered. In this section we consider a multi-round proposal algorithm that works on static settings, which we then adapt to dynamic cases. We also propose an energy-aware extension to the dynamic algorithm which helps in prolonging the network lifetime.

5.1 Multi-Round Proposal Algorithm (MRPA)

In this algorithm, each mission leader advertises its mission information (demand, profit and location) to the nearby sensors by means of broadcast. If the advertisement message needs to be sent over multiple hops then neighboring sensors rebroadcast the message so their neighbors can hear it. The number of hops over which the advertisement message is sent depends on the relation between the communication range, which is the maximum distance over which two sensors can communicate, and the sensing range. If the sensing range is larger than the communication range then sensors that are further away should be notified.

When a nearby sensor hears such an advertisement message for one or more missions, it sends a single proposal to the mission it perceives to be its best match. The ranking of missions is based on the profit of a mission weighted by the fraction of the mission's demand that the sensor can satisfy. Using the notation introduced in Section 3, sensor S_i ranks mission M_j according to B_{ij} where $B_{ij} = e_{ij}/d_j \times p_j$. The leader, on the other hand, selects proposing sensors in a greedy fashion according to their contribution to the mission. If the leader of a mission does not select a proposing sensor, then in the next round the sensor proposes to the next mission on its list. This algorithm consists of a series of proposal-reply rounds. The more rounds, the better the matching may be. However, as the number of rounds

Algorithm 4 Multi-Round Proposal

For leader of mission M_j**:**
  for each round k
    wait for proposing sensors
    sort proposing sensors in decreasing order of e_{ij}
    while $u_j < d_j$
      assign the next S_i (in sorted order) to M_j
      $u_j = u_j + e_{ij}$
    if M_j's satisfaction level $< \alpha(k)$ **then**
      release all sensors assigned to M_j

For sensor S_i**:**
  for each round k
    if S_i is *unassigned* **then**
      receive advertisement from all nearby missions
      ignore any mission already considered
      $j \leftarrow \arg\max_j B_j = (e_{ij}/d_j) \times p_j$
      send proposal to M_j
    else break

increases, the communication cost grows and we may obtain diminishing returns. Hence, there is a trade-off between solution quality and communication cost.

Since our aim is to achieve the highest profit from *successful* missions, we use a mechanism to prevent missions that will never be fully successful from holding up sensors that can help other missions. In each round, mission leaders assess the satisfaction level of their missions. If the level is not greater than an increasing threshold ($\alpha(k)$ for round k) then the mission is assumed to be unattainable and all its sensors are released. The threshold is initialized to a fixed value (e.g. 10% satisfaction) and incremented each round (e.g. by 10%). After a sufficient number of rounds, it will reach T, the preset value of the success threshold, at which time all missions that are not yet successful release their sensors. The rising threshold therefore yields two benefits: increasing the chance that the most satisfied missions will become fully satisfied and preventing sensors from spending their energy on missions that will not reach the minimum success threshold (for which we receive no profit). Algorithm 4 summarizes the steps taken by mission leaders.

We now analyze the both the runtime complexity and message complexity of Algorithm 4. We assume that the number of sensors is n, number of missions is m and number of rounds is k. The running time for sensor S_i is $O(km)$ as in each round a sensor may consider up to m missions. The mission leader's complexity is $O(n \log n)$ as in each round the mission leader has to sort up to n proposing sensors and select the best ones but over all the rounds each sensor proposes to a mission at most one time. If we assume that advertisement messages are only broadcast to immediate neighbors, then the message complexity of the algorithm including both sides, the sensor and the mission, is $O(m + kn)$ as we have m

advertisement messages, $O(kn)$ proposals by sensors and $O(n)$ reply messages from mission leaders.

5.2 Dynamic Proposal Algorithm (DPA)

MRPA is designed to work in the cases in which we have prior knowledge about the missions. The multiple rounds allow it to work well even if there are several missions that compete for the same sensing resources. Since it requires complete knowledge about missions it can also work in an environment that does not need fast response to new missions. An example would be a system that batches together a number of missions and runs the matching algorithm periodically, e.g. every few hours. However, in a fully dynamic setting, the network needs to have a fast response to incoming and outgoing missions. By handling missions as they arrive, we expect to encounter less competition for sensing resources, and so we can opt for a lighter-weight algorithm that does not need multiple rounds to complete. We call this algorithm the *Dynamic Proposal Algorithm* or DPA.

As in MRPA each mission has a leader which advertises mission information to nearby sensors. A sensor that hears this announcement can be in one of two states: (1) *not assigned*, in which case it proposes to the mission with its utility or (2) *assigned to a mission*, in which case the sensor calculates its effective profit for both missions (which is the B_{ij} value found above) and chooses either to stay with the current mission or to propose to the new mission, depending on which value is higher. So, this algorithm allows a mission to preempt an ongoing mission to increase the overall profit of the network.

After the mission leader collects the proposals, it tries first to satisfy the mission demands with sensors in the *not assigned* state by greedily picking sensors with highest utility. If these sensors are not sufficient, it tries to steal sensors from other ongoing missions, i.e., it chooses from sensors in the *assigned* state. If the collected utility at that time is at least T, then the mission leader sends assignment messages to the respective sensors which start collecting information to support the mission. If a sensor is selected which preempts an existing mission, the procedure below is followed.

Let us say that a new mission, M_j with leader L_j, started in an area close to an ongoing mission, M_k with leader L_k. If a sensor S_i that is currently assigned to M_k decides that its contribution will generate better profit if it is assigned to M_j, it notifies L_k of its intention. L_k then tries to find one or more sensors to replace S_i. If no such sensor(s) are found, the leader will agree on the reassignment as long as its current satisfaction level does not drop below T, which will cause the mission to fail. If the release of S_i will bring the allocated utility to lower than T then the reassignment is temporarily denied. If sensor S_i is *critical* to the new mission M_j, i.e. without it the mission will fail, a second test is performed. If the current profit value of M_j with S_i assigned to it is greater than that of M_k, the leader of M_k will release its hold on S_i and agree on the reassignment even if it will cause its own mission to fail. The reassignment becomes final once S_i is selected by M_j. Only at that time are the replacement sensor(s) activated. Algorithm 5 summarizes sensor S_i's response to the reception of an advertisement message.

To reduce both the interruption of ongoing missions and the communication overhead, preemption is limited to one level. That is if mission M_j preempted

Algorithm 5 Dynamic Proposal

For leader of mission M_j**:**
send advertisement of mission M_j
sort proposing sesnors in decreasing order of e_{ij}
while $u_j < d_j$
assign the next S_i (in sorted order) to M_j
$u_j = u_j + e_{ij}$

For sensor S_i**:**
receive advertisement of mission M_j
if S_i is not active **then**
propose to M_j with offer e_{ij}
else if S_i is assigned to M_k with $\frac{e_{ik}}{d_k} \times p_k < \frac{e_{ij}}{d_j} \times p_j$ **then**
ask current leader L_k for reassignment
propose to M_j with offer e_{ij} **only if** current leader agrees, i.e. **if**
L_k can find replacements **or**
M_k will still reach its threshold without replacements **or**
S_i is critical to M_j and M_j obtains greater profit than M_k
if selected for mission M_j **then**
notify leader of M_k to assign replacement(s)
else
continue operation on M_k

mission M_k, M_k will try to satisfy its demand with only available sensors and will not try to steal sensors that are already assigned. When a mission ends, the leader sends out a message to announce that the mission has ended and all assigned sensors are released.

Because the system is dynamic, missions that are not fully satisfied after the first assignment process will retry to obtain more sensors after some time. However, they only retry if there will be more available sensors. This can happen in the case when a nearby mission terminates and has its sensors released. This information can either be learned from the base station or by overhearing the message announcing the end of a mission.

We now analyze the runtime complexity and message complexity of Algorithm 5. If we assume that the number of sensors is n and number of missions is m, then the running time for sensor S_i is $O(m)$ as it may consider up to m missions. The reassignment takes constant time for the sensor. The mission leader's complexity, on the other hand, is $O(n \log n)$ as the mission leader has to sort up to n proposing sensors and select the best ones. Again, if we assumes that mission advertisement messages are only broadcast to immediate neighbors, then the message complexity of the algorithm including both sides, the sensor and the mission, is $O(m+n)$ as we have m advertisement messages, $O(n)$ proposals by sensors, and $O(n)$ replies from mission leaders. As the algorithm does not need several rounds to complete, we see a saving of factor k in the number of messages sent by the sensors over MRPA.

5.3 Energy-aware Dynamic Proposal Algorithm (EDPA)

A drawback of DPA is that it does not consider the remaining energy in sensors when making assignment decisions. It selects a sensor based only on the utility it provides. However, the energy level of such a sensor may have been depleted over time and using it for sensing will consume its remaining energy leading to its death. This can happen while other sensors around it that provide lower utility may still have full energy. With this observation, we extend DPA to make it energy-aware and call it the *Energy-aware Dynamic Proposal Algorithm* (EDPA) Algorithm. EDPA uses information about the proposing sensors current remaining energy level to make better assignment decisions which would ultimately lead to a longer lifetime.

Instead of using the utility of a sensor to the mission alone to make assignment decision, EDPA uses a function (f) of utility (U) and fraction of remaining energy (E). We define:

$$f(U, E) = U \times E^{\beta} \tag{4}$$

where β is a design parameter. If β is zero, EDPA becomes DPA and hence only the utility is considered. A higher value of β gives more preference to sensors with more remaining energy.

To consume energy more evenly among sensors, after the initial assignment, possible sensor candidates for a mission send periodic updates to the leader including their current energy levels. The leader then checks if it has consumed energy unevenly among all sensors that can contribute to the mission. At that time, the leader may choose to change the assignments of sensors by reapplying the decision function f. The periodic updates increase the communication overhead, but as will be shown in Section 6, this increase is not very high. As expected, this algorithm works better in a dense network in which there are many sensors that apply to a mission and hence more choices are available to the leader.

6. PERFORMANCE EVALUATION

To evaluate our algorithms we built a simulator in Java and tested them using randomly generated problem instances. We perform two sets of experiments. For both cases we test and compare the performance of the centralized greedy scheme and the distributed schemes. In the first set we test the static case. That is, we assume that the entire problem instance, including all sensors and all missions, is given simultaneously. In the second set we consider the dynamic problem in which missions arrive over time and depart after spending a certain amount of time being active. For the dynamic case, we also show how EDPA can use information about remaining energy of sensors to make better selection decisions to improve network lifetime.

6.1 Assumptions

Each mission has a demand, an abstract value of the amount of sensing resources it requires, which is exponentially distributed with an average of 2 and a maximum of 6. Also associated with each mission is a profit value, which measures its importance. The profit is also exponentially distributed, but with an average of 1. This

simulates common scenarios in which many missions demand few sensing resources and a smaller number demand more resources. The same applies to profit. The profit obtained from a successful mission M_j is equal to $p_j(u_j)$ as defined in Section 4.3. We consider a mission successful if it receives at least 50% of its demanded utility from allocated sensors (i.e. $T = 0.5$). Each sensor can only be assigned to a single mission.

The utility that sensor S_i provides to mission M_j is defined as a function of the distance D_{ij} between them. Many types sensors exhibit some kind of quality deterioration or signal attenuation based on distance. In order to evaluate their utilities to missions, we assume that all sensors know their geographical locations. Formally, the potential utility contribution is:

$$e_{ij} = \begin{cases} \frac{1}{1+D_{ij}^2/c,} & \text{if } D_{ij} \leq R_S \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

where $R_S = 30m$ is the global sensing range. This models typical signal attenuation which is an inverse function of the distance squared. Note that this utility function is only used for testing in our experiments and is not a property of our algorithms; it is not meant to model the exact behavior of any sensor. In our experiments we set $c = 60$.

Sensors are deployed in uniformly random locations in a $400m \times 400m$ field (the base station is located in the center of the left edge). Missions also are created in uniformly random locations in the field. The communication range of sensors is set to $40m$. When sensors are deployed we ensure that the network is connected. If a randomly created instance is not connected, it is discarded by the simulator.

As an upper-bound we include the profit results for the optimal fractional solution, described in Section 4. This is the optimal solution for the relaxed fractional problem in which sensors may divide their utility between multiple missions and all fractional profits are counted, regardless of whether the success threshold is reached or not. Our algorithms' performance may be judged in comparison to this upper bound. We have also find the optimal sensor assignments for several configurations.

Finally, we assume perfect communication channels with no errors or collisions. Hence, each message is sent only once and its delivery is guaranteed. All messages have the same size and hence only the number is considered when studying the communication overhead. For messages that travel over multiple hops, each hop counts as a single message in the total. When a broadcast message (e.g. mission advertisements by mission leaders or base station) is sent, it is received by all one-hop neighboring sensors and so we count it as one. All routing decisions are made based on a pre-configured routing table that follows the shortest path to destination.

6.2 Static Scenarios

6.2.1 *Setup.* In this experiment all missions occur simultaneously, with the same start and end times. We fix the number of sensors in the field to 500 and vary the number of missions from 10 to 100. In the following results we show the average of 10 runs.

For the centralized approaches, we show results for the greedy scheme. Based on the field size and the sensing range and ignoring the boundaries, the expected

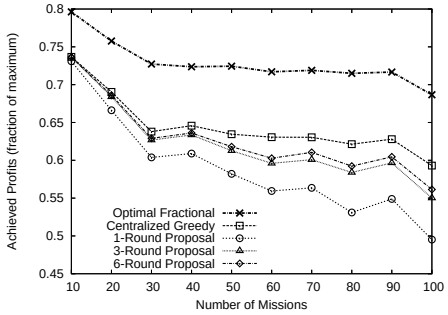


Fig. 5. Achieved profits

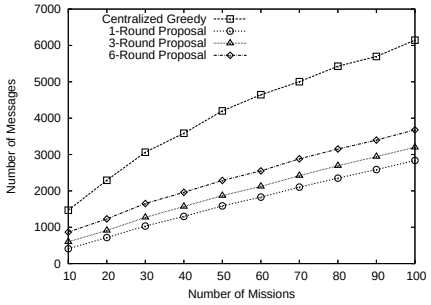


Fig. 6. Number of messages

number of sensors within sensing range of a mission is $\bar{\Delta} = 8.8$. Note that the maximum mission degree Δ defined in Section 4.1 will in general be greater than $\bar{\Delta}$, and so the greedy approximation guarantee will be at most $1/8.8 \approx 11\%$ of the optimal. The approximation ratio achieved empirically by the greedy algorithm is in fact much stronger than this worst-case guarantee. In our experiments, it achieves at least 85% of the optimal, based on a comparison to an upper bound on the optimal solution value.

Since such a comparison is necessarily conservative, to gain better insight on the algorithm's performance we tested it on several smaller problem instances for which we were able to find the actual optimal sensor assignments. The difference in the achieved profits between optimal and greedy was between 0 and 1.2%. Note that, as mentioned in Section 4.3, we do not show the results of the shifting PTAS algorithm because we found that its performance gain over the simple greedy algorithm to be insignificant relative to its very high computational cost.

For the distributed approach, we show results for MRPA with one round, three rounds and six rounds. These results illustrate the trade-off between solution quality and communication overhead. The growing threshold α for a mission to release sensors in MRPA is set to 10% in the first round and is increased by 10% for each subsequent round until it reaches T, or 50%. Advertisement messages are sent from mission leaders to all sensors within two hops.

6.2.2 *Results.* The first set of results (Figure 5) shows the fraction of the maximum mission profits achieved by the different schemes compared to the optimal fractional profits. The maximum profit is the sum of all missions profits. Note that when we create missions we do not guarantee that all of them can be satisfied by available sensors, i.e. some missions might not be satisfiable even if all available sensors are assigned to them. This can happen if a mission is located on the edge of the field with few surrounding sensors, for example. Furthermore, even if each mission is individually satisfiable, in a sparse network there may not be enough nearby sensors to mutually satisfy all missions.

The greedy centralized solution performs best followed by the 6-round proposal. However, its advantage lessens as the density of the sensors increases. For a same-

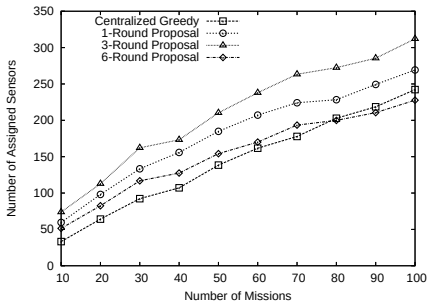


Fig. 7. Number of assigned sensors

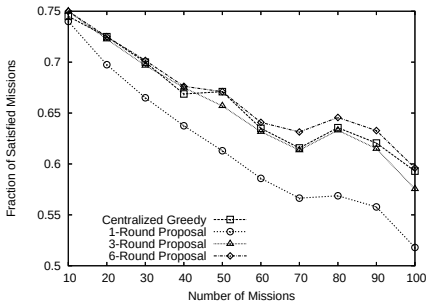


Fig. 8. Fraction of satisfied missions

sized field with 1000 sensors deployed (not shown in the Figures), all the curves except the one-round proposal become nearly aligned. This is expected since there are more sensors that can be assigned to the different missions. We note that the improvement in MRPA when going from a single round to 3 rounds is very pronounced. However, the improvement gained when jumping to 6 rounds is less apparent and may not justify the necessary communications overhead.

Figure 6 shows the communication overhead of the different schemes. As expected, the centralized scheme has the highest overhead. With MRPA, as the number of rounds increase more messages are exchanged. The savings in number of exchanged messages becomes more evident if we consider a dynamic system, in which sensor-mission utility values can change over time. In this case, the centralized scheme needs to collect information about current sensor utility values before running a matching process for each new mission that arrives. Distributed schemes, on the other hand, require the exchange of fewer messages since information about utility values only needs to be sent to the leader of the new mission, which is just a few hops away.

As the number of sensors and missions increase, the distributed schemes encounter greater overlap in the areas of local matching, which leads to more exchanged messages. This is true because as densities of sensors and missions increase more sensors can contribute to each mission and at the same time each sensor can contribute to more missions.

The number of sensors assigned can be used as a proxy for the amount of energy used (shown in Figure 7). The number of sensors used by the centralized scheme is very close to that of the 6-round proposal scheme. For MRPA, few sensors are assigned if one round is used. This is because this setting does not allow sensors that were rejected in this round to re-propose to other missions. With 3 rounds, the mission leaders release sensors that are not useful which allows these sensors to re-propose to other missions and hence the number of assigned sensors becomes higher. With 6 rounds most of the unused sensors are released which brings the number of the assigned sensors down to about that of the centralized scheme. Note that the schemes do not use all the available sensors even when the number of missions is large. This is because some sensors are not within the sensing range

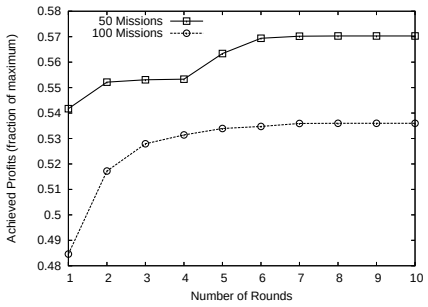


Fig. 9. Achieved profits vs rounds

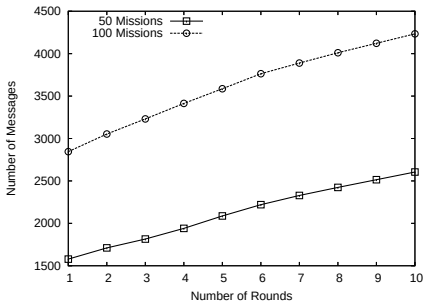


Fig. 10. Number of messages vs rounds

of any mission and hence remain idle. When we consider the results in this figure we should take into consideration the achieved profits in Figure 5. For example, even though the centralized greedy algorithm assigns close to 250 sensors when 100 missions are present, it achieves less than 60% of the possible profits.

Finally, Figure 8 show the fraction of satisfied missions for the different schemes. Note that our goal is not to maximize this number but rather to achieve the highest profit. The centralized scheme is successful in achieving the highest profit values but not always the largest fraction of satisfied missions. The 6-round proposal achieves higher fraction when the number of missions is large. This happens because the greedy centralized algorithm assigns sensors to missions in order of profit and hence may stop satisfying missions after a certain point because no sensors are longer available. So, even though the fraction of satisfied missions is less than that of the 6-round proposal, the amount of profits is higher as more profitable missions were picked. Because of its lack of global view, the 6-round proposal algorithm is satisfying more missions but with lower profits. Between the three multi-round schemes tested, there is a significant increase in the fraction of satisfied missions between one round and three rounds and less improvement between three rounds and six rounds.

From the above results, we see that the distributed schemes perform well. The difference in achieved profit values compared to the centralized scheme we tested is less than 8%. At the same time, it saves as much as 50% of the messages.

6.2.3 *Number of rounds.* To study the effect of the number of rounds in MRPA, we fix the number of sensors to 500 and perform two experiments with 50 and 100 missions. Figure 9 shows the relation between achieved profits and number of rounds. As can be expected, achieved profits initially increase with the number of rounds. However, the additional gains beyond 8 rounds is small since by that time all attainable missions have reached their success threshold. Figure 10 shows the communication overhead which increases linearly with the number of rounds.

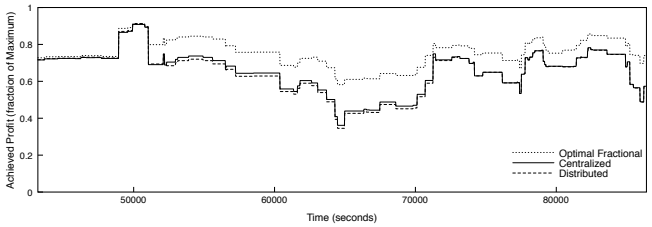


Fig. 11. Trace of network performance ($\lambda = 3$ missions/hr)

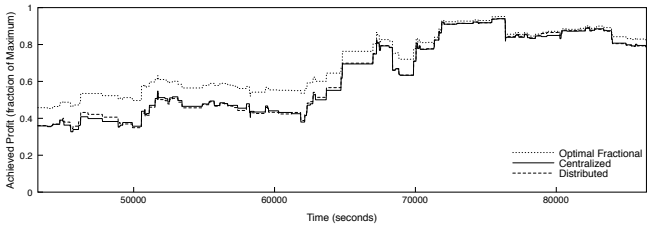


Fig. 12. Trace of network performance ($\lambda = 6$ missions/hr)

6.3 Dynamic Scenarios

6.3.1 *Setup.* Now we show the performance of our distributed proposal scheme in a dynamic setting in which missions arrive over time. The same aforementioned assumptions apply here. Moreover, we assume that missions arrive according to a Poisson distribution. The mission lifetimes are selected according to an exponential distribution with an average lifetime of one hour and a maximum of four hours. The exponential distribution is heavy-tailed which models realistic scenarios in which there are many short-lived missions and few long-lived ones. Although a centralized scheme is impractical in dynamic scenarios, due to high communication cost, we include its results to measure the performance of the distributed scheme. We assume that the centralized scheme is rerun for each mission arrival and departure.

6.3.2 *Results.* We compare the performance of DPA to results achieved by the greedy centralized scheme and the optimal fractional solution. Figures 11 and 12 show a trace of the achieved network profits during a period of 12 hours for arrival rate, $\lambda = 3$ missions/hour and 6 missions/hour respectively with a network of 500 sensors. We start the simulation at time zero and start recording the trace after 10 hours. As can be seen from the figures, the performance of DPA, which utilizes local information about missions, is very close to that of the centralized scheme.

Figures 13 and 14 show the average performance over a period of 50 hours (averaged over 10 runs) for a network with 500 sensors and 1000 sensors. Figure 13 shows the average achieved profits per unit of time (fraction of maximum) as the mission arrival rate is varied. We see that both the centralized and DPA perform almost equally. Note that, as expected, with a larger number of sensors the network can achieve higher profits.

The communication overhead of DPA is shown in Figure 14. The number of

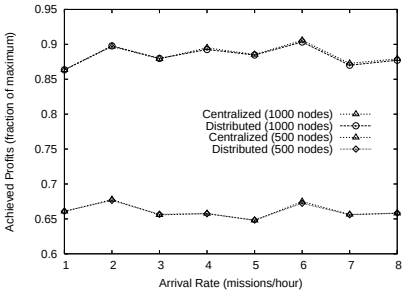


Fig. 13. Avg. achieved profits (per unit of time) in period of 50 hours

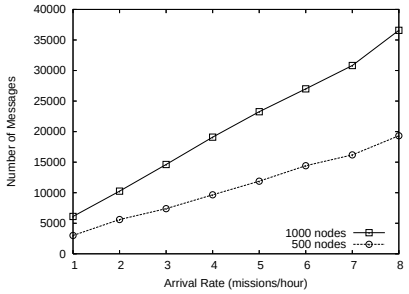


Fig. 14. Number of exchanged messages in period of 50 hours

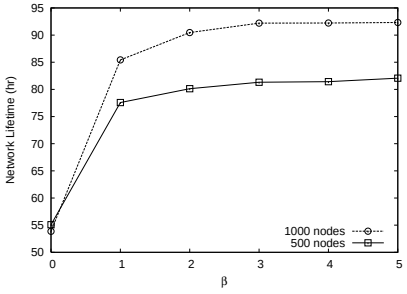


Fig. 15. Network lifetime (in hours) using EDPA

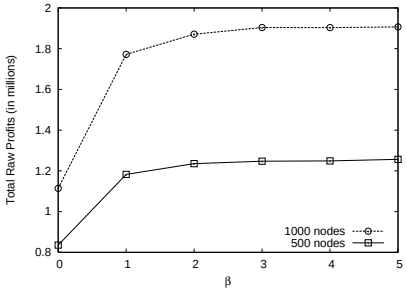


Fig. 16. Total raw profits using EDPA

messages grows linearly as the number of missions in the network increases. We do not show number of messages for the centralized scheme as this value will be very large compared to DPA. For each mission arrival and departure, the base station needs to collect information from all sensors that can contribute to the arriving mission to get status updates. The average number of messages exchanged per mission is around 40 for 500 sensors and 80 for 1000 sensors. This includes all the messages needed to advertise the mission and make all the assignment decisions including reassignments.

6.3.3 *Network Lifetime.* Figures 15-17 show the results for EDPA in a network a network of 500 sensors and 1000 sensors (average of 10 runs). The mission arrival rate is set to 6 missions/hour. All sensors start with energy to support 10 hours of continues sensing. We consider only energy consumed for sensing. Sensor reassignment is performed every 20 minutes to balance energy consumption. Choosing a smaller period may yield a more uniform assignment but will have a larger communication overhead.

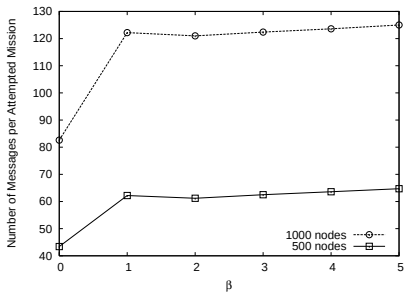


Fig. 17. Messages per mission using EDPA

We define the network lifetime as the time until the first sensor dies. Figure 15 shows the lifetime of the network for different values of β, a parameter used to control dependence on remaining energy. Recall that when $\beta = 0$, EDPA becomes DPA. The results show that when EDPA is used, network lifetime increases by 50% and 70%, for networks with 500 sensors and 1000 sensors, respectively. The increase is notable when β goes from 0 to 1, i.e when we start taking energy into account. After that point, the increase in lifetime is not very pronounced. The denser the network the more options the assignment scheme has and hence it is able to achieve longer lifetime.

Figure 16 shows the total achieved profits (sum of profit in every second during lifetime) for the different values of β. This can be thought of as the area under the curves for the two schemes shown in Figures 11 and 12. As expected, the profits increase when lifetime increases. But when there are more sensors the profit increase in more prominent due to the longer lifetime and the fact that more sensors allows for more satisfaction for missions.

Due to periodic updates, the communication overhead increases when EDPA is used ($\beta > 0$). Figure 17 shows the average number of messages per each attempted mission. We see an average increase of around 50% for both network sizes. Although the percentage increase may seem high, the actual number of exchanged messages per mission (around 60 for 500 sensors and 120 for 1000 sensors) is relatively small, especially if we consider that this number includes all the messages exchanged to setup a mission and is amortized over its lifetime.

7. CONCLUSION

In this article, we introduced a sensor-mission matching problem. We analyzed its complexity, defined constrained versions, and presented approximation algorithms for them. We showed that it is strongly NP-hard, even in the special case of zero success threshold. As such, we turned to approximation algorithms and heuristics. We considered a greedy centralized algorithm and several distributed schemes, which we then adapted to the dynamic setting. Our simulation results demonstrate that in spite of the theoretical difficulty, under realistic conditions the greedy algorithm performs near-optimally and the distributed schemes perform almost as well.

In our algorithms and experiments, we make the simplifying assumption that utilities from different sensors can be combined additively. Clearly this assumption applies to some but far from all settings. Building on the work presented here, we are currently investigating joint utility functions in which the combined utility of multiple sensors could be either greater than or less than the sum of the individual utilities. The joint utility setting is more complex and will require substantially revised algorithms. The algorithms presented here represent a first step towards a solution to this more general problem, but we also believe the additive utility setting is interesting in its own right. In the future, we also plan to perform additional experiments, using real-world data from applications in which additivity or similarly assumptions apply.

ACKNOWLEDGMENTS

This research was sponsored by US Army Research laboratory and the UK Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. 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 the US Army Research Laboratory, the US Government, the UK Ministry of Defence, or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon. We would also like to thank the anonymous reviewers for their valuable comments.

REFERENCES

Ahuja, R., Magnanti, T., and Orlin, J. 1993. *Network Flows*. Prentice Hall.

Aurenhammer, F. 1991. Voronoi diagrams - a survey of a fundamental geometric data structure. *ACM Computing Surveys*.

Berman, P. A d/2 approximation for maximum weight independent set in d-claw free graphs. In *Proceedings of SWAT 2000*. 214–219.

Bose, P., Morin, P., Stojmenovic, I., and Urrutia, J. 2001. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks 7,* 6, 609–616.

Byers, J. and Nasser, G. 2000. Utility-based decision-making in wireless sensor networks. In *Proceedings of MobiHoc '00*.

Clark, B., Colbourn, C., and Johnson, D. 1990. Unit disk graphs. *Discrete Math 86*, 165–177.

de Vries, S. and Vohra, R. 2003. Combinatorial auctions: a survey. *INFORMS J. on Computing 15-3*, 284–309.

Erlebach, T. and Fiala, J. 2001. Independence and coloring problems on intersection graphs of disks. *manuscript*.

Fleischer, L., Goemans, M. X., Mirrokni, V. S., and Sviridenko, M. 2006. Tight approximation algorithms for maximum general assignment problems. In *Proceedings of SODA 2006*. 611–620.

Galil, Z., Micali, S., and Gabow, H. 1986. An O(EV log V) algorithm for finding a maximal weighted matching in general graphs. *SIAM J. Comput. 15(1)*, 120–130.

Garey, M. and Johnson, D. 1979. *Computers and Intractability: A Guide to the Theory of NP-Colmpleteness*. Freeman.

Håstad, J. 1999. Clique is hard to approximate within $n^{1-\varepsilon}$. *Acta Mathematica 182*, 105–142.

Hazan, E., Safra, S., and Schwartz, O. 2006. On the complexity of approximating k-set packing. *Computational Complexity 15(1)*, 20–39.

Heinzelman, W., Chandrakasan, A., and Balakrishnan, H. 2000. Energy-efficient communication protocols for wireless microsensor networks. In *Proc. Hawaaian Int'l Conf. on Systems Science*.

Hochbaum, D. and Maass, W. 1985. Approximation schemes for covering and packing problems in image processing and VLSI. *J. ACM 32(1)*, 130–136.

Hunt, H., Marathe, M., Radhakrishnan, V., Ravi, S., Rosenkrantz, D., and Stearns, R. 1998. NC-approximation schemes for NP- and PSPACE-hard problems for geometric graphs. *J. Algorithms 26(2)*, 238–274.

Jia, L., Rajaraman, R., and Suel, T. 2002. An efficient distributed algorithm for constructing small dominating sets. *Distributed Computing 15(4)*, 193–205.

Johnson, M. P., Rowaihy, H., Pizzocaro, D., Bar-Noy, A., Chalmers, S., La Porta, T., and Preece, A. 2008. Frugal sensor assignment. In *DCOSS '08*.

Kaplan, L. 2006. Global node selection for localization in a distributed sensor network. *IEEE Transactions on Aerospace and Electronic Systems 42,* 1 (January), 113–135.

Karp, B. and Kung, H. 2000. Greedy perimeter stateless routing for wireless networks. In *Proceedings of MobiCom '00*.

Lu, J., Bao, L., and Suda, T. Coverage-aware sensor engagement in dense sensor networks. In *Proceedings of the International Confernece on Embedded and Ubiquitous Computing - EUC 2005*.

Marathe, M., Radhakrishnan, V., Hunt, H., and Ravi, S. 1997. Hierarchically specified unit disk graphs. *Theor. Comput. Sci. 174(1-2)*, 23–65.

Matsui, T. 1998. Approximation algorithms for maximum independent set problems and fractional coloring problems on unit disk graphs. In *Proceedings of the Japan Conference on Discrete and Computational Geometry*. 194–200.

Mullen, T., Avasarala, V., and Hall, D. L. Customer-driven sensor management. *IEEE Intelligent Systems 21,* 2 (Mar/April 2006), 41–49.

Papadimitriou, C. H. and Yannakakis, M. 1991. Optimization, approximation, and complexity classes. *J. Comput. System Sci. 43*, 425–440.

Perillo, M. and Heinzelman, W. March 2003. Optimal sensor management under energy and reliability constraints. In *Proceedings of the IEEE Conference on Wireless Communications and Networking*.

Rowaihy, H., Eswaran, S., Johnson, M. P., Verma, D., Bar-Noy, A., Brown, T., and La Porta, T. 2007. A survey of sensor selection schemes in wireless sensor networks. In *SPIE Defense and Security Symposium*.

Rowaihy, H., Johnson, M. P., Bar-Noy, T. B. A., and La Porta, T. 2007. Assigning Sensors to Competing Missions (long version). Tech. Rep. NAS-TR-0080-2007, Network and Security Research Center, Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA, USA. October.

Shih, K., Chen, Y., Chiang, C., and Liu, B. June 2006. A distributed active sensor selection scheme for wireless sensor networks. In *Proceedings of the IEEE Symposium on Computers and Communications*.

Sung, S. C. and Vlach, M. 2005. Maximizing weighted number of just-in-time jobs on unrelated parallel machines. *J. Scheduling 8-5*, 453–460.

Wang, L. and Xiao, Y. 2006. A survey of energy-efficient scheduling mechanisms in sensor networks. *Mobile Networks and Applications 11,* 5, 723–740.

Zhao, F., Shin, J., and Reich, J. 2002. Information-driven dynamic sensor collaboration. *IEEE Signal Processing Magazine 19,* 2 (March), 61–72.