Frugal Sensor Assignment

Matthew P. Johnson[1] Hosam Rowaihy[2] Diego Pizzocaro[3]
Amotz Bar-Noy[1] Stuart Chalmers[4] Thomas La Porta[2] Alun Preece[3]

[1] Dept. of Computer Science, Graduate Center, City University of New York, USA
[2] Dept. of Computer Science and Engineering, Pennsylvania State University, USA
[3] School of Computer Science, Cardiff University, UK
[4] Department of Computing Science, University of Aberdeen, UK

Abstract. When a sensor network is deployed in the field it is typically required to support multiple simultaneous missions, which may start and finish at different times. Schemes that match sensor resources to mission demands thus become necessary. In this paper, we consider new sensor-assignment problems motivated by frugality, i.e., the conservation of resources, for both static and dynamic settings. In general, the problems we study are NP-hard even to approximate, and so we focus on heuristic algorithms that perform well in practice. In the static setting, we propose a greedy centralized solution and a more sophisticated solution that uses the Generalized Assignment Problem model and can be implemented in a distributed fashion. In the dynamic setting, we give heuristic algorithms in which available sensors propose to nearby missions as they arrive. We find that the overall performance can be significantly improved if available sensors sometimes refuse to offer utility to missions they could help based on the value of the mission, the sensor's remaining energy, and (if known) the remaining target lifetime of the network. Finally, we evaluate our solutions through simulations.

1 Introduction

A sensor network deployed for monitoring applications may be tasked with achieving multiple, possibly conflicting, missions. Although certain types of sensors, such as seismic sensors, can receive data from their surroundings as a whole, other sensor types, such as cameras, are directional. In these cases, the direction of each sensor, and thus the mission it serves, must be chosen appropriately.

A given sensor may offer different missions varying amounts of information (because of geometry, obstructions, or utility requirements), or none at all. Missions, on the other hand, may vary in importance (or *profit*), amount of resources they require (or *demand*), and duration. In some but not all applications, it may be preferable to do one thing well than to do many things badly, that is, to fully satisfy one mission rather than give a small amount of utility to several. Given all currently available information, the network should choose the "best" assignment of the available sensors to the missions.

In this paper, we examine new sensor-assignment problems motivated by frugality, i.e., the conservation of resources. We consider two broad classes of environments: static and dynamic. The *static* setting is motivated by situations in which different users are granted control over the sensor network at different times. During each time period, the current user may have many simultaneous missions. While the current user

will want to satisfy as many of these as possible, sensing resources may be limited and expensive, both in terms of equipment and operational cost. In some environments, replacing batteries may be difficult, expensive, or dangerous. Furthermore, a sensor operating in active mode (i.e., assigned to a mission) may be more visible than a dormant sensor, and so in greater danger of being destroyed. Therefore, we give each mission in the static problem a budget so that no single user may overtax the network and deprive future users of resources. This budget serves as a constraint in terms of the amount of resources that can be allocated to a mission regardless of profit.

Our second environment is a *dynamic* setting in which missions may start at different times and have different durations. In these cases explicit budgets may be too restrictive because we must react to new missions given our current operating environment, i.e. the condition of the sensors will change over time. Instead, we use battery lifetime as a metric to capture network lifetime and evaluate a trade-off between cost in terms of network lifetime versus mission profit before assigning sensors.

In the dynamic setting we consider two cases. First, we assume no advanced knowledge of the target network lifetime, i.e. we do not know for how long the network will be required to operate. We call this the *general dynamic setting*. Second, we consider the case in which we have knowledge of the target network lifetime, i.e. the network is needed for a finite duration. We call this the *dynamic setting with a time horizon*. By using a trade-off of lifetime versus profit instead of a hard budget, we can use the knowledge of remaining energy of the sensors and the target network lifetime, if known, to adjust the aggressiveness with which sensors accept new missions.

Our contributions. We consider several sensor-assignment problems, in both budget-constrained static settings and energy-constrained dynamic settings. In the static setting, we give an an efficient greedy algorithm and a multi-round proposal algorithm whose subroutine solves a Generalized Assignment Problem (GAP), as well as an optimal algorithm for a simplified 1-d setting. In the dynamic setting, we develop distributed schemes that adjust sensors' eagerness to participate in new missions based on their current operational status and the target network lifetime, if known. We find in the static setting that in dense networks both algorithms perform well, with the GAP-based algorithm slightly outperforming the greedy. In the dynamic setting, we find that knowledge of energy level and network lifetime is an important advantage: the algorithm given both of these significantly outperforms the algorithm using only the energy level and the algorithm that uses neither. When both the energy and target lifetime are used we can achieve profits 17% - 22% higher than if only energy is used.

The rest of this paper is organized as follows. Section 2 discusses some related work. In Section 3 we review the sensor-mission assignment problem and summarize its variations. In Sections 4 and 5 we propose schemes for the static and dynamic settings, whose performance we evaluate in Section 6. Finally, Section 7 concludes the paper.

2 Related Work

Assignment problems have received sizable attention, in both experimental and theoretical communities. In wireless sensor networks, there has been some work in defining frameworks for single and multiple mission assignment problems. For example, [2] de-

fines a framework for modeling the assignment problem by using the notions of utility and cost. The goal is to find a solution that maximizes the utility while staying under a predefined budget. In [7], a market-based modeling approach is used, with sensors providing information or "goods". The goal is to maximize the amount of goods delivered without exceeding the sensor's budget.

The authors of [6, 8, 11], for example, solve the coverage problem, which is related to the assignment problem. They try to use the fewest number of sensors in order to conserve energy. The techniques used range from dividing nodes[5] in the network into a number of sets and rotating through them, activating one set at a time [8], to using Voronoi diagram properties to ensure that the area of a node's region is as close to its sensing area as possible [11]. Sensor selection schemes have also been proposed to efficiently locate and track targets. For example, [13] uses the concept of information gain to select the most informative sensor to track a target. In [4], the author attempts target localization using acoustic sensors. The goal is to minimize the mean squared error of the target location as perceived by the active nodes. A survey of the different sensor selection and assignment schemes can be found in [9].

The Generalized Assignment Problem (GAP) [3] is a generalization of the Multiple Knapsack Problem in which the weight and value of an item may vary depending on the bin in which it is placed. There is a classical FPTAS [12] for the core knapsack problem which performs a dynamic programming procedure on a discretization of the problem input. If, for example, the knapsack budget value is not too large, then the DP can find the optimal solution in polynomial time. A stricter version of the static sensor-assignment problem was formalized as Semi-Matching with Demands (SMD) in [1, 10]. In that formulation, profits are awarded only if a certain utility threshold is met, but no budgets are considered. The static problem we study here is a common generalization of these two previous problems, incorporating both budgets and a profit threshold.

3 Sensor-Mission Assignment Problems

With multiple sensors and multiple missions, sensors should be assigned in the "best" way. This goal is shared by all the problem settings we consider. There are a number of attributes, however, that characterize the nature and difficulty of the problem. In this section, we briefly enumerate the choices available in defining a particular sensor-mission assignment problem. In all settings we assume that a sensor can be assigned to one mission only, motivated e.g. by directional sensors such as cameras.

3.1 Static Setting

First consider the static setting. Given is a set of sensors $S_1, ..., S_n$ and a set of missions $M_1, ..., M_m$. Each mission is associated with a utility demand d_j, indicating the amount of sensing resources needed, and a profit p_j, indicating the importance of the mission. Each sensor-mission pair is associated with a utility value e_{ij} that mission j will receive if sensor i is assigned to it. This can be a measure of the quality of information that a

[5] We use the terms *node* and *sensor* interchangeably.

max:	$\sum_{j=1}^{m} p_j(y_j)$
s.t.:	$\sum_{i=1}^{n} x_{ij}e_{ij} \geq d_j y_j$, for each M_j,
	$\sum_{i=1}^{n} x_{ij}c_{ij} \leq b_j$, for each M_j,
	$\sum_{j=1}^{m} x_{ij} \leq 1$, for each S_i,
	$x_{ij} \in \{0,1\}\ \forall x_{ij}$ and
	$y_j \in [0,1]\ \forall y_j$

Table 1. MP **P** for static setting

max:	$\sum_t \sum_{j=1}^{m} p_j(y_{jt})$
s.t.:	$\sum_{i=1}^{n} x_{ijt}e_{ij} \geq d_j y_{jt}$, for each M_j and t,
	$\sum_{j=1}^{m} x_{ijt} \leq 1$, for each S_i and time t,
	$\sum_t \sum_{j=1}^{m} x_{ijt} \leq B$, for each S_i,
	$x_{ijt} \in \{0,1\}\ \forall x_{ijt}$ and
	$y_{jt} \in [0,1]\ \forall y_{jt}$

Table 2. MP **P'** for dynamic setting

sensor can provide to a particular mission. To simplify the problem we assume that the utility amounts received by a mission (u_j) are additive. While this may be realistic in some settings, in others it is not; we make this simplifying assumption. Finally, a budgetary restriction is given in some form, either constraining the entire problem solution or constraining individual missions as follows: each mission has a budget b_j, and each potential sensor assignment has cost c_{ij}. All the aforementioned values are positive reals, except for costs and utility, which could be zero. The most general problem is defined by the mathematical program (MP) **P** shown in Table 1.

A sensor can be assigned ($x_{ij} = 1$) at most once. Profits are received per mission, based on its satisfaction level (y_j). Note that y_j corresponds to u_j/d_j within the range [0,1] where $u_j = \sum_{i=1}^{n} x_{ij}e_{ij}$. With strict profits, a mission receives exactly profit p_j iff $u_j \geq d_j$. With fractional profits, a mission receives a fraction of p_j proportional to its satisfaction level y_j and at most p_j. More generally, profits can be awarded fractionally, but only if a fractional satisfaction threshold T is met, i.e.:

$$p_j(u_j) = \begin{cases} p_j, & \text{if } u_j \geq d_j \\ p_j \cdot u_j/d_j, & \text{if } T \leq u_j/d_j \\ 0, & \text{otherwise} \end{cases}$$

When $T = 1$, program **P** is an integer program; when $T = 0$, it is a mixed integer program with the decision variables x_{ij} still integral. The edge values e_{ij} may be arbitrary non-negative values or may have additional structure. If sensors and missions lie in a metric space, such as line or plain, then edge values may be based in some way on the distance D_{ij} between sensor i and mission j. In the *binary sensing* model, e_{ij} is equal to 1 if distance D_{ij} is at most the sensing range, r, and 0 otherwise. In another geometric setting, e_{ij} may vary smoothly based on distance, such as $1/(1 + D_{ij})$.

Similarly, the cost values c_{ij} could be arbitrary or could exhibit some structure: the cost could depend on the sensor involved, or could e.g. correlate directly with distance D_{ij} to represent the difficulty of moving a sensor to a certain position. It could also be unit, in which case the budget would simply constrain the number of sensors. Even if profits are unit, demands are integers, edge values are 0/1 (though not necessarily depending on distance), and budgets are infinite, this problem is NP-hard and as hard to approximate as Maximum Independent Set [1]. If we also add the restriction that sensors and missions lie in the plane and that 0/1 edge utility depends on distance (i.e., the edges form a bipartite unit-disk graph), the problem remains NP-hard [1].

In certain one-dimensional settings, however, the problem can be solved optimally in polynomial-time by dynamic programming. These settings may be motivated by e.g. coastline or national border surveillance. See appendix for details.

3.2 Dynamic Setting

In the dynamic setting we have options similar to the above except that now missions do not have explicit budgets and sensor assignments do not have explicit costs. What constraints the assignment problem is the limited energy that sensors have. We also have an additional time dimension. In this setting, each sensor has a battery size B, which means that it may only be used for at most B timeslots over the entire time horizon. Also, missions may arrive at any point in time and may last for any duration.

If a sensor network is deployed with no predetermined target lifetime, then the goal may be to maximize the profit achieved by each sensor during its own lifetime. However, if there is a finite target lifetime for the network, the goal becomes to earn the maximum total profits over the entire time horizon. We assume that the profit for a mission that lasts for multiple timeslots is the sum of the profits earned over all timeslots during the mission's lifetime. The danger of any particular sensor assignment is then that the sensor in question might somehow be better used at a later time. Therefore the challenge is to find a solution that competes with an algorithm that knows the characteristics of all future missions before they arrive. The general dynamic problem is specified by the mathematical program (MP) **P'** shown in Table 2.

If *preemption* is allowed, i.e. a new mission is allowed to preempt an ongoing mission and grab some of its sensors, then in each timeslot we are free to reassign currently used sensors to other missions based on the arrival of new missions without reassignment costs. In this case, a long mission can be thought of as a series of unit-time missions, and so the sensors and missions at each timeslot form an instance of the NP-hard static (offline) problem. If preemption is forbidden, then the situation for the online algorithm is in a way simplified. If we assume without loss of generality that no two missions will arrive at exactly the same time, then the online algorithm can focus on one mission at a time. Nonetheless, the dynamic problem remains as hard as the static problem, since a reduction can be given in which the static missions are identified with dynamic missions of unit length, each starting ϵ after the previous one. In fact, we can give a stronger result, covering settings both with and without preemption.

Proposition 1. *There is no constant-competitive algorithm for the online dynamic problem,* even assuming that all missions are unit-length and non-overlapping.

Proof. First consider the problem with $B = 1$, i.e. sensors with a battery lifetime of one timeslot. Suppose at time 1, there is M_1 with $p_1 = \epsilon$ requiring S_1 (i.e. otherwise unsatisfiable); we must choose the assignment because there may be no further missions, yet at time 2 there may be a M_2 with profit $p_2 = 1$. This yields the negative result for $B = 1$. Next suppose $B = 2$. Then consider the following example: at time 1, M_1 with $p_1 = \epsilon$ requires S_1, so we must assign because there may be no future missions; at time 2, M_2 with $p_2 = \exp(\epsilon)$ requires S_1, so we must assign for the same reason; at time 3, M_3 with $p_2 = \exp(\exp(\epsilon))$ requires S_1, but it is now empty, and the algorithm fails. This construction can be extended to arbitrary B. □

As a generalization of the static problem, the offline version of dynamic problem is again NP-hard to solve optimally; moreover, the general online version cannot be solved with a competitiveness guarantee.

4 Static Setting

In this section we describe two algorithms to solve the static-assignment problem: *Greedy* and *Multi-round Generalized Assignment Problem (MRGAP)*. The former requires global knowledge of all missions to run and hence is considered to be centralized whereas the latter can be implemented in both centralized and distributed environments which makes it more appealing to sensor networks.

4.1 Greedy

The first algorithm we consider (Algorithm 1 shown in Table 3) is a greedy algorithm that repeatedly attempts the highest-potential-profit untried mission. Because fractional profits are allowed only beyond the threshold percentage T, this need not be the mission with maximum p_j. For each such mission, sensors are assigned to it, as long the mission budget is not yet violated, in decreasing order of cost-effectiveness, i.e. the ratio of edge utility for that mission and the sensor cost. The running time of the algorithm is $O(mn(m + \log n))$. No approximation factor is given for this efficiency-motivated algorithm since, even for the first mission selected, there is no guarantee that its feasible solution will be found. This by itself is an NP-hard 0/1 knapsack problem, after all.

4.2 Multi-round GAP (MRGAP)

The idea of the second algorithm (Algorithm 2 shown in Table 4) is to treat the missions as knapsacks that together form an instance of the Generalized Assignment Problem (GAP). The strategy of this algorithm is to find a good solution for the problem instance *when treated as GAP*, and then to do "postprocessing" to enforce the lower-bound constraint of the profit threshold, by removing missions whose satisfaction percentage is too low. Releasing these sensors may make it possible to satisfy other missions, which suggests a series of rounds. In effect, missions not making "good" progress towards satisfying their demands are precluded from competing for sensors in later rounds.

Cohen et al. [3] give an approximation algorithm for GAP which takes a knapsack algorithm as a parameter. If the knapsack subroutine has approximation guarantee $\alpha \geq 1$, then the Cohen GAP algorithm offers an approximation guarantee of $1 + \alpha$. We use the standard knapsack FPTAS [12], which yields a GAP approximation guarantee of $2 + \epsilon$. Because GAP does not consider lower bounds on profits for the individual knapsacks, which is an essential feature of our sensor-assignment problem, we enforce it using the postprocessing step.

The algorithm works as follows. The threshold is initialized to a small value, e.g. 5%. Over a series of rounds, a GAP solution is found based on the current sensors and missions. After each round, missions not meeting the threshold are removed, and their sensors are released. Any sensors assigned to a mission that has greater than 100% satisfaction, and which can be released without reducing the percentage below 100%, are released. We call such sensors *superfluous*. Sensors assigned to missions meeting the threshold remain assigned to those missions. These sensors will not be considered in the next round, in which the new demands and budgets of each mission will become the remaining demand and the remaining budget of each one of them. Finally, the threshold

while true **do**
  for each available M_j
    $u_j \leftarrow \sum_{S_i(unused)} e_{ij}$;
  $j \leftarrow \arg\max_j p_j(u_j)$;
  if $p_j(u_j) = 0$ **break**;
  $u_j \leftarrow 0$; $c_j \leftarrow 0$
  for each unused S_i in decr. order of e_{ij}/c_{ij}
    if $u_j \geq d_j$ or $e_{ij} = 0$
      break;
    if $c_j + c_{ij} \leq b_k$
      assign S_i to M_j;
      $u_j \leftarrow u_j + e_{ij}$;
      $c_j \leftarrow c_j + c_{ij}$;

Table 3. Algorithm 1: Greedy

initialize set of missions $M \leftarrow \{M_1 \ldots M_m\}$;
initialize global threshold $T \leftarrow 0.05$;
for $t = 0$ to T step 0.05
  run the GAP algorithm of [3] on M and the unassigned sensors;
  in the resulting solution, release any superfluous sensors;
  if M_j's satisfaction level is $< t$, for any j
    release all sensors assigned to M_j;
    $M \leftarrow M - \{M_j\}$;
  if M_j is completely satisfied OR has no remaining budget, for any j
    $M \leftarrow M - \{M_j\}$;

Table 4. Algorithm 2: Multi-Round GAP

is incremented, with rounds continuing until all sensors are used, all missions have succeeded or been removed, or until the actual success threshold T is reached. The GAP instance solved at each round is defined by the following linear program:

max: $\sum_j \sum_i p_{ij} x_{ij}$ (with $p_{ij} = p_j \cdot e_{ij}/\hat{d_j}$)
s.t.: $\sum_{S_i(unused)} x_{ij} c_{ij} \leq \hat{b_j}$, for each remaining M_j,
$\sum_{M_j(remaining)} x_{ij} \leq 1$, for each unused S_i, and $x_{ij} \in \{0, 1\}\ \forall x_{ij}$

Here $\hat{d_j}$ is the remaining demand of M_j, that is, the demand minus utility received from sensors assigned to it during previous rounds. Similarly, $\hat{b_j}$ is the remaining budget of M_j. The concepts of demand and profit are encoded in the gap model as $p_{ij} = p_j \cdot e_{ij}/\hat{d_j}$. This parameter represents the fraction of demand satisfied by the sensor, scaled by the priority of the mission. In each GAP computation, we seek an assignment of sensors that maximizes the total benefit brought to the demands of the remaining mission.

One advantage of MRGAP is that it can be implemented in a distributed fashion. For each mission there can be a sensor, close to the location of the mission, that is responsible for running the assignment algorithm. Missions that do not contend for the same sensors can run the knapsack algorithm simultaneously. If two or more missions contend for the same sensors, i.e. they are within distance $2r$ of each other, then synchronization of rounds is required to prevent them from running the knapsack algorithm at the same time. To do this, one of the missions (e.g. the one with the lowest id) can be responsible for broadcasting a synchronization message at the beginning of each new round. However, since r is typically small compared to the size of the field, we can expect many missions to be able to do their computations simultaneously.

The total running time of the algorithm depends on the threshold T and step value chosen as well as the density of the problem instance, which will determine to what extent the knapsack computations in each round can be parallelized.

5 Dynamic Setting

We have shown in Section 3.2 that the dynamic problem is NP-hard to solve and that without assuming any aditional constraints there are no competitive solutions that can provide guaranteed performance. In this section, we therefore propose heuristic-based schemes to solve the dynamic sensor-mission assignment problem. These schemes are similar in their operation to the dynamic proposal scheme that we have proposed in [10] but with a new focus. Rather than maximizing profit by trying to satisfy all available missions, we focus here on maximizing the profit over network lifetime by allowing the sensors to refuse the participation in missions they deem not worthwhile.

We deal with missions as they arrive. A *mission leader*, a node that is close to the mission's location, is selected for each mission. Finding the leader can be done using a geographic-based routing techniques [5]. The mission leaders are informed about their missions' demands and profits by a base station. Then they run a local protocol to match nearby sensors to their respective missions. Since the utility a sensor can provide to a mission is limited by a sensing range, only nearby nodes are considered. The leader advertises its mission information (demand and profit) to the nearby nodes (e.g. two-hop neighbors). The number of hops the advertisement message is sent over depends on the relation between the communication range and the sensing range of sensors.

When a nearby sensor hears such an advertisement message, it makes a decision either to propose to the mission and become eligible for selection by the leader or to ignore the advertisement. The decision is based on the current state of the sensor (and the network if known) and on potential contribution to mission profit that the sensor would be providing. We assume knowledge of the (independent) distributions of the various mission properties (namely, demand, profit and lifetime), which can be learned from historical data. To determine whether a mission is worthwhile, a sensor considers a number of factors: (1) the mission's profit, relative to the maximum profit, (2) the sensor's utility to the mission, relative to the mission's demand, (3) the sensor's remaining battery level, (4) the remaining target network lifetime, if known.

After gathering proposals from nearby sensors, the leader selects sensors based on their utility offers until it is fully satisfied or there are no more sensor offers. The mission (partially) succeeds if it reaches the success threshold; if not, it releases all sensors.

Since we assume all distributions are known, the share of mission profit potentially contributed by the sensor (i.e. if its proposal is accepted) can be compared to the expectation of this value. Based on previous samples, we can estimate the expected mission profit $E[p]$ and demand $E[d]$. Also, knowing the relationship between sensor-mission distance and edge utility, and assuming a uniform distribution on the locations of sensors and missions, we can compute the expected utility contribution $E[u]$ that a sensor can make to a typical mission *in its sensing range*. We use the following expression to characterize the expected partial profit a sensor provides to a typical mission:

$$E\left[\frac{u}{d}\right] \times \frac{E[p]}{P} \tag{1}$$

We consider two scenarios. In the first, the target network lifetime is unknown, i.e. we do not know for how long will the network be needed. In this case, sensors choose

missions that provide higher profit than the expected value and hence try to last longer in anticipation of future high profit missions. In the second, the target network lifetime is known, i.e. we know the duration for which the network will be required. Sensors, in this case, take the remaining target network lifetime into account along with their expected lifetime when deciding whether to propose to a mission. In the following two subsections we describe solutions to these two settings.

5.1 Energy-aware Scheme

In this scheme, the target lifetime of the sensor network is unknown. For a particular sensor and mission, the situation is characterized by the actual values of mission profit (p) and demand (d) and by the utility offer (u), as well as the fraction of the sensor's remaining energy (f). For the current mission, a sensor computes the following value:

$$\frac{u}{d} \times \frac{p}{P} \times f \tag{2}$$

Each time a sensor becomes aware of a mission, it evaluates expression (2). It makes an offer to the mission only if the value computed is greater than expression (1). By weighting the actual profit of a sensor in (2) by the fraction of its remaining battery value, the sensors start out eager to propose to missions but become increasingly selective and cautious over time, as their battery levels decrease. The lower a sensor's battery gets, the higher relative profit it will require before proposing to a mission. Since different sensors' batteries will fall at different rates, in a dense network we expect that most feasible missions will still receive enough proposals to succeed.

5.2 Energy and Lifetime-aware Scheme

If the target lifetime of the network is known, then sensors can take it into account when making their proposal decisions. To do this, a sensor needs to compute what we call the *expected occupancy time*, denoted by t_α, i.e. the amount of time a sensor expects to be assigned to a mission during the remaining target network lifetime. To find this value we need to determine how many missions a given sensor is expected to see. Using the distribution of mission locations, we can compute the probability that a random mission is within a given sensor's range. If we combined this with the remaining target network lifetime and arrival rate of missions we can find the expected number of missions to which a given sensor will have the opportunity to propose. So, if the arrival rate and the (*independent*) distributions of the various mission properties are known, we can compute t_α as follows:

$$t_\alpha = \tau \times \lambda \times g \times \gamma \times E[l]$$

where:

- τ is the remaining target network lifetime, which is the initial target network lifetime minus current elapsed time.
- λ is the mission arrival rate.

- $g = \pi r^2/A$ is the probability that a given mission is within sensing range (assumes uniform distribution on missions). r is the sensing range and A is the area of the field in which sensors are deployed.
- $E[l]$ is the expected mission lifetime.
- γ is the probability that a sensor's offer is accepted. *Computing* this value would imply a circular dependency. It was chosen a priori to be 0.25.

For each possible mission, the sensor now evaluates an expression which is modified from (2). The sensor considers the ratio between its remaining lifetime and its expected occupancy time. So, if t_b is the amount of time a sensor can be actively sensing, given its current energy level, the expression becomes:

$$\frac{u}{d} \times \frac{p}{P} \times \frac{t_b}{t_\alpha} \tag{3}$$

If the value of expression (4) is greater than that of expression (1), then the sensor proposes to the mission. If the sensor's remaining target lifetime is greater than its expected occupancy time, the sensor proposes to *any* mission since in this case it expects to survive until the end of the target time. The effect on the sensor's decision of weighting the mission profit by the ration (t_b/t_α) is similar to the effect weighting the fraction of remaining energy (f) had in expression (2); less remaining energy, all things being equal, make a sensor more reluctant to propose to a mission. As the network approaches the end of its target lifetime, however, this ratio will actually increase, making a sensor more willing to choose missions with profits less than what it "deserves" in expectation. After all, there is no profit at all for energy conserved past the target lifetime.

6 Performance Evaluation

In this section we discuss the result of different experiments used to evaluate our schemes. We implemented the schemes in Java and tested them on randomly generated problem instances. We report the results of two sets of experiments. In the first set, we test the static setting, in which all missions occur simultaneously. In the second set, we consider the dynamic setting, in which missions arrive, without warning, over time and depart after spending a certain amount of time being active.

6.1 Simulation Setup

Each mission has a demand, which is an abstract value of the amount of sensing resources it requires, and a profit value, which measures its importance (the higher the profit the more important). The profit obtained from a successful mission M_j is equal to $p_j(u_j)$ as defined in Section 3. Each sensor can only be assigned to a single mission. Once assigned, the full utility of the sensor is allocated to support the mission. We consider a mission to be successful if it receives at least 50% of its demanded utility from allocated sensors (i.e. $T = 0.5$).

We assume that mission demands are exponentially distributed with an average of 2 and a minimum of 0.5. Profits for the different missions are also exponentially distributed with an average of 10 and a maximum of 100. This simulates realistic scenarios

in which many missions demand few sensing resources and a smaller number demand more resources. The same applies to profit. The simulator filters out any mission that is not individually satisfiable, i.e. satisfiable in the absence of all other missions. For a sufficiently dense network, however, we can expect there to be few such impossible missions.

The utility of a sensor S_i to a mission M_j is defined as a function of the distance, D_{ij}, between them. In order for sensors to evaluate their utilities to missions, we assume that all sensors know their geographical locations. Formally the utility is:

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

where r is the global sensing range. This follows typical signal attenuation models in which signal strength depends inversely on distance squared. In our experiments, we set $c = 60$ and $r = 30m$.

Nodes are deployed in uniformly random locations in a $400m \times 400m$ field. Missions 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 (that is any sensor can communicate with any other sensor possibly over multiple hops). If a randomly created instance is not connected, it is ignored by the simulator. We do not consider communication overhead other than noting the fact that distributed schemes will typically have lower communication cost than centralized schemes and hence are preferable in that sense. We have actually studied the communication overhead of similar schemes in [10].

6.2 Static Setting

In this experiment, all missions occur simultaneously. We fix the number of sensors in the field and vary the number of missions from 10 to 100. Each sensor has a uniformly random cost in $[0, 1]$, which is the same no matter to which mission it is assigned. The associated cost can represent the actual cost in real money or a value that represents the risk of the sensor being discovered in hostile environments if it is activated, etc. Each mission has a budget, drawn from a uniform distribution with an average of 3 in the first experiment and varies from 1 to 10 in the second. In the following results we show the average of 20 runs.

The first series of results shows the fraction of the maximum mission profits achieved by the different schemes. The maximum profit is the sum of all missions profits. We show the profits for the greedy algorithm, Multi-Round GAP (MRGAP), and an estimate of the optimal value, running on two classes of sensor networks, sparse (250 nodes), and dense (500 nodes) (Figures 1 and 2, respectively). The estimate on the optimal bound is obtained by solving the LP relaxation of program **P**, in which all decision variables are allowed to take on fractional values in the range $[0, 1]$, and the profit is simply fractional based on satisfaction fraction, i.e. $p_j y_y$ for mission M_j with no attention paid to the threshold T. The MRGAP scheme, which recall can be implemented in a distributed fashion, achieves higher profits in all cases than does the greedy scheme

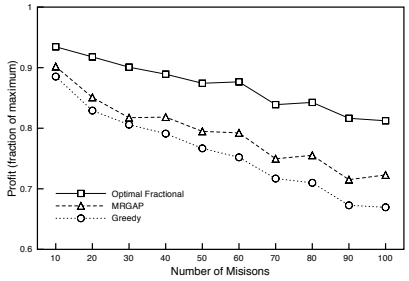

Fig. 1. % of max. profit achieved (250 nodes)

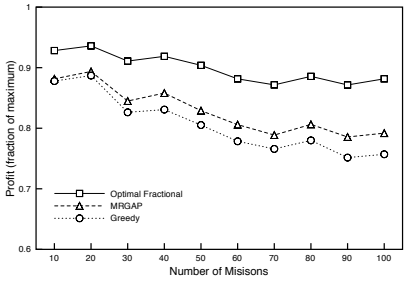

Fig. 2. % of max. profit achieved (500 nodes)

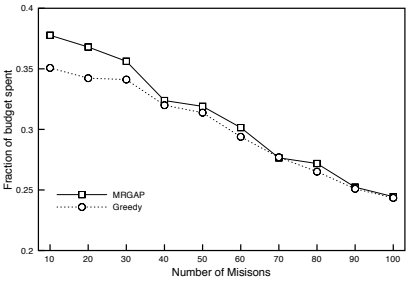

Fig. 3. Fraction of spent budget (250 nodes)

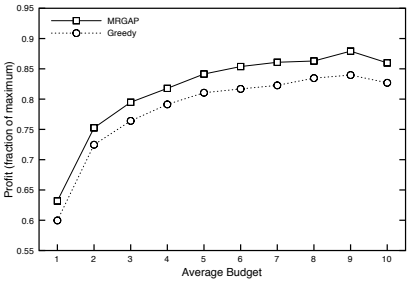

Fig. 4. Varying the average budget (250 nodes)

which is strictly centralized (because missions have to be ordered in terms of profit). The difference, however, is not very large.

Figure 3 shows the fraction of the total budget each scheme spent to acquire the sensing resources it did in a network with 250 nodes. The MRGAP scheme achieves more profits than the greedy algorithm and spends a modest amount of additional resources. The fraction of remaining budget is significant (more than 60% in all cases), which suggests that either successful missions had higher budgets than they could spend on available sensors or that unsuccessful missions had lower budgets than necessary and hence they were not able to reach the success threshold and their budgets were not spent. When the number of missions is large this can be attributed to the fact that there simply were not enough sensors due to high competition between missions. We performed another set of experiments, in which the number of missions was fixed at 50 and average budget given to missions was varied between 1 and 10. Figure 4 shows the results for a network with 250 nodes. We see that the achieved profit increases rapidly in the beginning with the budget but slows down as the issue then becomes not the limited budget but rather the competition between missions.

6.3 Dynamic Setting

In the dynamic problem, missions arrive without warning over time and the sensors used to satisfy them have limited battery lives. The goal is to maximize the total profit achieved by missions over the entire duration of the network. In this section, we test the dynamic heuristic algorithms on randomly generated *sensor network histories* in order to gauge the algorithms' real-world performance.

In addition to the assumptions made in Section 6.1, here we also assume the following. Missions arrive according to a Poisson distribution with an average arrival rate of 4 missions/hour or 8 missions/hour depending on the experiment. Each sensor starts with a battery that will last for 2 hours of continuous sensing (i.e., $B = 7200$ in seconds). We assume that this battery is used solely for sensing purposes and is different than the the sensor's main battery which it uses for communication and maintaining its own operation. Mission lifetimes are exponentially distributed with an average of 1 hour. We limit the minimum lifetime to 5 minutes and the maximum to 4 hours. The number of nodes used in the following experiments is set to 500 nodes.

We test the performance of the dynamic schemes described in Section 5. We compare the energy-aware (*E-aware*) and the energy and lifetime-aware (*E/L-aware*) schemes with a basic scheme that does not take energy or network lifetime into account when making the decision on to which mission to propose (i.e. sensors propose to any mission in their range). For comparison purposes, we also show the performance of the network if we assume that sensors have infinite energy (i.e. $B \geq$ simulation time).

Of course, finding the true optimal performance value is NP-hard, so we cannot do an exact comparison. Even the LP relaxation of program **P'** above may be infeasible because of the number of decision variables. To provide some basis of comparison, therefore, we define a further relaxation which is feasible to solve and provides an upper-bound of the optimal. This formulation condenses the entire history into a single timeslot. The profits and demands are multiplied by the duration of the mission. Since time is elided, the sensor constraint now asserts only that a sensor be used (fractionally) for at most B timeslots, over the entire history, where B is the battery lifetime. Note that the solution value provided is the total profits over the entire history, not a time-series. We indicate its value in the results by a straight line drawn at the average profit corresponding to this total.

In both the E-aware and the E/L-aware schemes, we need to compute the expected profit of a mission (Expression 1 above), to determine whether the sensor should propose to the mission. Because we cap the distributions – limit the minimum demand and profit for missions and cap them by the available resources in the first case and by 100 in the second – the actual averages are not equal to the a priori averages of the distributions. We found an empirical average demand of d' = 1.2 and an average profit of $p' =$ 10.9. The empirical average duration, which is used to evaluate expression (3) above, was found to be 3827.8 seconds (roughly an hour).

Figure 5 shows the achieved profit (as a fraction of maximum possible) per-timeslot. We assume the network is considered to be of use as long as this value stays above 50% (shown with a fine horizontal line). The target network lifetime is 3 days (shown as a fine vertical line) and the simulations are run for one week of time. Knowledge of the target network lifetime is used by the E/L-aware and in the LP. The other schemes assume that network will potentially have infinite duration.

From Figure 5 we see that the profits of all schemes stay above the 50% threshold for the target lifetime. The basic scheme achieves most of its profits in the beginning and then profits go down (almost linearly) as time progresses. The E-aware scheme tries to conserve its resources for high profit missions. Because it ignores the fact that we care more about the first 3 days than anytime after that it, becomes overly conservative

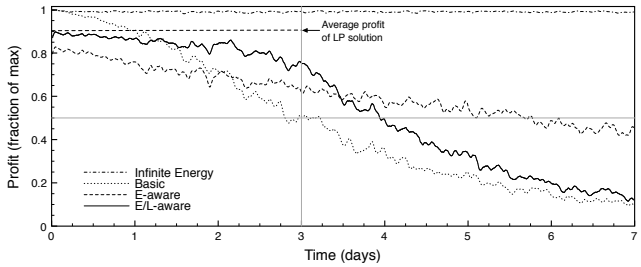

Fig. 5. Fraction of achieved profits. (arrival rate = 4)

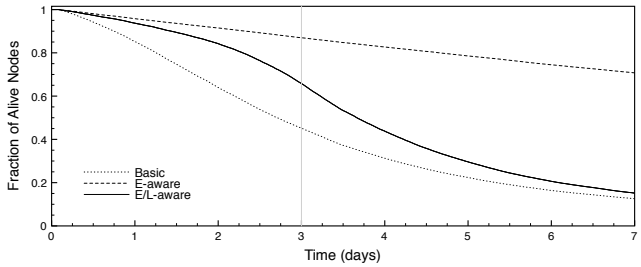

Fig. 6. Fraction of alive nodes (arrival rate = 4)

and ignores many missions. Such a scheme is better suited to the case when there is no known target lifetime for the network and we want the network to last as long as possible. We see that the profit for E-aware does not fall below the 50% threshold until the end of the sixth day.

In the E/L-aware scheme, nodes will initially be aggressive in accepting missions that might not provide their expected value but become more cautious as their energy is used. However, unlike E-aware, as their remaining expected occupancy time approaches their remaining lifetime, sensors will again accept missions with lower and lower profits. The curves for E-aware and E/L-aware cross by the middle of the fourth day, at which point the E-aware becomes better. When compared to the average LP solution, we see that E/L-aware is very close with a difference (on the average) of few percentage points. We also found that in terms of the sum of all profits during the target network lifetime (i.e. the area under the curve for the first 3 days), the E/L-aware achieves about 84% of the profits compared to 72% for the E-aware. This means that E/L-aware achieves close to 17% higher profits. If the sensors battery lifetime is increased from 2 to 3 hours the percentage increase becomes about 22%.

The fraction of remaining alive sensors over time is shown in Figure 6. Because in the basic scheme sensors propose to any mission within its range no matter how low the profit is, nodes start dying rapidly. By the the end of the third day, only half the nodes can be used for sensing and by the end of the seventh day this falls below 15%. In E-aware, nodes become very cautious as their batteries run low, which helps the network to last for longer without significant sacrifice of achieved profits per timeslot. By the end of the 7 days, about 72% of the nodes remain living. For E/L-aware, sensors accept more missions, and hence are used at a higher rate, as the target lifetime of the network approaches. In the figure, we can see this happening by the second day, when the curve

of E/L-aware diverges from that of E-aware. By the end of the seventh day, it has used nearly as much energy as the basic scheme.

One thing to note is that we assume E/L-aware acts like the basic scheme once the target lifetime of the network has passed, i.e. sensors propose to all nearby missions. If this behavior were changed to emulate the E-aware, we expect the energy usage to slow down which would conserve a larger fraction of alive nodes. A similar effect is expected to happen to profit. As the fraction of alive nodes will be higher, the decrease in profit after target network lifetime will slow down.

We omit figures showing the fraction of achieved profit and fraction of alive nodes over time, with twice the previous arrival rate (8 missions/hour). Due to the increased number of missions, sensors are used more rapidly and hence both the profit and fraction of alive nodes decrease quickly. The basic scheme passes the 50% profit line by the middle of the second day and both E-aware and E/L-aware pass that point in the beginning of the fourth day. But by that point, E/L-aware achieves significantly higher profits than E-aware. Similar effects are seen on the fraction of alive nodes.

7 Conclusion and Research Directions

In this paper, we defined new sensor-assignment problems motivated by frugality and conservation of resources, in both static and dynamic settings. We proposed schemes to match sensing resources to missions in both settings and evaluated our these schemes through simulations. In the static case, we found that the multi-round GAP scheme, which can be implemented in a distributed fashion, outperformed a centralized greedy solution. In the dynamic setting, we found that overall performance can be significantly improved if sensors are allowed to sometimes refuse to offer utility to "weak" missions. Performance improves when sensors make this decision based both on their remaining energy level and the remaining target network lifetime.

More sophisticated utility models, taking into consideration not just physical attributes such as distance but also some sort of semantics, could be developed. Above, we based the potential utility of a sensor-missing pairing on their separating distance. However, in the case of video sensors, for example, the closest sensor to an event might not be the best candidate for selection because its view of the field is obstructed or it cannot provide sufficient frame rate. Also, regarding the value of utility from multiple sensors, we make the simplifying assumption of additive utilities, which is realistic in some but clearly not all scenarios. In 3-d reconstruction, for example, the benefit of having two viewing angles separated by 45 degrees may be much greater than two views from essentially the same location. In ongoing work, we are investigating models that allow for such non-additive joint utility.

Acknowledgements. 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.

References

1. A. Bar-Noy, T. Brown, M. Johnson, T. La Porta, O. Liu, and H. Rowaihy. Assigning sensors to missions with demands. In *ALGOSENSORS 2007*.
2. J. Byers and G. Nasser. Utility-based decision-making in wireless sensor networks. In *Proceedings of MOBIHOC 2000*.
3. R. Cohen, L. Katzir, and D. Raz. An efficient approximation for the generalized assignment problem. *Inf. Process. Lett.*, 100(4):162–166, 2006.
4. L. Kaplan. Global node selection for localization in a distributed sensor network. *IEEE Transactions on Aerospace and Electronic Systems*, 42(1):113–135, January 2006.
5. B. Karp and H. Kung. Greedy perimeter stateless routing for wireless networks. In *Proceedings of MobiCom '00*, pages 243–254, Boston, MA, August 2000.
6. J. Lu, L. Bao, and T. Suda. Coverage-aware sensor engagement in dense sensor networks. In *Proceedings of the International Confernece on Embedded and Ubiquitous Computing - EUC 2005*, Dec. 2005.
7. T. Mullen, V. Avasarala, and D. L. Hall. Customer-driven sensor management. *IEEE Intelligent Systems*, 21(2):41–49, Mar/April 2006.
8. M. Perillo and W. Heinzelman. Optimal sensor management under energy and reliability constraints. In *WCNC '03*.
9. H. Rowaihy, S. Eswaran, M. Johnson, D. Verma, A. Bar-Noy, T. Brown, and T. La Porta. A survey of sensor selection schemes in wireless sensor networks. In *SPIE Defense and Security Symposium*, 2007.
10. H. Rowaihy, M. Johnson, T. Brown, A. Bar-Noy, and T. La Porta. Assigning Sensors to Competing Missions. Technical Report NAS-TR-0080-2007, Network and Security Research Center, Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA, USA, October 2007.
11. K. Shih, Y. Chen, C. Chiang, and B. Liu. A distributed active sensor selection scheme for wireless sensor networks. In *Proceedings of the IEEE Symposium on Computers and Communications*, June 2006.
12. V. V. Vazirani. *Approximation Algorithms*. Springer, 2001.
13. F. Zhao, J. Shin, and J. Reich. Information-driven dynamic sensor collaboration. *IEEE Signal Processing Magazine*, 19(2):61–72, March 2002.

Appendix: 1-d Static Setting

In 1-d settings such as border surveillance, sensors and missions lie on the line, and edge weights e_{ij} depend in some way on the distance between S_i and M_j. If $e_{ij} = 1/(1 + D_{ij})$, for example, then the 1-d problem is strongly NP-hard, even with no budget constraint, since the known 3-Partition reduction [10] also works in 1-d. To adapt the reduction to 1-d, place all missions at one point on the line, and choose the sensor locations so that the resulting edge weights equal the corresponding 3-Partition element values. If we adopt a binary sensor model, however, then the budget-constrained 1-d SMD problem is in *P*. We give a polynomial-time algorithm for a 1-dimensional, budget-constrained version of SMD in which edge-weights are 0/1 based on sensing range and the budget limits the *total number of sensors* that may be used. In this setting, $e_{ij} = 1$ if $D_{ij} \leq r$ and 0 otherwise. We may therefore assume without loss of generality that the demands d_j are integral, but profits p_j may be arbitrary positive values. Assignments and profits are both non-fractional. For now, assume budgets are infinite.

We first observe that if r is large enough so that $e_{ij} = 1$ for all i, j then this is an instance of the 0/1 knapsack problem, solvable in polynomial time (in terms of *our problem*'s input size, not

knapsack's). In this case, sensors are simply the units in which knapsack weights are expressed, with the knapsack budget equal to n, so the knapsack DP runs in time $O(mn)$. In general, though, we will have $e_{ij} = 0$ for some i, j. We can solve the problem by a modification of the standard 0/1 knapsack DP [12]. We construct an $m \times n$ table of optimal solution values, where $v(j, i)$ is the optimal value of a sub-instance comprising the first j missions and the first i sensors. Row 1 and Column 1 are initialized to all zeros (optimal profits with the zero-length prefix of missions, and the zero-length prefix of sensors, respectively). We fill in the entries row-by-row, using a value d'_j defined below:

$$v(j,i) = \begin{cases} v(j-1,i), \text{if } M_j \text{ is not satisfiable with } S_1, ..., S_i \\ \max(v(j-1,i),\ p_j + v(j-1, i-d'_j)), \text{o.w.} \end{cases}$$

The two entries of the *max* correspond to satisfying M_j or not. The first option is clear. If we do satisfy M_j in a solution with only sensors $1, ..., i$ available, the important observation is that we may as well use for M_j the rightmost d_j sensors possible, since in this sub-instance M_j is the rightmost mission. (The DP will consider satisfying M_j when solving sub-instances with fewer sensors.) Note that we might be unable to assign the rightmost d_j sensors, since some of these may be too far to the right for M_j to be within their sensing range. Let d'_j equal d_j plus the number of sensors among $S_1, ..., S_i$ too far to the right to apply to M_j. Any sensor too far to the right to reach M_j is also too far to the right to reach M_k for $k < j$. What remains among this sub-instance will therefore be missions $1, ..., j-1$ and sensors $1, ..., i - d'_j$. This completes the recursion definition.

Proposition 2. *With unbounded budgets, the 1-d problem can be solved optimally in time $O(mn)$.*

Proof. Optimality is by induction. The running time results from the table size.

$v(0, j, i) \leftarrow 0$ **for** $0 \le j \le m, 0 \le i \le n$
for $b = 1$ to B **do**
  $v(b, 0, i) \leftarrow 0$ **for** $0 \le i \le n$
  for $j = 1$ to m **do**
    for $i = 0$ to n **do**
      if M_j is not satisfiable with b sensors among $1..i$ **then**
        $v(b, j, i) \leftarrow v(b, j-1, i)$
      else
        $v(b, j, i) \leftarrow \max(v(b, j-1, i),\ p_j + v(b - d_j, j-1, i - d'_j))$

Table 5. Algorithm 3: 1-d 0/1 dynamic program

In the formulation of the infinite-r setting above, the number of sensors n becomes the knapsack budget, whereas in the general setting the sensors must be assigned to missions in range. We now introduce an explicit budget into the problem. Let the model be the same as above, except now also given is an integral budget $B \le n$, which is the total number of sensors that may be used. First delete any missions with $d_j > B$. Then the budgeted version is solved by creating a 3-d DP table, whose third coordinate is budget. We initialize this table by filling in all nm entries for budget 0 with value 0. Then for layer b, we again fill in the entries row-by-row.

Notice that the remaining budget to be spent on the "prefix" instance is $b - d_j$, not $b - d'_j$, since only d_j sensors are actually assigned to mission M_j. Thus we conclude:

Proposition 3. *Algorithm 3 is optimal and runs in time $O(mn^2)$.*