Near Optimal Multi-Application Allocation in Shared Sensor Networks

You Xu; Abusayeed Saifullah;
Yixin Chen; Chenyang Lu
Washington University in St. Louis
St. Louis, MO 63130, USA
{yx2,saifullaha,chen,lu}@cse.wustl.edu

Sangeeta Bhattacharya
Intel Labs, India
Bangalore South Taluk
Bangalore 560103, India
sangeeta.bhattacharya@intel.com

ABSTRACT

Recent years have witnessed the emergence of shared sensor networks as integrated infrastructure for multiple applications. It is important to allocate multiple applications in a shared sensor network, in order to maximize the overall Quality of Monitoring (QoM) subject to resource constraints (e.g., in terms of memory and network bandwidth). The resulting constrained optimization problem is a difficult and open problem since it is discrete, nonlinear, and not in closed-form. This paper makes several important contributions towards optimal multi-application allocation in shared sensor networks. (1) We formulate the optimal application allocation problem for a common class of distributed sensing applications whose QoM can be modeled as variance reduction functions. (2) We prove key theoretical properties of the optimization problem, including the monotonicity and submodularity of the variance reduction functions and the multiple knapsack structure of constraints; (3) By exploiting these properties, we propose a local search algorithm, which is efficient and has a good approximation bound, for application allocation in shared sensor networks. Simulations based on both real-world datasets and randomly generated networks demonstrate that our algorithm is competitive against simulated annealing in term of QoM, with up to three orders of magnitude reduction in execution times, making it a practical solution towards multi-application allocation in shared sensor networks.

Categories and Subject Descriptors

G.1.6 [**Numerical Analysis**]: Optimization—*Constrained optimization, Nonlinear programming*; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*; G.2.1 [**Algorithms**]: Combinatorics—*Combinatorial algorithms*

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
MobiHoc'10, September 20–24, 2010, Chicago, Illinois, USA.
Copyright 2010 ACM 978-1-4503-0183-1/10/09 ...$10.00.

General Terms

Algorithms, Performance, Theory

Keywords

Submodular Optimization, Shared Sensor Network, Resource Allocation

1. INTRODUCTION

While wireless sensor networks (WSNs) have traditionally been used as specialized platforms for single applications, recent years have witnessed the emergence of integrated WSNs as shared infrastructure for multiple applications. Shared sensor networks have been deployed in many application domains such as urban sensing [7], building automation and environmental monitoring [1]. For example, a smart building may employ an integrated WSN to support multiple applications including temperature and humidity monitoring, security alarms, light control, energy metering, and structural health monitoring. Compared to separate application-specific sensor networks, a shared sensor network can be more cost effective and more flexible as it enables resource sharing among applications and dynamic resource allocation in response to changes in the environment and user needs.

A fundamental challenge faced by shared WSNs is to deal with severe resource constraints of sensor nodes. For example, the TelosB mote [2], a representative sensor platform, only has 10 KB of RAM, a 250 Kbps radio, and a 16-bit CPU running at 8 MHz. It is, therefore, important to optimize the allocation of multiple applications among sensor nodes in order to maximize the overall Quality of Monitoring (QoM) subject to multiple resource constraints. Multi-application allocation in a shared WSN is a challenging and open optimization problem. The objective function for this type of problem is defined over a finite discrete set representing the assignments of applications to different subsets of nodes. The resulting constrained optimization problem is difficult since it is usually discrete, nonlinear, and does not have closed-form objective functions.

In this paper, we tackle the optimal application allocation problem for a common class of networked sensing applications, whose QoM can be formulated using *variance reduction* [13] functions that map a set of nodes to a QoM value in presence of inter-node QoM correlation. Variance reduction is an important QoM attribute for distributed sensing applications for estimating the spatial distributions of environmental variables by sampling at a discrete set of locations. For example, intelligent air conditioning requires fine grained estimation of the temperature distribution based on the measurement of a subset of nodes in a shared sensor network. The temperature measurements of different sensors are correlated with each other, and the degree of their correlations depends on the sensor locations and the spatial distribution of temperature. To exploit the correlation of sensor readings, probabilistic models have been developed that enable the prediction of sensor readings at all nodes based on the sensor readings of a subset of nodes. Different subsets of nodes provide different reductions in the variance of the estimated sensor readings. The higher the variance reduction, the higher the confidence in the predictions. Such probabilistic methods are fairly general and have been validated in a broad range of applications that monitor temperature, humidity, and pollution in waste water [13, 17, 18].

Our approach exploits the *submodular* property of the variance reduction functions in networked sensing applications. Intuitively, a function f that maps a subset of a set S to a real value is submodular if it has a *diminishing return* property, i.e., adding an element to a smaller subset of S makes a bigger difference than adding it to a larger subset of S.

The submodularity of node allocation is due to the inherent property that sensor readings from different nodes are often correlated. For instance, since the temperature readings from different nodes in the same room are correlated with each other, allocating a new node to a temperature monitoring application results in diminishing improvement to the QoM as the set of nodes allocated to the application grows. Submodularity of sensor allocation for monitoring temperature [5] and water quality [13, 17, 18] has been observed in previous studies of real-world datasets. While the submodularity property has previously been used for sensor selection for target localization [15] and for sensor placement. We propose to exploit submodularity for allocating subsets of nodes to multiple contending applications in a shared sensor network.

Optimal multi-application allocation in shared WSNs represents an open problem in optimization theory. To address this difficult optimization problem, we provide several major theoretical results: 1) We show that the optimal multi-application allocation problem is a submodular optimization problem with multiple knapsack constraints; 2) We prove that a common QoM function based on variance reduction is monotonic; 3) We propose a fractional relaxation based greedy local search algorithm for solving QoM optimization problem and prove that our algorithm can achieve a 1/3-approximation bound. The proposed algorithm and approximation bound are rather general and can be applied for any submodular and monotonic QoM functions. Empirically, we verify the submodularity and monotonicity of our variance reduction using Intel Lab's real-world data on humidity and temperature monitoring [3] . We also present simulation results and analyze the scalability and execution time of our algorithm. We compare our algorithm against two standard allocation algorithms, bin packing and simulated annealing. The results demonstrate that our algorithm achieves a much higher system QoM and is much more efficient than other algorithms. As a result, the algorithm is an attractive and practical solution for application allocation in shared sensor networks.

2. RELATED WORK

2.1 Resource Allocation

Most existing works on resource allocation for sensor networks focus on a single application and a single resource constraint. Several sensor selection schemes presented in the literature suggest mapping sensor node sets to utility values [5, 6]. However, instead of considering multiple applications and multiple resource constraints as is done in this paper, they only consider a single application scenario and a single resource constraint. For example, the utility-based sensor selection scheme presented in [5] maximizes the utility of a single application under an energy constraint. Their objective function is built from the network flow perspective assuming the submodularity of network flow. Similarly, the scheme presented in [6] also maximizes a single application's utility under an energy constraint. In the utility-based resource allocation system presented in [22], nodes act as self-interested agents that select actions to maximize the utility of a single application under energy constraints. The sensor placement algorithm proposed by Krause *et al.* [13] can be treated as a submodular optimization problem with a single application in the network. None of these work deals with the allocation of multiple applications in shared sensor networks.

With the emergence of shared sensor networks, programming abstractions and systems have been developed in recent years to support multiple applications in such systems. Several projects developed group-based abstractions to support multiple concurrent applications [11, 25, 14]. A market-oriented approach was proposed for bandwidth allocation in shared sensor networks [21]. However, none of these considers QoM while allocating applications.

Utility-based Multi-Application Deployment Environment (UMADE) [4] is an integrated system for allocating and deploying applications in a shared sensor networks. While UMADE aims to allocate applications based on QoM, the work focuses on the design and implementation of the system architecture and employs a simple allocation algorithm without an approximation bound. Thus, the contribution of UMADE is complementary to this paper, which focuses on algorithm design and theoretical analysis of application allocation in shared sensor networks.

2.2 Submodular Optimization

Optimal multi-application allocation under resource constraints is complicated since it violates some key assumptions in the classical optimization theory. We discuss classic and recent results and their limitations below.

DEFINITION 1. *Given a finite set S of n elements and a function $f : 2^S \mapsto R$, f is submodular if and only if $f(A \cap B) + f(A \cup B) \leq f(A) + f(B)$ for any $A, B \subseteq S$. f is monotonic if and only if $f(A) \leq f(B)$ for any $A \subseteq B \subseteq S$.*

DEFINITION 2. *Given a finite set S of n elements, a submodular optimization problem (SOP) is*

$$\text{maximize}_{\,V \subseteq S} \quad f(V), \tag{1}$$

$$\text{subject to:} \quad h_i(V) = 0, \quad i = 1, \cdots, M \tag{2}$$

where $f : 2^S \mapsto R$, f is submodular function on S and $h_i : 2^S \mapsto \{0, 1\}, i = 1, \cdots, M$ are constraint functions.

In the next section, we will formulate our QoM optimization problem to an SOP.

Definition 3. ***(Knapsack Constraint)*** *Given a finite set S, a constraint h in a SOP on S is a knapsack constraint if it satisfies that, $h(V) = 0$ if and only if $\sum_{s_i \in S} w_i * x_i \leq R$, where x_i is a binary variable representing whether s_i is in V ($x_i = 1$) or not ($x_i = 0$), w_i is a positive weight assigned to each s_i, and R is a positive constant.*

In our multi-application allocation problem, since there are multiple resources and more than one sensor node, and naturally we will have one knapsack constraint for each resource on each sensor, the constraints in our problem are multiple knapsack constraints.

Nemhauser and Wolsey's seminal work [12] showed that a greedy algorithm can achieve a $1 - 1/e$ approximation rate in maximizing a non-decreasing submodular function on a uniform matroid. Krause *et al.* first introduced this greedy algorithm to sensor placement problems under uniform matroid constraints [16]. Feige *et al.* [9] further proved that there is no $(1 - 1/e + \epsilon)$-approximation polynomial algorithm for any constant $\epsilon > 0$, unless $P = NP$. Thus, $(1 - 1/e)$ is considered to be the best approximate rate we can get for polynomial algorithms. Fisher *et al.* [10] proposed a $1/2$ approximation algorithm for SOPs with single matroid constraints. Some recent work improved this bound to $(1 - 1/e)$ assuming that f is the sum of some matroid ranks [20]. For SOPs with one knapsack constraint, Sviridenko [24] proposed a $(1 - 1/e)$-approximation algorithm.

For SOPs with $k > 1$ knapsack constraints, Lee *et al.* [20] proposed an $1/5$ approximation algorithm. Conforti [8] proposed an algorithm with $\frac{1}{1+k+1/k}$-approximation rate when these k constraints are in an independent system, which is a general case for knapsack constraints. Kulik *et al.* [19] gives an $(1 - 1/e)$ bound for monotonic submodular objective function with multiple knapsack constraints. This theoretical approximation bound is better than our algorithm proposed in this paper. However, as we will discuss in detail in Section 5, it requires an enumeration of all possible subsets of a large set, which requires an exponential number of computations.

3. PROBLEM FORMULATION

A shared sensor network consists of resource constrained sensor nodes and a base station with more resources. Our optimization problem is run on the base station, which then decides the allocation of applications among the nodes. Many distributed sensing applications are designed to estimate *spatially correlated* phenomena. While our formulation of variance reduction for an application is based on [13], the novelty of our problem formulation is to incorporate variance reduction in the multi-application allocating problem in a shared sensor network.

Different subsets of nodes provide different reductions in the variance of the estimated sensor readings. The higher the variance reduction, the higher the confidence in the predictions. Therefore, we formulate our objective function in the form of variance reduction for this common class of applications.

Suppose V is the set of sensor nodes we will deploy application t on. For any two sensor nodes i and $j \in V$, $i \neq j$, the covariance σ_{ij} can be calculated based on measured data. We define a **kernel matrix** $\mathcal{K}$ where the elements in the ith row and jth column is σ_{ij}. $\mathcal{K}$ is essentially the covariance matrix for sensor nodes. For two subsets of sensor nodes A and $B \subseteq V$, we denote their covariance matrix by $\mathcal{K}_{AB}$, with rows corresponding to A and columns corresponding to B extracted from $\mathcal{K}$.

Given a finite subset $A \subset V$ with applications allocated, if the sensor reading for any application on any given node complies with a Gaussian process, then, for any sensor node $y \in V$, we can predict its reading using maximum likelihood estimation as $\arg\max_{X_y} P(X_y|X_A)$. Note that the distribution of X_y has variance as $\sigma^2_{y|A} = \mathcal{K}_{yy} - \mathcal{K}_{yA}\mathcal{K}^{-1}_{AA}\mathcal{K}_{Ay}$. Thus, for a given set A with application allocated, the variance of the unknown region $\bar{A} = V \backslash A$ is

$$\sigma^2_{\bar{A}|A} = tr(\mathcal{K}_{\bar{A}\bar{A}}) - tr(\mathcal{K}_{\bar{A}A}\mathcal{K}^{-1}_{AA}\mathcal{K}_{A\bar{A}}),$$

where $tr()$ is the trace function of a matrix.

If we consider the QoM as the confidence of the measurement, naturally, we want to minimize the variance of $\bar{A}$ given A such that the quality of sensing is maximized. Namely, we want to maximize the negation of the variance. Since we have $tr(\mathcal{K}) = tr(\mathcal{K}_{AA}) + tr(\mathcal{K}_{\bar{A}\bar{A}})$, the objective function for one application is:

$$\text{maximize} \quad Q = tr(\mathcal{K}_{AA}) + tr(\mathcal{K}_{\bar{A}A}\mathcal{K}^{-1}_{AA}\mathcal{K}_{A\bar{A}}) \qquad (3)$$

Now we turn to multiple applications. Suppose we have p applications to allocate, the objective function is to maximize $\sum_{t=1}^{p} w_t Q_t$, where each Q_t is the QoM value for application t and $w_t > 0$ is the weight of each application assigned by user.

Applications may be deployed dynamically at different points of time based on user demand. To handle dynamic application arrivals, a shared sensor network should support flexible trade-off between QoM and the resource consumption of an application. Hence, the allocation algorithm needs to be reasonably efficient (in terms of time), because the allocation needs to be recomputed dynamically in response to application arrivals/departures and changes in application weight, as well as node additions and removals.

We consider two critical resource constraints, namely, bandwidth and memory constraints (CPU utilization constraint may be handled similarly as the memory constraint). First, for each sensor, the total memory consumed by all applications assigned to the sensor cannot exceed its limits. Second, for each sensor node, the total bandwidth consumed by receiving and sending data and the external interferences cannot exceed the channel capacity.

To incorporate the bandwidth constraints, we construct an Interference-Communication (IC) graph of the network. The IC graph consists of all the nodes as vertices and has two types of directed edges: *communication edge* and *interference edge*. A communication edge $\overrightarrow{uv}$ indicates that the packets transmitted by node u can be received by node v. An interference edge $\overrightarrow{uv}$ means that the transmission of node u interferes with any transmission intended for v even though the transmission of u may not be correctly received by v. A subset of the communication edges forms the routing tree rooted at the sink. Existing protocols such as the RID protocol [26] can be used to construct the IC graph. While our protocol-based interference model is not as general as an SNR based physical model, an advantage of our model is that makes our algorithm more practical as it can leverage the existing RID protocol for interference detection.

We denote the set of all nodes that have interference with u by I_u and all nodes that u receives data from by R_u. Also, we assume that the amount of data sent out by u is d_u.

Therefore, suppose the channel capacity is a uniform number C for all nodes and each node has memory limit M_i, the constrained optimization formulation is

$$
\begin{aligned}
(P_1): \qquad & \text{maximize} \quad && \sum_{t=1}^{p} w_i Q_t \\
& \text{subject to} && \sum_{t=1}^{p} a_{t,j} m_{t,j} \le M_j, \\
& && \sum_{l \in I_j} d_l + \sum_{t=1}^{p} a_{t,j} b_{t,j} + \sum_{l \in R_j} d_l \le C \\
& && \forall j = 1, \cdots, n
\end{aligned}
$$

where $a_{t,j}$ is a binary variable denoting if application t is assigned to node j. $m_{t,j}$ and $b_{t,j}$ are the usage of memory and bandwidth respectively when application t is assigned to a sensor node j, and n is the number of sensors. It is also easy to see that we have in total $k = 2n$ constraints.

Note that in the above formulation, d_j can be recursively defined as $d_j = \sum_{l \in R_j} d_l + \sum_{t=1}^{p} a_{t,j} b_{t,j}$ for a non-root node j and d_j is 0 for root node. It is easy to see that all all memory and bandwidth constraints here are knapsack constraints. This kind of constraint formulation is also quite general and can be used to characterize various communication patterns among nodes, such as the pattern in a data collection application that collects data from every node on the routing tree.

4. THEORETICAL PROPERTIES

Now we study the characteristics of P_1. These characteristics facilitate the design and analysis of our search algorithm.

4.1 Monotonicity

We will prove in this subsection that our objective function of P_1 is monotonic. The monotonicity is a strong property that enables us to improve the approximation bound.

Assuming that the application set $\mathcal{P}$ contains p applications and the node set $\mathcal{V}$ contains n nodes. Any application-to-node assignment (or assignment for short) is a subset of $\mathcal{M} = \mathcal{P} \times \mathcal{V}$. We have the following theorem:

THEOREM 1. *For p applications and n sensor nodes, if A and B are two application-to-node assignments where $A \subseteq B \subseteq \mathcal{M}$, we have $Q(A) \le Q(B)$ if all kernel matrices for all p applications are semi-positive definite (s.p.d.).*

To prove Theorem 1, since Q is a sum of Q_t for $t = 1, \cdots p$, we only need to prove that Q_t is monotonic if its kernel matrix $\mathcal{K}_t$ is s.p.d. We prove this result by providing the following lemma.

LEMMA 1. *Given a s.p.d. matrix $\mathcal{K}$ in a block separated form as*

$$
\mathcal{K} = \begin{pmatrix} A & D & E \\ D^T & B & F \\ E^T & F^T & C \end{pmatrix},
$$

we have:

$$
\begin{aligned}
tr(A) \quad &+ \quad tr((DE)(DE)^T A^{-1}) && (4) \\
&\le \quad tr((A+B) + \begin{pmatrix} E \\ F \end{pmatrix} \begin{pmatrix} E \\ F \end{pmatrix}^T \begin{pmatrix} P & Q \\ Q^T & R \end{pmatrix}) && (5)
\end{aligned}
$$

where

$$
\begin{pmatrix} P & Q \\ Q^T & R \end{pmatrix} = \begin{pmatrix} A & D \\ D^T & B \end{pmatrix}^{-1}. \tag{6}
$$

We first expand (4) to $tr(A) + tr(D^T A^{-1} D + E^T A^{-1} E) \le tr(A+B) + tr(E^T P E + F^T Q^T E + E^T Q F + F^T R F)$.

The proof consists of two parts.

Part 1: First we prove that $tr(A + D^T A^{-1} D) \le tr(A+B)$.

Since $\mathcal{K}$ is an *s.p.d.* matrix, for any real vectors x and y, we have

$$
(x,y)^T \begin{pmatrix} A & D \\ D^T & B \end{pmatrix} (x,y) > 0. \tag{7}
$$

Since an *s.p.d.* matrix always has a Cholesky factorization, we rewrite A as $U^T U$ and plug it in to (7). Since U is invertible and $U^{-T} = (U^T)^{-1}$, the left side becomes $x^T U^T U x + x^T U^T U^{-T} D y + y^T D^T U^{-1} U x + y^T B y + y^T D^T U^{-1} U^{-T} D y - y^T D^T A^{-1} D y$. Thus, (7) is actually $(Ux + U^T D y)^T (Ux + U^{-T} D y) + y^T(-D^T A^{-1} D + B) y > 0$.

Since the linear system $Ax + Dy = 0$ with respect to x always has a solution for any y since A is *s.p.d.*, the vector $(Ux + U^T D y)$ can always be 0 for any given y. This implies that $y^T(-D^T A^{-1} D + B) y > 0$ for any y. Thus, we know that $(-D^T A^{-1} D + B)$ is *s.p.d.*, which implies that $tr(-D^T A^{-1} D + B) > 0$ and $tr(D^T A^{-1} D) < tr(B)$.

Part 2: Now we prove $tr(E^T A^{-1} E) \le tr(E^T P E + F^T Q^T E + E^T Q F + F^T R F)$, which can be written as $tr(\Delta) \ge 0$ where $\Delta = E^T(P - A^{-1})E + F^T Q^T E + E^T Q F + F^T R F)$. Since we have $AP + DQ^T = I$ and $AQ + DR = 0$, namely, $P + A^{-1} D Q^T = A^{-1}$ and $Q R^{-1} Q^T + A^{-1} D Q^T = 0$, we get $E^T(P - A^{-1})E = E^T(Q R^{-1} Q^T)E$. We have $\Delta = E^T(Q R^{-1} Q^T)E +$
$F^T Q^T E + E^T Q F + F^T R F$, which leads to

$$
\Delta = \begin{pmatrix} E^T & F^T \end{pmatrix} \begin{pmatrix} Q & 0 \\ R & I \end{pmatrix} \begin{pmatrix} R^{-1} & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} Q & R^T \\ 0 & I \end{pmatrix} \begin{pmatrix} E & F \end{pmatrix}^T
$$

Since R^{-1} is *s.p.d.*, $tr(\Delta)$ is non-negative. □

We have also verified the monotonicity and submodularity of the variance reduction function on Intel Research Berkeley Lab's dataset [3]. Among a random selection of $30,000,000$ pairs of assignments, more than 98% have satisfied monotonicity and submodularity constraints, the rest 2% are due to measuring noises.

5. OPTIMIZATION ALGORITHM

In this section, we present our constrained optimization algorithm for solving P_1 and prove its approximation bound. The novelty of our work is to exploit the monotonicity of the objective function to obtain a tighter approximation bound by adopting the fractional relaxation framework proposed in [20].

5.1 Basic definitions

The objective function that we are going to optimize is $f = Q$. All application to node assignments are subsets of the set $V = \mathcal{M} = \mathcal{P} \times \mathcal{V}$, where $\mathcal{P}$ and $\mathcal{V}$ are the sets of applications and sensors, respectively. Since f maps a subset of V to a value, we name V as the **ground set** of f and $|V| = m$. We also use notation $[m]$ to denote the set $\{1, \cdots, m\}$. We use o_k to denote k normalized weight vectors corresponding to k resource constraints, all having capacities normalized to 1. We define $f^* = max\{f(v) : v \in V\}$ and assume each singleton set in V is feasible for the constraints (otherwise, those violating elements can be removed from the problem).

We assume that the global optimal value for f subject to resource constraints is **Opt**. We further assume that any feasible assignment, which is a subset of V, can be partitioned into two sets : a heavy set and a light set, where elements in the heavy set has a weight factor in the knapsack constraints larger than a given threshold δ. The heavy set and the light set of the optimal solution V^* are denoted by H and L, respectively.

Our **fractional relaxation greedy (FRG)** algorithm consists of the following steps.

Step 1 Identify heavy elements and light elements based on a given threshold δ. Enumerate all possible assignments of heavy elements and find the optimal subset of heavy elements.

Step 2 Solve a relaxed problem on light elements in which the function f is extended to a continuous function F.

Step 3 Round back the solution to the relaxed problem from Step 2 to a solution to the original problem. Combine the heavy elements and light elements to get the final solution.

In Step 1, to decide heavy and light elements, we choose a fixed constant $\eta > 0$. Let $c = \frac{16}{\eta}$, we let $\delta = \frac{1}{4c^3k^4}$ to be the threshold for heavy elements. As each knapsack constraint has been normalized with capacity value 1, its corresponding heavy element set H_i has at most $1/\delta$ elements since $\delta \cdot |H_i| \le \sum_{e \in H} o_i(e) \le 1$. Therefore, if there are k knapsack constraints in total, we have $|H| \le k/\delta$, which is a constant. By enumeration, we can get $f(H)$ under a time complexity of $n^{O(k/\delta)}$. So, we only focus on the light elements in Steps 2 and 3.

5.2 Step 2: Solving a relaxed problem

In this step, we extend f defined on the subsets of V to F which is defined on fractional values.

Let F be a function that maps from $[0,1]^m \to \mathbb{R}^+$. For $y \in [0,1]^m$, let $\hat{y}$ denote a random $0-1$ vector where each element is independently rounded to 1 with probability y_j or 0 otherwise. Then, F is defined as:

$$F(y) = E[f(\hat{y})] = \sum_{S \subseteq V} f(S) \cdot \prod_{i \in S} y_i \cdot \prod_{j \notin S} (1 - y_j). \qquad (8)$$

F is a monotonic and concave function which extends f to the continuous domain $[0,1]^m$. However, extending f to F changes the problem domain from a finite set to a continuous domain, which is not suitable for local search. Hence, we define a new set $\mathcal{G} \in [0,1]^m$ called the discretized set of F where for each dimension i, $\mathcal{G}_i$ take values in the form of l/s_i where s_i is an integer and $l \in [s_i]$. We also denote the continuous domain of F that satisfies the resource constraints as $\mathcal{U}$.

```
Result: A local optimum $y$ of the relaxed optimization
        function $F$
$y = \arg\max F(\{x\})$, $x$ is a singleton set;
$flag = True$ ;
while $flag$ do
    $\mathcal{N}$ = neighborhood of $y$;
    for $y'$ in $\mathcal{N}$ do
        $flag = False$;
        if $y' \in \mathcal{U} \cap \mathcal{G}$ and $F(y') \ge (1+\epsilon)F(y)$ then
            $y \leftarrow y'$ ;
            $flag = True$ ;
    end
end
```

Algorithm 1: The local search algorithm.

Since F is monotonic and concave, the discretization F is still a monotonic function. Previous work also proved that the discretized version of F is still submodular [20].

We solve the following relaxed problem:

$$\{F(y) \ : \ o_i y \le 1, \ \ y \in \mathcal{G} \cap \mathcal{U}, \ \ \forall i \in [k], 0 \le y_i \le 1\} \qquad (9)$$

We use a local search method to solve (9). For any $y \in \mathcal{G} \cap \mathcal{U}$, y' is in the neighborhood of y if y' differs from y by at most $2k$ elements, among which at most k elements get increased and at most k elements get decreased. The local search procedure is shown in Algorithm 1.

For $x, y \in \mathbb{R}^m$, we define $(x \vee y)_j := \max(x_j, y_j)$ and $(x \wedge y)_j := \min(x_j, y_j)$ for $j \in [m]$. Lemma 3.5 in [20] gives the following result:

LEMMA 2. *For a local optimum $y \in \mathcal{U} \cap \mathcal{G}$ and any $x \in \mathcal{U}$, we have $(2 + 2m\epsilon) \cdot F(y) \ge F(y \wedge x) + F(y \vee x) - \frac{1}{2m} f^*$ for any $\epsilon > 0$.*

Based on monotonicity, we give the following, stronger result.

LEMMA 3. *For a local optimum $y \in \mathcal{U} \cap \mathcal{G}$ and any $x \in \mathcal{U}$, we have $F(y) \ge (\frac{1}{2} - \epsilon)$**Opt** for any sufficiently small $\epsilon > 0$.*

PROOF. From monotonicity, we see that $F(y \wedge x) + F(y \vee x) \ge F(x)$ since $x \subseteq x \vee y$ and $F(y \wedge x) \ge 0$. Then, from Lemma 2, we get
$(2 + 2m\epsilon) \cdot F(y) \ge F(y \wedge x) + F(y \vee x) - \frac{1}{2m} f^* \ge F(x) + \frac{1}{2m} f^*$. Since x is an arbitrary vector, we can always let $x = x^*$ such that $F(x^*) = f^* =$ **Opt**. Thus, we have $(2 + 2m\epsilon) \cdot F(y) \ge (1 + 1/2m)$**Opt**. Dividing both sides by $(2 + 2m\epsilon)$, we have $F(y) \ge (\frac{1}{2} + \frac{1/2m - n\epsilon}{2+2m\epsilon}) \cdot$ **Opt**, which is larger than $(\frac{1}{2} - \epsilon)$**Opt** for sufficiently small ϵ.

The above result shows that our local search algorithm gives an 1/2 approximate solution for the relaxed problem.

Based on the monotonicity result, it is easy to see that for any x satisfying the knapsack constraints, . This result shows that the local search algorithm provides an 1/2-approximate solution for the relaxed problem on F. $\square$

5.3 Step 3: Rounding back to a solution

Till now, we have established an 1/2-approximation algorithm for the relaxed problem in (9). In Step 3, we round the relaxed solution back to an integer solution for the original problem P_1 using algorithm 2. In [20], Lee proved the following two lemmas which we will use directly.

```
Data: A fractional solution $y$
Result: A feasible subset $S$ of $V$
$S \leftarrow \emptyset$ ;
foreach element $e$ in $V$ do
    add $e$ to $S$ with probability $(1-\epsilon)y_e$ ;
    if $S$ violates constraints then return $\emptyset$ ;
end
return $S$ ;
```
Algorithm 2: The rounding algorithm in Step 3.

Lemma 4. *Let $\alpha(S) = \max\{o_i(S) : i \in [k]\}$, for any $a \geq 1$, $Pr[\alpha(S) \geq a] \leq k \cdot e^{-cak^2}$.*

Lemma 5. *For any $a \geq 0$, $\max\{f(S) : \alpha(S) \leq a+1\} \leq 2(1+\delta)k(a+1)\mathbf{Opt}$.*

Now we prove the following theorem, which concerns the expected value of $f(S)$, assuming $f(\emptyset) = 0$.

Theorem 2. *The simple rounding algorithm for light elements obtains an expected value $E[f(S)] \geq (1/2 - \eta)\mathbf{Opt}$.*

Proof. Consider disjoint events $A_0 = \{\alpha(S) \leq 1\}$ and $A_l = \{\alpha(S) \in (l, 1+l]\}$ for $l \in \mathbb{N}$. We use **ALG** to denote $E[f|A_0]Pr[A_0]$. Obviously, **ALG** is the expected objective function value for this algorithm's output, as we need $\alpha(S) \leq 1$ to to satisfy the knapsack constraints.

According to Lemma 4 and Lemma 5, we have,

$$\begin{aligned} F(x) &= E[f] = E[f|A_0]Pr[A_0] + \sum_{l\geq 1} E[f|A_l]Pr[A_l] \\ &\leq \mathbf{ALG} + \sum_{l\geq 1} ke^{-clk^2} 2(1+\delta)k(l+1)\mathbf{Opt} \\ &\leq 8\mathbf{Opt} \cdot l \cdot k^2 \cdot e^{-clk^2} \end{aligned}$$

Thus, we have

$$\mathbf{ALG} = F(x) - \sum_{l\geq 1} E[f|A_l]Pr[A_l] \geq F(x) - \frac{8}{c}\mathbf{Opt}$$

From Lemma 3 we know that $F(x) \geq (1/2 - \epsilon)\mathbf{Opt}$. Let $\eta/2 = \epsilon$, we get that $\mathbf{ALG} \geq 1/2\mathbf{Opt} - (\eta/2 + 8/c)\mathbf{Opt} = (1/2 - \eta)\mathbf{Opt}$. $\square$

Now we give an approximation bound for the overall solution after combining heavy and light elements.

Theorem 3. *Our fractional relaxation greedy (FRG) algorithm is a $(\frac{1}{3} - \eta)$-approximation algorithm for solving $P1$.*

Proof. Recall that we use H and L to denote the heavy and light elements in an optimal integer solution, respectively. The enumeration of heavy elements in Step 1 of FRG will give an objective function value at least $f(H)$. Theorem 2 shows that the simple rounding algorithm for light elements produces a solution of expected value at least $(1/2 - \eta) \cdot f(L)$. Thus, using the convexity of the max function, we have

$$\begin{aligned} \max\{f(H), (\frac{1}{2} - \eta)f(L)\} &\geq \frac{1}{3}f(H) + \frac{2}{3}(1/2 - \eta)f(L) \\ &\geq (\frac{1}{3} - \eta)f(H \cup L). \end{aligned}$$

The above inequality shows the approximation bound.

Our $(1/3 - \eta)$ bound is better than the $(1/5 - \epsilon)$ bound in [20]. Another work in optimizing monotonic submodular functions subject to knapsack or general linear constraints is proposed by Kulik *et al.* [19] that has $(1-\epsilon)(1-e^{-1})$ approximation bound. While this theoretical bound is better, we found it not practical for solving our shared resource optimization problem. The proposed algorithm requires an enumeration of all possible assignment set T such that $|T| \leq ek\epsilon^{-3}$ followed by an optimization procedure for each assignment similar to our proposed algorithm. Consider a typical system configuration in our shared resource network with about 100 constraints, if we want an approximation bound of 1/6, ϵ here should at most be 0.73, therefore makes the number $ek\epsilon^{-3}$ be at least 600. Thus, in the worst case, all 2^{600} possible subsets of the assignment need to be enumerated, which is impractical. Our proposed algorithm, on the other hand, requires no enumeration on T. Instead of solving optimization problems multiple times for each T, our proposed algorithm only requires one round of optimization and rounding. Therefore, in general, Kulik *et al.*'s work is of theoretical significance, but impractical in solving our shared network optimization problem.

6. EVALUATION

We evaluate the performance of our algorithm by experiments using Intel Berkley Lab's dataset [3]. In our evaluation, we compare the performance of our algorithm with that of a simulated annealing (SA) algorithm and a bin packing algorithm. As our optimization problem is essentially a discrete non-linear optimization problem, there are three categories of methods for solving it. The first is stochastic search. Simulated Annealing is a representative approach in stochastic search that can achieve global optimality asymptomatically. The second is heuristic based. Both our algorithm and bin-packing are in this family. The third category is branch and bound. However, since our objective function is not in a closed form, it is difficult to draw lower/upper bounds making branch and bound unsuitable for our problem. Therefore, we choose to compare against only SA and bin-packing, two representative algorithms for our optimization problem $P1$. SA is a probabilistic algorithm for global optimization. We have used the SA package in the AIMA library [23] with a user-defined neighborhood function such that for any application assignment y, all its neighbors in $\mathcal{N}(y)$ contain only feasible assignments. We use the default cooling factor of SA and let it run 30 minutes for each case. The bin packing algorithm sorts the applications (in non-increasing order) according to the ratio of their weights and their bandwidth demands and uses this list to sequentially assign applications to each sensor node. Our first set of experiments involves two applications (temperature monitoring and humidity monitoring) based on temperature and humidity readings of 20 sensor nodes of Intel Berkley Lab's dataset [3]. Based on this dataset, we then generate readings for different numbers of applications and different numbers of nodes representing networks of different sizes. In both of the cases, we first perform experiments considering only the bandwidth constraints of the radio. We, then, extend the experiments by adding the memory constraints of the nodes. Based on our experience in application deployment on our physical testbed of TelosB motes, we assume the channel capacity to be 30Kbps. We analyze the QoM achieved by the algorithms as well as their execution times. The baseline bin packing algorithm is implemented using Matlab. We implemented our FRG and SA algorithm in Python with Numpy package. All results are gathered on a Mac OS X machine with CPU frequency at 1.33GHz and 2GB memory.

6.1 Experiments With Intel Berkley Lab's Data

Generating kernel matrix requires sufficiently large number of observations for calculating the covariance. The Intel data has sensor readings of several months where every node generated about 1000 readings per day for every application. Thus, the readings for a particular day from all sensor nodes are enough for us to generate the kernel matrix for the applications. We pick a random date 2004-03-08 as the time window to pick sensor readings. Out of total 54 nodes, we further pick 20 nodes along with a set of time stamps such that most of these nodes have temperature and humidity readings at these time stamps. Then, for an application, we enumerate each pair of nodes to calculate the covariance between two observed sequences aligned according to time stamps.

The sensor nodes of Intel Berkley Lab are placed in a two-dimensional euclidean space. We perform the experiments on the geometry graph that represents the WSN formed by these 20 nodes as illustrated below. We scale the x and y coordinates of the nodes and randomly assign communication and interference radii in the range 15 - 20m and 15 - 22m, respectively, to all of them. These ranges of communication and interference radii have been determined based on our experiences in experiment on our physical WSN testbed. A node x whose communication radius is r can successfully transmit packets to any node whose euclidean distance from x is no more than r. On the other hand, transmission of a node x whose interference radius is r can interfere the transmission intended for any node whose euclidean distance from x is no more than r. The node on the top left of the euclidean space is considered as the sink.

We first run all three algorithms 5 times considering only the bandwidth constraints. Initially, we assume the bandwidth demands of application 1 (temperature monitoring) and application 2 (humidity monitoring) to be 700bs (bits per second) and 600bs, respectively. We pick $\eta = 0.1$ in our algorithm, which leads to the fact that almost all elements are light elements. Therefore, cost of the extensive search on heavy elements, despite being exponential, is negligible. The weights of application 1 and application 2 are considered as 2 and 1, respectively.

We then repeat our experiments by considering both bandwidth and memory constraints. We randomly assign memories in the range 1024 - 2824B to every node. We initially assume the memory demands of application 1 and application 2 to be 800B and 900B, respectively. The bandwidth demands of application 1 and application 2 are assigned to 800bs and 700bs, respectively. Application weights are same as the previous step. We repeat the experiments 5 times by increasing the memory requirement by 50B for each application in each run. The QoM and the execution times of all algorithms are recorded in Table 2.

As shown in Table 1 and Table 2, our algorithm outperforms the bin packing in most of the cases in achieving QoM, and in some cases, its quality is even better than SA that runs for 30 minutes. The execution time of our algorithm is comparable to the bin packing algorithm under the bandwidth constraint. While our algorithm is slower than bin packing under both memory and bandwidth constraints, its execution time remains within 2 seconds. Note that the execution times of both our algorithm and bin packing are three orders of magnitude shorter than the 30 min that SA is allowed to run.

The experimental result of 2 applications on 20 nodes shows that both the quality and the speed of our algorithm are competitive, but the problem scale is too small for our algorithm to show its advantage in larger systems. In the following subsection we will vary the network size and the number of applications to study the scalability of our algorithm.

6.2 Varying Network Sizes and Number of Applications

We evaluate the scalability of our algorithm by creating networks of 25, 50, and 100 nodes and for 5, 10, and 15 applications. We randomly generate the coordinates of the nodes in an euclidean space of $100m \times 100m$. Both the x-coordinate and y-coordinate of every nodes are between 0 - 100m. We assign communication and interference radii to the nodes in a way similar to the previous step. We continue generating the nodes until we get a connected graph based on the communication radii of the nodes. Since we don't have real data of sensor readings for generating the covariance matrices with different numbers of applications and nodes, we randomly generate the covariance matrix by a positive definite matrix.

For networks of different sizes and different numbers of applications, we first perform the experiments considering only the bandwidth constraints. We generate the bandwidth requirements randomly in the range 100bs - 1000bs for all applications. Weights of the applications are assigned randomly in the range 1 - $\frac{m}{2}$, where m is the number of applications. We then repeat the experiments by adding the memory constraints of the nodes. We randomly assign memories in the range 1024 - 2824B to all nodes. Memory requirements of the applications are also generated randomly in the range 600 - 1000B.

Figure 1(a) and Figure 1(b) show the QoM of different algorithms for different constraints in a network of 25 nodes. Similar results for network of 50 and 100 nodes are shown in Figure 2(a), 2(b) and Figure 3(a), 3(b), respectively. From the figures, we notice that the greedy algorithm achieves much higher QoM than bin packing always since the latter does not do any optimization. It also performs much better than SA in most of the cases. Only in few cases can SA beat our greedy algorithm. This happens because SA is run for a long period of time (30 minutes). [1] We notice that, for bin packing and greedy algorithm, the QoM increases both with the increase of network size and with the increase of number of applications which is expected. However, this does not happen to SA in many cases which is expected.

The running times of our greedy algorithm and the bin packing algorithm for varying number of applications and different network sizes are shown in Figure 4(a) and 4(b). The figures show the times required for running experiments both under single constraint and under multiple constraints.

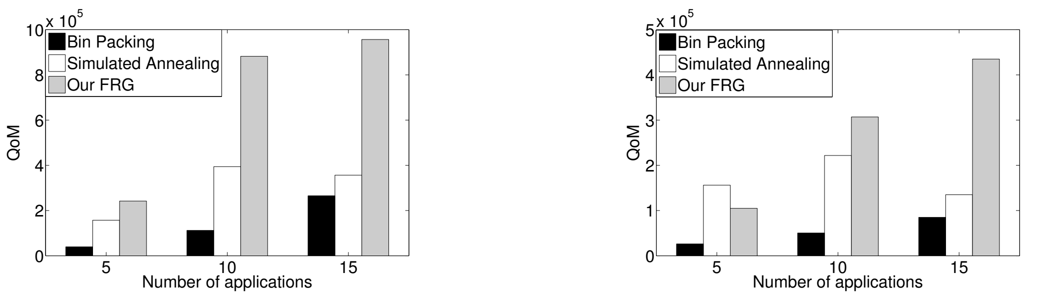
[1]Given similar running time as our algorithm, the SA algorithm in any of these cases could not converge a feasible point, as we convert constraints to penalty functions and starts from a random initial point (this is the standard treatment for applying SA to constrained optimization problem).

Run	Bin Packing		Simulated Annealing		Our FRG Algorithm	
	Objective	Time	Objective	Time	Objective	Time
1	1062.11	0.37	1225.57	1800	1293.40	0.33
2	1062.11	0.44	1178.65	1800	1217.16	0.35
3	1061.97	0.42	1145.66	1800	1151.69	0.18
4	1010.42	0.42	1094.89	1800	1017.60	0.65
5	1009.31	0.39	1032.35	1800	1009.36	0.24

Table 1: Solution quality and time (in seconds) of three algorithms under bandwidth constraints.

Run	Bin Packing		Simulated Annealing		Our FRG Algorithm	
	Objective	Time	Objective	Time	Objective	Time
1	1020.31	0.54	1221.79	1800	1112.96	1.71
2	1020.31	0.50	1005.18	1800	1023.63	1.81
3	1020.31	0.55	1071.98	1800	1020.97	1.84
4	1015.05	0.53	998.01	1800	1020.69	1.93
5	1015.05	0.58	1037.9	1800	1032.09	1.14

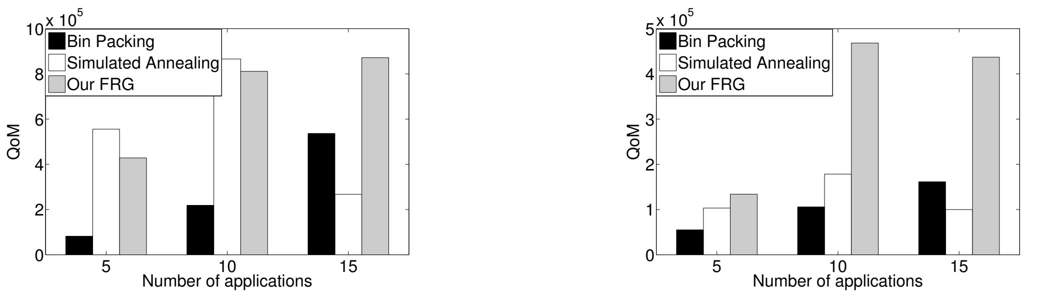
Table 2: Solution quality and time (in seconds) of three algorithms under bandwidth and memory constraints.



(a) Under bandwidth constraints

(b) Under bandwidth & memory constraints

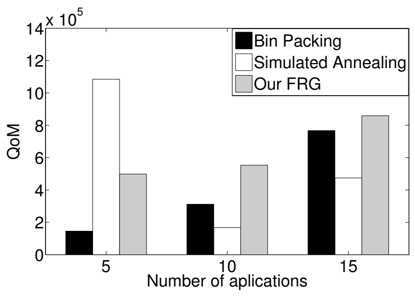
Figure 1: QoM in 25-node network



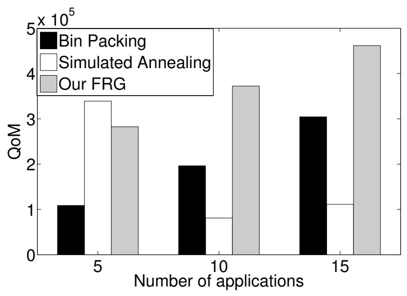
(a) Under bandwidth constraints

(b) Under bandwidth & memory constraints

Figure 2: QoM in 50-node network

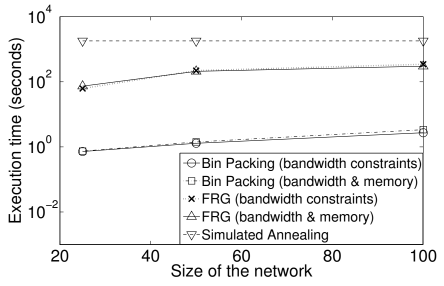


(a) Under bandwidth constraints

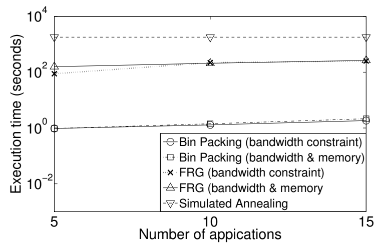


(b) Under bandwidth & memory constraints

Figure 3: QoM in 100-node network



(a) Different network sizes (for 10 applications)



(b) Different number of applications (for 50 nodes)

Figure 4: Execution times for different number of applications and different network sizes

Note that we let the SA algorithm run for 30 minutes in every case. Figure 4(a) shows the execution times for allocating 10 applications on a network by varying its size. Figure 4(b) shows the execution times for allocating different number of nodes on a network of 50 nodes. Our greedy algorithm scales well with the number of sensor nodes (Fig. 4(a)) and the number of applications (Fig. 4(b)). For example, it only takes 301 seconds to allocate 10 applications in a 100-node network subject to both bandwidth and memory constraints. This result demonstrates our algorithm provides a practical solution for application allocation in shared sensor networks.

7. CONCLUSION

In this paper, we address the multi-application allocation problem in shared sensor networks. We prove the monotonicity and submodularity of a common QoM function called variance reduction. By exploiting these properties, we propose an efficient local search algorithm for multi-application allocation in shared sensor networks. Our allocation algorithm has rigorous approximation bounds and is generally applicable to any QoM functions with submodular and monotonic proprieties. We verify the theoretical properties on a real-world dataset from the Intel Research Berkeley Lab. Simulations based on both real-world dataset and randomly generated networks demonstrate that our algorithm care competitive against simulated annealing in term of QoM, with up to three orders of magnitude reduction in execution times. Our results demonstrate that our algorithm is a practical choice for application allocation in shared sensor networks.

8. ACKNOWLEDGMENTS

This work is supported by NSF NeTS-NOSS Grant CNS-0627126.

9. REFERENCES

[1] http://research.cens.ucla.edu/areas/2005/NIMS/.

[2] `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datash%eet.pdf`.

[3] `http://db.csail.mit.edu/labdata/labdata.html`.

[4] S. Bhattacharya, A. Saifullah, C. Lu, and G.-C. Roman. Multi-application deployment in shared sensor networks based on quality of monitoring. In *RTAS'10*.

[5] F. Bian, D. Kempe, and R. Govindan. Utility-based sensor selection. In *IPSN'06*.

[6] J. Byers and G. Nasser. Utility-based decision-making in wireless sensor networks. In *MobiHoc'00*.

[7] Citysense. http://www.citysense.net/.

[8] M. Conforti and G. CornuŐjols. Submodular set functions, matroids and the greedy algorithm: Tight worst-case bounds and some generalizations of the rado-edmonds theorem. *Discrete Applied Mathematics*, 7(3):251 – 274, 1984.

[9] U. Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, 1998.

[10] M. Fisher, G. Nemhauser, and L. Wolsey. An analysis of approximations for maximizing submodular set

functions-ii. *Mathematical Programming Study*, 8:73–87, 1978.

[11] C. Frank and K. Römer. Algorithms for generic role assignment in wireless sensor networks. In *SenSys'05*.

[12] L. Wolsey G. Nemhauser and M. Fisher. An analysis of approximations for maximizing submodular set functions-i. *Mathematical Programming*, 14(1):265–294, 1978.

[13] C. Guestrin, A. Krause, and A. Singh. Near-optimal sensor placements in gaussian processes. In *ICML'05*.

[14] J. Horey, A. Maccabe, and A. Mielke. A dynamic tasking architecture for sensor networks. In *WWSNA'07*.

[15] V. Isler and R. Bajcsy. The sensor selection problem for bounded uncertainty sensing models. In *IPSN'05*.

[16] A. Krause and A. Gupta. Near-optimal sensor placements: Maximizing information while minimizing communication cost. In *IPSN'06*, pages 2–10.

[17] A. Krause, J. Leskovec, C. Guestrin, J. VanBriesen, and C. Faloutsos. Efficient sensor placement optimization for securing large water distribution networks. *Journal of Water Resources Planning and Management*, 134(6):516–526, November 2008.

[18] A. Krause, B. McMahan, C. Guestrin, and A. Gupta. Robust submodular observation selection. *Journal of Machine Learning Research (JMLR)*, 9:2761–2801, December 2008.

[19] A. Kulik, H. Shachnai, and T. Tamir. Maximizing submodular set functions subject to multiple linear constraints. In *SODA'09*.

[20] J. Lee, V. Mirrokni, V. Nagarjan, and M. Sviridenko. Non-monotone submodular maximization under matroid and knapsack constraints, 2009.

[21] Q. Ma, D. Parkes, and M. Welsh. A utility-based approach to bandwidth allocation and link scheduling in wireless networks. In *ATSN'07*.

[22] G. Mainland, D. Parkes, and Welsh M. Decentralized, adaptive resource allocation for sensor networks. In *NSDI'05*.

[23] S. Russel and P. Norvig. *Artificial Intelligence: A Modern Approach, 2nd Edition*. Prentice-Hall, 2003.

[24] M. Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41 – 43, 2004.

[25] Y. Yu, L. Rittle, V. Bhandari, and J. LeBrun. Supporting concurrent applications in wireless sensor networks. In *SenSys'06*.

[26] G. Zhou, T. He, J. Stankovic, and T. Abdelzaher. RID: radio interference detection in wireless sensor networks. In *INFOCOM'05*.