

Submodular Game for Distributed Application Allocation in Shared Sensor Networks

Chengjie Wu, You Xu, Yixin Chen, Chenyang Lu
Department of Computer Science & Engineering
Washington University in St. Louis
{wu, yx2, chen, lu}@cse.wustl.edu

Abstract—Wireless sensor networks are evolving from single-application platforms towards an integrated infrastructure shared by multiple applications. Given the resource constraints of sensor nodes, it is important to optimize the allocation of applications to maximize the overall Quality of Monitoring (QoM). Recent solutions to this challenging application allocation problem are centralized in nature, limiting their scalability and robustness against network failures and dynamics. This paper presents a distributed game-theoretic approach to application allocation in shared sensor networks. We first transform the optimal application allocation problem to a submodular game and then develop a decentralized algorithm that only employs localized interactions among neighboring nodes. We prove that the network can converge to a pure strategy Nash equilibrium with an approximation bound of $1/2$. Simulations based on three real-world datasets demonstrate that our algorithm is competitive against a state-of-the-art centralized algorithm in terms of QoM.

I. INTRODUCTION

Traditionally, wireless sensor networks (WSNs) are used as specialized platforms where only a single application is deployed on each sensor. Recently, large-scale, integrated WSNs that support multiple applications start to emerge. Many application domains such as urban sensing [1], building automation and environmental monitoring [2] have already adopted the integrated WSN paradigm to support multiple applications. Compared to separate application-specific sensor networks, a shared WSN 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.

Severe resource constraints limit the allocation of all possible applications to sensors in a shared WSN. For example, the TelosB mote [3], a representative sensor platform, only has 10 KB of RAM, a 250 Kbps radio, and a 16-bit CPU running at 8 MHz. On the other hand, the Quality of Monitoring (QoM) of applications depends on application allocations. Therefore, it is important to optimize the allocation of multiple applications among sensor nodes in order to maximize the overall QoM, subject to resource constraints. This problem is challenging because it is essentially a discrete optimization with an exponentially large solution space.

Some recent works utilize the submodularity of the QoM function to tackle this discrete optimization problem. Submodularity is an important property of the QoM functions for

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 to the function values than adding it to a larger subset of S . The submodularity of QoM 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 [4] and water quality [5]–[7] has been observed in previous studies of real-world datasets.

Many existing works centered around submodular optimization have been proposed for optimization problems in sensor networks. Recent theoretical works also show approximation algorithms that can achieve a $(1 - 1/e)$ -approximation bound [8]. Xu *et al.* [9] proposed a greedy algorithm and achieved a $1/3$ approximation bound. Submodular optimization approaches are also used in sensor selection and placement applications [6], [7]. However, all these existing submodular optimization approaches are essentially *centralized* solutions. For WSNs, a centralized algorithm implies there is either a node or gateway that maintains the global information of the network.

A centralized approach is not desirable for WSNs due to its limitations in scalability and fault tolerance. First, a shared WSN is usually of large scale in terms of the number of nodes and hop counts. Hence, it is inefficient or even impossible to achieve global information sharing that is required by centralized optimization algorithms. Second, in a centralized approach, much of the computation and communication happens on a single point resulting in a single point of failure. To address the limitations of centralized approaches, we study *distributed* optimization approaches for application allocations. Meanwhile, we still exploit the submodularity property of QoM functions to achieve desirable approximation bounds.

In this paper, we provide several major theoretical results: 1) We propose the *covariance cover* function as a new QoM metric that is amenable to distributed optimization; 2) We show that the optimal multi-application allocation problem with covariance cover as objective function is a submodular optimization problem with multiple knapsack constraints; 3) We

propose a game theory based distributed algorithm for solving this submodular optimization problem and prove that our algorithm can achieve a $1/2$ -approximation bound when each sensor achieves optimal allocation of applications. We also prove our algorithm can achieve a $1/(1 + \beta)$ -approximation bound when each sensor achieves a $1/\beta$ -approximate allocation of applications. Simulations based on three real-world datasets demonstrate that our distributed algorithm can achieve comparable QoM as a state-of-the-art centralized algorithm [9], while scaling effectively in terms of both execution time per node and the communication overheads.

II. RELATED WORK

Originated from centralized optimization, subgradient methods have been used to optimize problems where the gradients of the objectives are hard to obtain, while the subgradients of objective functions with respect to a subset of variables are easy to obtain [10], [11]. The subgradient optimization method can be used as a distributed optimization algorithm for problems in WSNs, in which each sensor node optimizes the objective function distributively using its own subgradient value. However, the subgradient method is not suitable for the multi-application allocation problem in a large-scale, multi-hop WSN, due to the fact that in each iteration of the algorithm, it is still required to propagate the solution for the subsequent subgradient calculation. That is to say, although the optimization is localized to each sensor, global communication is still required.

Game-theoretic approaches have been proposed to address the above issue. In these approaches, communications are made only between certain sensors in a user-defined neighborhood. Another unique property of game-theoretic approaches is that they do not assume that agents (in this case, sensor nodes) work cooperatively. Instead, selfish sensor nodes optimize a local version of the objective functions, often called “utilities” or “private utilities”, independently, until none of them can further improve their private utilities by making a different decision.¹ When these utilities are carefully designed to reflect the objective function, the overall objective function, also called “social utility”, is subsequently optimized by these noncooperative agents [12].

When the social and private utilities are carefully designed, game-theoretic approaches guarantee a constant optimization bound [12]–[14]. Since the utility system decides the nature of the game, needless to say, for game-theoretic approaches to work for multi-application allocation problems, designing the utility system is critical. Specifically, in a distributed solution, a utility system that is easy to calculate and has no global information propagation requirements is desirable. In other words, in the application allocation problem, a utility system should reflect the QoM value based on the decisions of each sensor, while it does not require global communication in the network.

¹In Game Theory, a state where no player can improve its utility is called an equilibrium state.

Previous works proposed different QoM formulations [5], [9], [15], including variance reduction and mutual information gain. However, neither is suitable as the objective function in a distributed game-theoretic approach, which requires that a sensor’s utility is independent of other sensors that are not in its neighborhood. This condition is violated when using variance reduction or mutual information as QoM, because one sensor’s utility of allocating an application is related to all sensors that carry the application. We address this issue by proposing a new QoM metric that is submodular and suitable for game-theoretic distributed optimization while serving as an effective proxy for variance reduction in QoM optimization.

III. PROBLEM FORMULATION

In this section, we first review the *variance reduction* QoM formulation. After discussing the disadvantages of using variance reduction in distributed algorithms, we propose a new QoM metric called *covariance cover* that is amenable to distributed solutions. In the end, we formulate the application allocation problem in shared WSNs using covariance cover as QoM metric.

A. QoM Formulation

Variance reduction is commonly used to measure QoM in WSN applications [5], [9]. Assuming sensor readings follow a Gaussian Process, the variance reduction measures how much the variance of the readings from the unallocated sensors.

Variance reduction is calculated based on covariance. Assuming K is the covariance matrix for sensor nodes, and for two subsets of sensor nodes $G, H \subseteq V$, the covariance matrix of G and H is denoted by K_{GH} , where its rows corresponding to G and columns corresponding to H extracted from K . For a given set G with application allocated, the variance of the unassigned set $\bar{G} = V \setminus G$ is

$$\sigma_{\bar{G}|G}^2 = \text{tr}(K_{\bar{G}\bar{G}}) - \text{tr}(K_{\bar{G}G}K_{GG}^{-1}K_{G\bar{G}}),$$

where $\text{tr}()$ is the trace of a matrix.

In this application allocation problem, the goal is to minimize the variance of \bar{G} given G such that the quality of sensing is maximized. Namely, we want to maximize the negation of the variance. Given $\text{tr}(K) = \text{tr}(K_{GG}) + \text{tr}(K_{\bar{G}\bar{G}})$, variance reduction for one application is:

$$Q_{VR} = \text{tr}(K_{GG}) + \text{tr}(K_{\bar{G}G}K_{GG}^{-1}K_{G\bar{G}}) \quad (1)$$

Variance reduction is just one of the many possible ways to formulate QoM. There is an inherent disadvantage of using variance reduction for our problem. It is not feasible for a sensor node with limited memory resource to store a kernel matrix that is quadratic to the size of the network, and it is expensive to compute variance reduction since it involves matrix multiplication and inversion. To overcome this inherent disadvantage, we decompose the variance reduction and propose a new formulation which is more amenable to distributed approaches.

We begin with introducing the network model. A network consists of a group of sensors $\{1, 2, \dots, n\}$. Each sensor node

can be presented as a vertex. For a pair of sensors i and j , if each of them is in the other's communication range, i and j are defined as a pair of neighbors. The network can be presented as a graph $G = (V, E)$, where $V = \{1, 2, \dots, n\}$ and $(i, j) \in E$ if and only if i and j are a pair of neighbors. Now we will decompose the variance reduction based on two assumptions.

Theorem 1. *Variance reduction formulation (1) is equivalent to*

$$\sum_{i \in G} K_{ii} + \sum_{(i,j) \in E, i \in G \text{ or } j \in G} K_{ij}^2,$$

if (I) the covariance of any two nodes is nonzero if and only if they are a pair of neighbors, and (II) any two allocated nodes are not a pair of neighbors.

Proof: Let us first simplify the variance reduction formulation. Since any two allocated nodes are not neighbors of each other, and only neighbors have nonzero variance, we can prove K_{GG} is an identity matrix. It immediately follows that

$$\begin{aligned} Q_{VR} &= \text{tr}(K_{GG}) + \text{tr}(K_{\bar{G}G} K_{GG}^{-1} K_{G\bar{G}}) \\ &= \sum_{i \in G} K_{ii} + \sum_{i \in G, j \in \bar{G}} K_{ij}^2. \end{aligned}$$

Since only a pair of neighbors have nonzero covariance,

$$Q_{VR} = \sum_{i \in G} K_{ii} + \sum_{i \in G, j \in \bar{G}, (i,j) \in E} K_{ij}^2.$$

We assume two allocated nodes are not neighbors, which means if $(i, j) \in E, i \in G$, then $j \in \bar{G}$. It follows

$$Q_{VR} = \sum_{i \in G} K_{ii} + \sum_{(i,j) \in E, i \in G \text{ or } j \in G} K_{ij}^2. \quad \square$$

One question raises naturally: how realistic are the assumptions? We argue that the proposed two assumptions, although sometimes violated, provide good approximations of the real-world scenarios. First, it is reasonable to assume a pair of nearby sensors have larger covariance. For example, the temperature measurements of two different sensors in the same office room are more correlated than two sensors in different rooms. Second, since our applications have the inherent property of submodularity, allocating two neighboring nodes simultaneously typically does not give much gain in terms of QoM. To maximize QoM, a good solution should naturally allocate nodes that have unallocated nodes as neighbors. Actually, our submodular game algorithm is not limited by these two assumptions, it can handle situations when assumptions do not hold.

We name the new QoM formulation *covariance cover*. Denoting $\tau_{ij} = K_{ij}^2$ as the weight of edge (i, j) , and $\tau_{ii} = K_{ii}$ as the weight of node i , we define the covariance cover formulation as

$$Q_{CC} = \sum_{i \in G} \tau_{ii} + \sum_{(i,j) \in E, i \in G \text{ or } j \in G} \tau_{ij}. \quad (2)$$

B. Application Allocation Problem Formulation

Given QoM metric as covariance cover, we want to further formulate the application allocation problem in shared sensor networks.

When there are multiple applications $P = \{1, 2, \dots, p\}$ with weights $\{w^1, w^2, \dots, w^p\}$, we want to maximize the summation QoM of all applications $\sum_{t=1}^p w^t Q^t$, where Q^t is the covariance cover for application t .

$$Q^t = \sum_{i \in G^t} \tau_{ii}^t + \sum_{(i,j) \in E, i \in G^t \text{ or } j \in G^t} \tau_{ij}^t$$

This problem is challenging because of critical resource constraints, e.g., CPU and memory constraints. For each sensor, the total memory and CPU consumed by all applications can not exceed its limits. Therefore, suppose each node has m resource constraints $R = \{1, 2, \dots, m\}$, the capacity of node i on resource k is $C_{i,k}$, and application t consumes $c_{i,k}^t$ units of resource k on node i , the constrained optimization problem can be formulated as:

$$\begin{aligned} \max \quad & QoM = \sum_{t \in P} w^t Q^t \\ Q^t = & \sum_{i \in G^t} \tau_{ii}^t + \sum_{(i,j) \in E, i \in G^t \text{ or } j \in G^t} \tau_{ij}^t \\ \text{s.t.} \quad & \sum_{t \in G^t} c_{i,k}^t \leq C_{i,k}, \quad \forall i \in V, \forall k \in R \end{aligned}$$

here G^t is the set of nodes which are assigned application t .

It is easy to see that all resource constraints here are knapsack constraints. This type of constraint formulation also 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.

IV. SUBMODULAR GAME

In this section, we will formulate a non-cooperative game based on the covariance cover formulation discussed in the previous section, which leads to a completely distributed algorithm. We introduce typical terminologies in game theory at first.

A. Submodular Game Formulation

Suppose we have n sensor nodes, and each sensor node i in the network is an agent i in the game. For each sensor, its strategy a_i is the subset of applications that can run on it.

$$\begin{aligned} a_i &= \{t \mid \text{application } t \text{ runs on sensor } i\} \\ &= \{t \mid i \in G^t, \forall t \in P\}. \end{aligned}$$

Under the resource constraint we discussed earlier, the *strategy set* \mathcal{A}_i of player i is

$$\mathcal{A}_i = \{a_i \mid \sum_{t \in a_i} c_{i,k}^t \leq C_{i,k}, \forall k \in R\},$$

A *pure strategy* is one in which each agent decides to carry out a specific strategy. In game theory, *mixed strategy* is also widely discussed. However, we only discuss pure strategy in

this paper, because in reality of sensor networks, it is hard to implement strategies with probability distribution. Also, we prove that our game has at least one Nash equilibrium with pure strategies. We denote the *strategy space* of the game as $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_n$.

A game is always defined on a utility system. To build the utility system, we need to define the utility function at first. Given a strategy profile $A = (a_1, a_2, \dots, a_n) \in \mathcal{A}$, let $A \oplus a'_i$ denote the strategy profile obtained if agent i changes its strategy from a_i to a'_i . Formally, $A \oplus a'_i = (a_1, \dots, a_{i-1}, a'_i, \dots, a_n)$.

The goal of our game is to maximize the *social utility* $\gamma : 2^V \rightarrow \mathcal{R}$ defined on pure strategy profile $A = \{a_1, \dots, a_n\}$ as

$$\begin{aligned} \gamma(A) &= \sum_{t=1}^p \gamma^t(A) \\ &= \sum_{t=1}^p w^t \left(\sum_{t \in a_i \text{ or } t \in a_j, (i,j) \in E} \tau_{ij}^t + \sum_{t \in a_i, i \in V} \tau_{ii}^t \right) \quad (3) \\ &= \sum_{t=1}^p w^t \left(\sum_{(i,j) \in E, i \in G^t \text{ or } j \in G^t} \tau_{ij}^t + \sum_{i \in G^t} \tau_{ii}^t \right). \end{aligned}$$

Remind G^t is the set of nodes who are assigned application t .

For each agent i , we define a *private utility* $\phi_i : 2^V \rightarrow \mathcal{R}$ as:

$$\begin{aligned} \phi_i(A) &= \sum_{t \in a_i} \phi_i^t(A) \\ &= \sum_{t \in a_i} w^t \left(\tau_{ii}^t + \sum_{j \in \mathcal{N}_i} \frac{\tau_{ij}^t}{1 + \delta_{j \in G^t}} \right) \quad (4) \end{aligned}$$

where $\mathcal{N}_i = \{j | (i, j) \in E\}$ is sensor i 's neighborhood. For edge (i, j) , if not only i , but also j runs application t , (i, j) 's edge weight τ_{ij}^t need to be equally shared by both i and j . Otherwise, sensor i will account all (i, j) 's edge weight into its private utility.

The goal of each sensor is, therefore, to select a strategy in order to maximize its private utility under resource constraints. Clearly, such strategies may not produce a good solution with respect to the social utility γ . However, we will show that the strategies sensors finally select will result in a reasonable good social utility γ in next section.

To localize the optimization problem to each sensor, given strategies of its neighbors fixed, we redefine sensor i 's private utility $\phi_i(A)$ as its utility function $u_i(x_i)$, which is a function of its own decisions x_i . Its decision $x_i = \{x_i^1, \dots, x_i^p\}$ is redefined from its strategy a_i , where $x_i^t = 1$ means $t \in a_i$.

$$u_i(x_i) = \sum_{t=1}^p w^t \left[\tau_{ii}^t + \sum_{j \in \mathcal{N}_i} \frac{\tau_{ij}^t}{1 + \delta_{j \in G^t}} \right] x_i^t$$

We denote $\Omega_i^t = \left[\tau_{ii}^t + \sum_{j \in \mathcal{N}_i} \frac{\tau_{ij}^t}{1 + \delta_{j \in G^t}} \right]$ as a constant, assuming strategies of i 's neighbors are given. To maximize

its utility function, sensor node i needs to solve a integer programming problem:

$$\begin{aligned} \text{Max} \quad & u_i(x_i) = \sum_{t=1}^p w^t \Omega_i^t x_i^t \\ \text{where} \quad & x_i^t \in \{0, 1\}, \forall t \in P \quad (5) \\ \text{s. t.} \quad & \sum_{t=1}^p c_{i,k}^t x_i^t \leq C_{i,k} \quad \forall k \in R \end{aligned}$$

Actually, this is a typical multidimensional knapsack problem. There is a rich package of literature to solve this problem. We propose two algorithms based on p , which is the number of applications. If p is not larger than T_p , our solution will adopt a naive enumeration algorithm. Basically, it enumerates all possible application assignments and returns an optimal solution. Otherwise, our solution will adopt a polynomial time approximate algorithm, which is proven to have a $\frac{1}{1+m}$ approximation bound (section 9.4.2 of [16]), where $m = |R|$ is the number of resource constraints. Here T_p is the threshold for p , we set it to 5 in our implementation. We show the sketch of our solution for problem (5) in Algorithm 1.

Algorithm 1: Algorithm for knapsack problem (5)

Set $\hat{x} = \{0, \dots, 0\}$;

if $p \leq T_p$ **then**

Adopt the Enumeration Algorithm;

 Enumerate $x \in \{0, 1\}^p$, return optimal solution \hat{x} .

else

Adopt the Approximation Algorithm;

 Relax problem (5) to a linear programming problem and compute an optimal solution x_{LP} of the LP-relaxation.

 Set $I = \{t | x_{LP}^t = 1\}$

 Set $F = \{t | 0 < x_{LP}^t < 1\}$

 Return $\hat{x} = \max\{\sum_{t \in I} w^t, \max\{w^t | t \in F\}\}$

end

Now we analyze computational cost of our algorithm. If $p \leq T_p$, the time complexity is $O(\binom{p}{d})$ where d is the maximum number of applications that can be allocated on one node. And if p is larger than T_p , the relaxed linear programming (LP) problem is significantly simplified due to the small numbers of resource constraints as well as applications. The number of resource constraints is usually no more than 3 (e.g., memory, CPU, and bandwidth). The number of applications is also small due to the limited resources available per node. Our algorithm employs an efficient and practical solution as follows. We solve the dual problem of the aforementioned LP problem which only has three variables (the shadow prices of memory, CPU and bandwidth constraints) and p constraints (p is the number of applications). Even a naive LP solver that enumerates all possible extreme points (each of the three constraints determines one extreme point) and finds the best feasible one has the computational complexity $O(p \binom{p}{3}) = O(p^4)$, and the memory requirement

of the naive enumeration algorithm is $O(1)$. Either way, the cost of each individual multidimensional knapsack problem is $O(p^d)$, where d is a small integer.

B. Submodular Game Algorithm

Now we discuss our distributed submodular game algorithm. In the beginning stage, sensor nodes in the network need to get two key parameters about applications set P : 1) types of required sensor readings; 2) the frequency of each sensor reading. These two key parameters are distributed to the network from a central facility like base station. After this stage, no central facility is needed in the algorithm, so our algorithm is fully distributed.

Algorithm 2: Game algorithm for sensor node i

initialization;

- i measures sensor readings for each application t ;
- i broadcasts all sensor readings in its neighborhood;
- i calculates τ_{ii}^t and τ_{ij}^t for every neighbor j ;

if Timer Λ_i fires then

if receiving strategy changes from neighbors then
 i runs algorithm 1, output $\hat{x} \rightarrow$ strategy;
if strategy changes then
 i broadcasts its strategy in neighborhood;
end

end

end

In the initialization stage, each node measures sensor readings for a certain interval and broadcasts sensor readings in its neighborhood. Based on neighbor j 's readings, node i can calculate the covariance between i and j as well as τ_{ij}^t .

Algorithm 2 shows the detailed decision-making procedure for each sensor. In each round of the game, nodes share the same time interval T . Each node generates a random number Λ_i ($\Lambda_i < T$) as a timer using a unique seed, such that two timers will not fire at the same time. Each timer Λ_i will fire once and only once during each time interval T for sensor node i to solve the allocation problem (5) locally. If a new strategy is generated, node i will broadcast it in the neighborhood. Otherwise i will keep quiet. Each sensor node i also receives messages from its neighbors about their updated strategies. The algorithm terminates when no strategy changes are made in a round.

Here we analyze the efficiency of our game algorithm. From the computational cost perspective, we already give the computational cost of each node in each round as $O(p^d)$. Since both p and d are small integers, it is reasonable to say the computational cost is acceptable on a sensor node with limited resources. From a network perspective, we want to analyze the communication cost. In each round, sensor node i needs to receive messages from all its neighbors and broadcast its own strategy in its neighborhood if necessary. We denote the *Expected Transmission Count (ETX)* of link e is ν_e . Since sensor node i broadcasts in the neighborhood, in the worst

case, the number of messages it needs to send is the maximum of all the ETXs in its neighborhood $\zeta_i = \max_{j \in \mathcal{N}_i} \nu_{(i,j)}$. So the overall number of packets sensor i sends in the game is lower than $\kappa \zeta_i$, given κ is the number of overall number of rounds. Because κ is always a small number (less than 12) based on our evaluation, the communication cost is relatively small.

V. CONVERGENCE AND APPROXIMATION BOUND

In this section, we first show the social utility (3) is submodular. Then we prove our submodular game $(\gamma, \cup_{i \in V} \phi_i)$ defined in (3) and (4) can converge to a pure strategy Nash equilibrium with an approximation bound of $\frac{1}{2}$, if sensors use the enumeration algorithm to solve the multidimensional knapsack problem (5). If sensors use the $\frac{1}{1+m}$ -approximate solution for the knapsack problem, the game can converge to a $(1+m)$ -approximate pure strategy Nash equilibrium and the approximation bound is $\frac{1}{2+m}$, where m is number of resource constraints.

A. Submodularity

Definition 1. (Submodularity) Let V be a finite set, a function $f : 2^V \rightarrow R$ is submodular if for any $A \subseteq B \subseteq V$ and $x \in V - B$, $f(B \cup \{x\}) \leq f(A \cup \{x\})$.

Recall that we defined the social utility (3) as a function of pure strategy profiles in the last section. We redefine the social utility here as a function of the set of sensors that we allocate applications on: $\gamma = \sum_{t=1}^p w^t Q^t(G^t)$, where

$$Q^t(G^t) = \sum_{\{(i,j)|i \in G^t \text{ or } j \in G^t\}} \tau_{ij}^t + \sum_{i \in G^t} \tau_{ii}^t.$$

This definition is equivalent to the one we defined in (3), but it is now defined on the set of sensors. Based on this set based definition, we can prove the social utility γ is submodular.

Theorem 2. *The social utility γ is submodular.*

Proof: Since $\gamma = \sum_{t=1}^p w^t Q^t(G^t)$, we only need to show $Q^t(G^t), \forall t \in P$ is a submodular function. By definition, we need to prove: if $A \subseteq B \subseteq V$ and $x \in V - B$, $f(B \cup \{x\}) \leq f(A \cup \{x\})$.

If $x \in B$, it is obvious that $Q^t(B \cup \{x\}) - Q^t(B) = 0 \leq Q^t(A \cup \{x\}) - Q^t(A)$.

If $x \notin B$, it follows:

$$\begin{aligned} & A \subseteq B \\ \Rightarrow & \{(i,k)|i = x \ \& \ k \notin B\} \subseteq \{(i,k)|i = x \ \& \ k \notin A\} \\ \Rightarrow & \sum_{i=x \ \& \ k \notin B} \tau_{ik}^t \leq \sum_{i=x \ \& \ k \notin A} \tau_{ik}^t \\ \Rightarrow & Q^t(B \cup \{x\}) - Q^t(B) \leq Q^t(A \cup \{x\}) - Q^t(A) \quad \square \end{aligned}$$

B. Convergence and Pure Nash Equilibrium

Now we will discuss the convergence of our game. By defining a potential function, we show the increase of each agent's private utility will lead to the increase of the potential function. Then we can prove our game will converge at a *pure strategy Nash equilibrium*. Here we assume our *Submodular*

Game Algorithm (Algorithm 2) is using the enumeration algorithm.

Definition 2. (Pure Strategy Nash Equilibrium) A pure strategy profile $A \in \mathcal{A}$ is a pure strategy Nash equilibrium if no agent has an incentive to change its strategy. For any agent i ,

$$\forall a'_i \in \mathcal{A}_i, \phi_i(A \oplus a'_i) \leq \phi_i(A).$$

Equivalently, given the other agents' strategies, a_i is the best response of agent i .

Theorem 3. A pure strategy Nash equilibrium always exists for the utility system $(\gamma, \cup_i \phi_i)$ we defined in (3) and (4). And Submodular Game Algorithm (Algorithm 2) converges to a pure strategy Nash equilibrium.

Proof: The proof starts from defining the potential function of the game. We define the potential function ψ for a strategy profile A as

$$\psi(A) = \sum_{t=1}^p w^t \left(\sum_{i \in V, t \in a_i} \tau_{ii}^t + \sum_{(i,j) \in E, t \in a_i \text{ or } t \in a_j} \sum_{l=1}^{n_{(i,j)}^t} \frac{\tau_{ij}^t}{l} \right)$$

where $n_{(i,j)}^t$ is the number of agents which are assigned application t as well as the end points of edge (i,j) . $n_{(i,j)}^t$ is 2 if both i and j are assigned application t , and it is 1 if only one of i and j is assigned the application t .

Assume sensor i changes its strategy from a_i to a'_i , as a result the strategy profile of the game changes from A to A' . Here a_i is the set of applications which are assigned to sensor i in original strategy profile A , and a'_i is that in new strategy profile A' . Let $G = a_i - a'_i$ and $H = a'_i - a_i$. We use E_i to denote the set of edges which coincide with sensor i . Since the change only happens on node i and edges sit on i , we will ignore other nodes and edges in following proof.

$$\begin{aligned} & \psi(A') - \psi(A) \\ = & \sum_{t \in a'_i} w^t \tau_{ii}^t - \sum_{t \in a_i} w^t \tau_{ii}^t + \\ & \sum_{j \in \mathcal{N}_i} \sum_{t \in H} w^t \frac{\tau_{ij}^t}{n_{(i,j)}^t + 1} - \sum_{j \in \mathcal{N}_i} \sum_{t \in G} w^t \frac{\tau_{ij}^t}{n_{(i,j)}^t}; \\ = & \phi_i(A') - \phi_i(A) \\ = & \sum_{t \in a'_i} w^t \tau_{ii}^t - \sum_{t \in a_i} w^t \tau_{ii}^t + \\ & \sum_{t \in H} w^t \sum_{j \in \mathcal{N}_i} \frac{\tau_{ij}^t}{n_{(i,j)}^t + 1} - \sum_{t \in G} w^t \sum_{j \in \mathcal{N}_i} \frac{\tau_{ij}^t}{n_{(i,j)}^t} \end{aligned}$$

Obviously, $\psi(A') - \psi(A) = \phi_i(A') - \phi_i(A)$, we prove that the increase of the private utility of i is exactly the same as increase of the potential function of the game.

Once each individual sensor improves its private utility, the potential function ψ of the game also gets increased. Since the maximum value of this potential function is finite, the algorithm will converge in finite rounds. \square

C. Valid Utility Game and Approximate Nash Equilibrium

Now we want to prove our submodular game $(\gamma, \cup_{i \in V} \phi_i)$ is a valid utility system.

Definition 3. (Utility System) [12] A game is called a utility system if and only if the private utility of an agent is at least as great as the loss in social utility resulting from the agent dropping out of the game. That is, the game $(\gamma, \cup_i \phi_i)$ is a utility system if and only if it has the property $\phi_i(A) \geq \gamma'_{a_i}(A \oplus \emptyset_i)$.

Definition 4. (Valid Utility System) [12] A utility system is said to be valid if and only if the sum of private utilities of the agents is at most the social utility. That is, the utility system $(\gamma, \cup_i \phi_i)$ is a valid utility system if and only if it has the property $\sum_i \phi_i(A) \leq \gamma(A)$.

We want to prove that the game we defined in (3) and (4) is a valid utility system. At first, we prove it is a utility system.

Theorem 4. The game $(\gamma, \cup_i \phi_i)$ defined in (3) and (4) is a utility system.

Proof:

$$\begin{aligned} & \gamma'_{a_i}(A \oplus \emptyset_i) \\ = & \gamma(A) - \gamma(A^{-i} \oplus \emptyset_i) \\ = & \sum_{t \in a_i} w^t \left(\sum_{j \in \mathcal{N}_i | t \notin a_j} \tau_{ij}^t + \tau_{ii}^t \right) \\ \leq & \sum_{t \in a_i} w^t \left(\sum_{j \in \mathcal{N}_i | t \notin a_j} \tau_{ij}^t + \sum_{j \in \mathcal{N}_i | t \in a_j} \frac{\tau_{ij}^t}{2} + \tau_{ii}^t \right) \\ = & \phi_i(a_i). \quad \square \end{aligned}$$

Theorem 5. The utility system $(\gamma, \cup_i \phi_i)$ defined in (3) and (4) is valid.

Proof: We need to prove the utility system $(\gamma, \cup_i \phi_i)$ has the property $\sum_{i \in V} \phi_i(A) \leq \gamma(A)$.

First, we define the set of covered edges for application t as $E^t = \{(i,j) | i \in G^t \text{ or } j \in G^t\}$. Equation (3) shows that each $e = (i,j) \in E^t$ contributes τ_{ij}^t to $\gamma(A)$. We use a vector (ξ_i, ξ_j) to denote e 's contribution to $\phi_i(A)$ and $\phi_j(A)$. There are three cases here:

$$(\xi_i, \xi_j) = \begin{cases} (\tau_{ij}^t, 0), & \text{if } i \in G^t, j \notin G^t \\ (0, \tau_{ij}^t), & \text{if } i \notin G^t, j \in G^t \\ (\frac{1}{2}\tau_{ij}^t, \frac{1}{2}\tau_{ij}^t), & \text{if } i \in G^t, j \in G^t \end{cases}$$

Since e 's contribution to $\gamma(A)$ equals to the sum of its contribution to $\phi_i(A)$ and $\phi_j(A)$, after we sum up all $e \in E$,

$$\sum_{t \in P} w^t \sum_{(i,j) \in E, i \in G^t \text{ or } j \in G^t} \tau_{ij}^t = \sum_{i \in V} \left(\sum_{t \in a_i} w^t \sum_{j \in \mathcal{N}_i} \frac{\tau_{ij}^t}{n_e^t} \right).$$

Now we consider contribution of nodes, it is obviously that

$$\sum_{t \in P} w^t \sum_{i \in G^t} \tau_{ii}^t = \sum_{i \in V} \sum_{t \in a_i} w^t \tau_{ii}^t.$$

Combining both contribution of edges and nodes,

$$\gamma(A) = \sum_{i \in V} \phi_i(A). \quad \square$$

We cite below an important result on valid game [12].

Lemma 1. *Let γ be a non-decreasing, submodular set function. If $(\gamma, \cup_i \phi_i)$ is a valid utility system then for any pure strategy Nash equilibrium $A^* \in \mathcal{A}$, we have $\gamma(A^*) \geq \frac{1}{2}OPT$, where OPT is the optimal social utility.*

Combining Theorem 2, Theorem 3, Theorem 5 and Lemma 1, we get following theorem.

Theorem 6. *For the submodular game $(\gamma, \cup_i \phi_i)$ we defined in (3) and (4), there exists at least one pure strategy Nash equilibrium. And for its any pure strategy Nash equilibrium $A^* \in \mathcal{A}$, we have $\gamma(A^*) \geq \frac{1}{2}OPT$.*

Now we consider the case in which each sensor runs the approximation algorithm and can only get a $\frac{1}{\beta}$ approximate solution instead of optimal solution for the multidimensional knapsack problem. $\frac{1}{\beta}$ approximate solution ($\beta > 1$) means the solution is not less than $\frac{1}{\beta}$ of the optimal solution. We can prove that our algorithm can achieve a β -approximate Nash equilibrium.

Definition 5. (β -approximate Nash Equilibrium) *A pure strategy profile $A \in \mathcal{A}$ is a β -approximate Nash equilibrium if no agent can find a better alternative pure strategy in which its private utility is more than β times better than its current private utility. That is for any agent i ,*

$$\forall a'_i \in \mathcal{A}_i, \quad \phi_i(A \oplus a'_i) \leq (1 + \beta)\phi_i(A)$$

Theorem 7. *For the submodular game defined in (3) and (4), Submodular Game Algorithm (Algorithm 2) with the $\frac{1}{\beta}$ -approximation algorithm converges to a β -approximate Nash equilibrium $A \in \mathcal{A}$.*

Proof: The proof follows the same way of theorem 3. By bounding the value of the potential function, we can prove out Submodular Game Algorithm can reach a β -approximate Nash equilibrium. \square

We cite the following important result on approximate Nash equilibria [12].

Lemma 2. *Let γ be a non-decreasing, submodular set function, and $(\gamma, \cup_i \phi_i)$ be a valid utility system. In any β -approximate Nash equilibrium $A \in \mathcal{A}$ we have $\gamma(A) \geq \frac{1}{1+\beta}OPT$, where OPT is the optimal social utility.*

Theorem 8. *For the submodular game defined in (3) and (4), Submodular Game Algorithm (Algorithm 2) with a β -approximate solution converges to a β -approximate Nash equilibrium $A \in \mathcal{A}$, and we have $\gamma(A) \geq \frac{1}{1+\beta}OPT$, where OPT is the optimal social utility.*

Theorem 8 follows Theorem 5, Theorem 7 and Lemma 2.

In our implementation, we use a $\frac{1}{1+m}$ -approximation algorithm, so our Submodular Game Algorithm can converge to a $(1+m)$ -approximate Nash equilibrium with an $\frac{1}{2+m}$ approximation bound.

VI. EVALUATION

In this section, we evaluate our **Submodular Game Algorithm (SG)** by comparing it against a state-of-the-art centralized optimization algorithm **Fractional Relaxation Greedy (FRG)** [9]. We conduct simulations on three real world datasets:

Intel dataset is collected in Intel Berkley lab [17]. 54 Mica2Dot sensor nodes with weather boards were used to collect topology information along with humidity, temperature and light values. The data collection last for more than one month at a sampling period of 31 seconds. In our evaluation, we generate the covariance matrices using data collected from 20 nodes in one day.

DARPA dataset is collected in the DARPA SensIT vehicle detection experiments [18]. 75 WINS NG 2.0 nodes are deployed to detect vehicles driving through several intersecting roads in 29 Palms, CA. Each WINS NG 2.0 node is equipped with three sensing modalities: acoustic (microphone), seismic (geophone) and infrared (polarized IR sensor). All nodes are deployed in an area of approximately $900 \times 300m^2$. In our evaluation, we use acoustic and seismic readings from 23 nodes in the dataset to generate covariance matrices.

BWSN dataset is acquired by running simulations on a 129-node sensor network used in Battle of the Water Sensor Networks (BWSN) [19]. We use the "bwsn-utilities" [20] program to simulate 10000 random injection events to this network for a duration of 96 hours and use the generated event detection data to calculate the covariance matrices. We use two event injection strategies to build two sets of data as two applications.

For each dataset, we can calculate the covariance matrices based on the sensor readings. The Packet Reception Ratio (PRR) of each link is included in the Intel dataset. We generate PRR for DARPA and BWSN datasets based on location information of sensors following the way proposed in [21]. We then generate different network topologies by assigning different PRR bounds. Only links with PRR higher than the PRR bound is used for communication. In our simulations, we repeat Algorithm 2 in Section IV 10 times for each network topology. Because the number of applications is at most 3 in our simulations, we employ naive enumeration to solve the multidimensional knapsack problem (5) on each sensor node. As we proved in Theorem 6, SG will terminate at a pure strategy Nash equilibrium and the approximation bound is no less than $\frac{1}{2}$. We implemente our SG algorithm in Matlab. All results are gathered on a Macbook Pro machine with CPU frequency at 2.4GHz and 4GB memory.

Figure 1 analyzes the behavior of SG. Here we define *covariance cover ratio* as the ratio between covariance cover achieved by the algorithm and the maximum covariance cover in the network. Since searching an optimal solution is too computational expensive, to assess the tightness of our bound, we compare our solution with the maximum covariance cover, i.e., the sum of all edge weights and node weights in the network. Figure 1(a) shows that the covariance cover of our

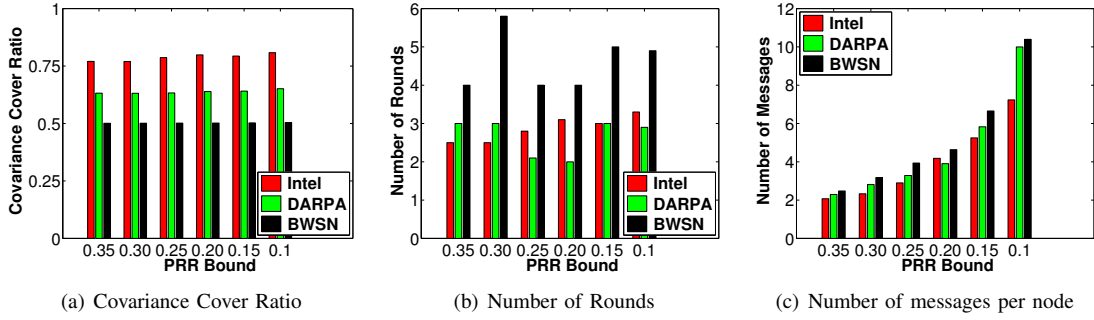


Fig. 1. Game Behavior Analysis

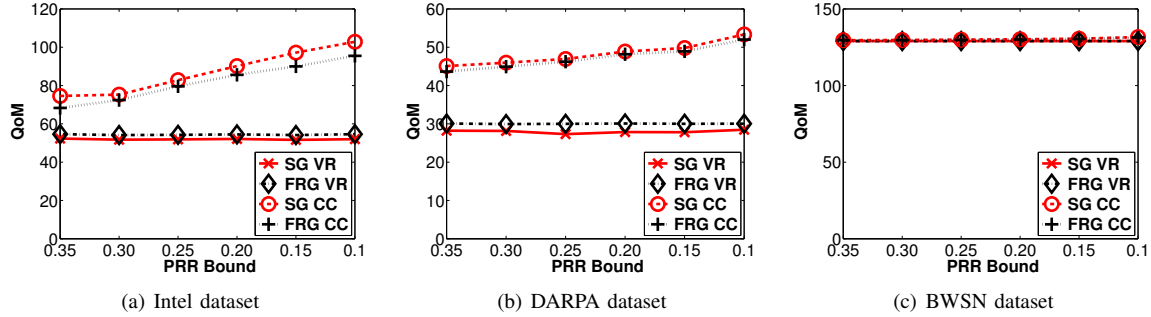


Fig. 2. QoM Performance Analysis

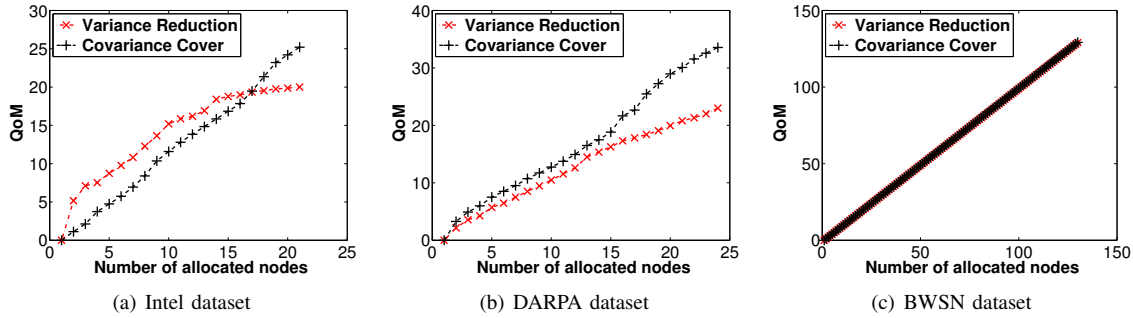


Fig. 3. Comparison between VR and CC

solution is consistently no less than half of the maximum covariance cover, which means our solution is no less than half of the optimal solution. Results in Figure 1(a) indicates the tightness of our $\frac{1}{2}$ bound.

Figure 1(b) shows the maximum number of rounds for SG to converge is below 6 across all cases in all three real-world datasets. The communication cost of our algorithm is evaluated in Figure 1(c). It shows the average number of messages sent per node. As each sensor node has more neighbors with a lower PRR threshold, the average number of messages sent by each node increases from 2 to 10. Our results show that the communication cost required by SG in terms of number of packets is moderate.

Figure 2 compares the performance of SG with FRG [9] in terms of variance reduction (VR) and covariance cover (CC). The variance reduction delivered by SG is always above 98% of that achieved by FRG in all three datasets. The covariance cover of SG is consistently higher than FRG. For BWSN

dataset, the difference between different methods is within 2, which makes four curves difficult to tell. This result indicates the decentralized approach employed by SG is competitive with the centralized solution in terms of QoM.

Figure 3 investigates the correlation between covariance cover and variance reduction. We increase the number of allocated sensor nodes ϱ under same PRR threshold 0.5. It is difficult to distinguish VR and CC in BWSN dataset, because the difference of them is within 1. Results in the other two datasets show variance reduction and covariance cover are very close when ϱ is less than $n/2$, where n is total number of sensors. This is because when ϱ is small, allocated nodes are not neighbors of each other, which coincides with our assumption in Theorem 1. The difference increases when ϱ exceeds $n/2$, but a higher covariance cover is always associated with a higher variance reduction. This result shows that covariance cover can be used as an effective proxy to optimize the variance reduction of a node allocation.

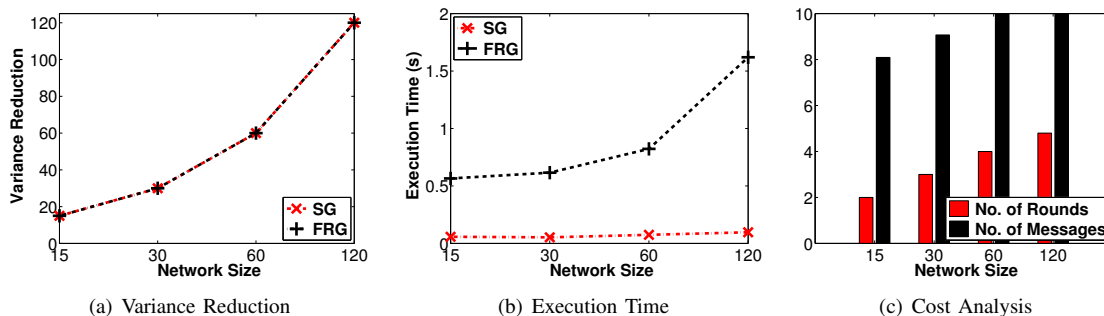


Fig. 4. Scalability Analysis

We evaluate the scalability of our algorithm by selecting different subsets of sensor nodes from the BWSN dataset. Figure 4(a) shows SG is highly competitive against FRG in terms of variance reduction. Note the difference between SG and FRG is consistently within 2, hence the SG and FRG curves are almost indistinguishable here. The execution times of SG and FRG for varying size of networks are compared in Figure 4(b). Since the SG is a distributed algorithm, we show the average execution time per node. For FRG, we show its overall execution time because it is a centralized algorithm. Our results show that SG remains fast as the number of nodes increases, with the run time remaining below 0.1 second. While the Macbook machine used in our simulation is more powerful than typical sensor nodes, the short execution times nevertheless indicates that SG is practical on sensors. More importantly, the solution scales effectively with network size. In comparison, the run time of FRG increases significantly as the number of nodes increases.

It is important to note that SG brings significant advantages than a centralized algorithm in several important ways. It does not incur the communication overhead for collecting the topology information of the entire network. Furthermore, it is robust against network disconnection as it does not depend on a single base station.

In Figure 4(c), we analyze the number of rounds and communication cost of SG. Both the number of rounds and messages per node increase moderately as the network size increases. The number of rounds remains within 10, indicating the scalability of our decentralized algorithm.

VII. CONCLUSIONS

This paper presents a distributed game-theoretic approach to application allocation in shared sensor networks. We first transform the optimal application allocation problem to a submodular game and then develop a decentralized algorithm that only employs localized interactions among neighboring nodes. We prove that the network can converge to pure strategy Nash equilibrium with a approximation bound. Simulations based on three real-world datasets demonstrate that our algorithm is competitive against a state-of-the-art centralized algorithm while scaling effectively with network size.

VIII. ACKNOWLEDGMENT

This work is supported by NSF grants CNS-1017701 (NeTS), CNS-1035773 (CPS), CNS-1144552 (NeTS) and Microsoft Research New Faculty Fellowship.

REFERENCES

- [1] "Citysense," <http://www.citysense.net/>.
- [2] <http://research.cens.ucla.edu/areas/2005/NIMS/>.
- [3] <http://www.memsc.com/products/wireless-sensor-networks/wireless-modules.html>.
- [4] F. Bian, D. Kempe, and R. Govindan, "Utility-based sensor selection," in *IPSN*, 2006.
- [5] C. Guestrin, A. Krause, and A. P. Singh, "Near-optimal sensor placements in gaussian processes," in *ICML*, 2005.
- [6] 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*, vol. 134, no. 6, pp. 516–526, 2008.
- [7] A. Krause, B. McMahan, C. Guestrin, and A. Gupta, "Robust submodular observation selection," *JMLR*, vol. 9, pp. 2761–2801, Dec 2008.
- [8] A. Kulik, H. Shachnai, and T. Tamir, "Maximizing submodular set functions subject to multiple linear constraints," in *SODA*, 2009.
- [9] Y. Xu, A. Saifullah, Y. Chen, C. Lu, and S. Bhattacharya, "Near optimal multi-application allocation in shared sensor networks," in *MobiHoc*, 2010.
- [10] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, Jan. 2009.
- [11] M. Rabbat and R. Nowak, "Distributed optimization in sensor networks," in *IPSN*, 2004.
- [12] A. Vetta, "Nash equilibria in competitive societies, with applications to facility location, traffic routing and auctions," in *FOCS*, 2002.
- [13] R. Johari and J. N. Tsitsiklis, "Efficiency loss in a network resource allocation game," *Journal Mathematics of Operations Research*, vol. 29, no. 3, pp. 407–435, 2004.
- [14] O. Ben-zwi and A. Ronen, "The local and global price of anarchy of graphical games," in *SAGT*, 2008.
- [15] S. Bhattacharya, A. Saifullah, C. Lu, and G. C. Roman, "Multi-application deployment in shared sensor networks based on quality of monitoring," in *RATS*, 2010.
- [16] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack problems*. Springer, 2004.
- [17] <http://db.csail.mit.edu/labdata/labdata.html>.
- [18] M. F. Duarte and Y. H. Hu, "Vehicle classification in distributed sensor networks," *Journal of Parallel and Distributed Computing*, vol. 64, no. 7, pp. 826–838, Jul. 2004.
- [19] A. Ostfeld and et al., "The Battle of the Water Sensor Networks (BWSN): A Design Challenge for Engineers and Algorithms," *Journal of Water Resources Planning and Management*, vol. 134, no. 6, pp. 556–568, 2008.
- [20] <http://www.water-simulation.com/wsp/about/bwsn/>.
- [21] M. Zuniga and B. Krishnamachari, "Analyzing the transitional region in low power wireless links," in *SECON*, 2004.