Cooperative Allocation and Scheduling of Tasks for Minimizing Interval Data Sampling in WSNs

Yawei Zhao, National University of Defense Technology
Deke Guo, National University of Defense Technology
Jia Xu, Guangxi University
Pin Lv, Guangxi University
Jianping Yin, National University of Defense Technology

Data sharing amongst multiple tasks is significant to reduce energy consumption and communication cost in low-power wireless sensor networks. Current proposals have already scheduled the discrete point sampling tasks to decrease the amount of sampled data. However, less effort has been done for the applications requiring continuous interval sampling data. Additionally, most of prior work focuses on scheduling tasks on a single sensor node. In this paper, we broaden the scope to the entire network and attempt to minimize the total amount of interval sampling data instead of traditional discrete point sampling data. Our basic idea is to cooperatively allocate and schedule multiple sampling tasks by utilizing the overlap amongst these tasks. Since the allocation problem is NP-hard and the scheduling problem is NP-complete, we design two dedicated 2-factor approximation methods for task allocation and scheduling, respectively. Extensive real experiments are conducted on a testbed of 50 sensor nodes to evaluate the effectiveness of our proposals. Moreover, the scalability of our proposals is verified by using the widely-used simulation tool, i.e., TOSSIM. The experimental results indicate that, compared with the state-of-the-art solution, our methods reduce the amount of sampled data significantly.

CCS Concepts: •**Networks** → **Sensor networks;** Cyber-physical networks; •**Information systems** → *Spatial-temporal systems;*

Additional Key Words and Phrases: Data sharing, interval sampling tasks, data aggregation, coverage, WSNs

1. INTRODUCTION

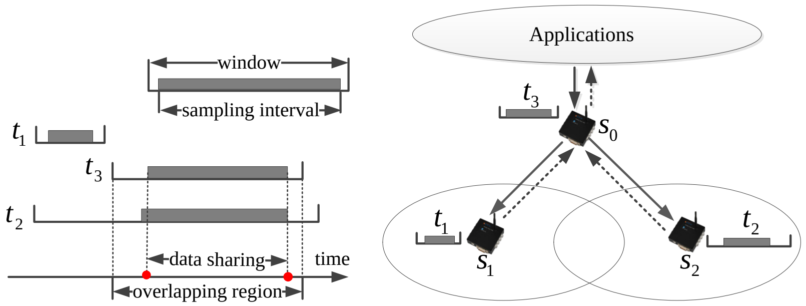
Many successful applications of low-power wireless sensor networks (WSNs) under different scenarios, e.g., earthquake monitoring [Cerullo et al. 2005], railway diagnosis [Suzuki et al. 2007], wild-life protection [Szewczyk et al. 2004], and structure monitoring [Xu et al. 2004], are time-consuming and labor-intensive. Besides, they are all constrained by multiple scarce resources such as power, memory and computational resources. The experience from some real projects [Mo et al. 2009; Jiang et al. 2009] shows that the lifetime of an initially deployed WSN is restricted to only a few months without recharging. Therefore, some resource-sharing strategies are proposed for different applications to improve the efficiency of the WSNs.

Data sharing amongst different applications is such a resource-sharing strategy. It has caught much attention especially for the continuous interval sampling data. Such interval sampling data is derived by performing data sampling for a time interval. For

This work is supported by the National Natural Science Foundation of China (NSFC), under Grant 61170287, Grant 61232016, Grant 61402494, Grant 61402513, and Grant 61202487.

Author's addresses: Y. Zhao and J. Yin, College of Computer, National University of Defense Technology; D. Guo, Science and Technology on Information System Engineering Laboratory, National University of Defense Technology; J. Xu and P. Lv, School of Computer, Electronics and Information, Guangxi University.

DOI: http://dx.doi.org/10.1145/0000000.0000000



(a) Task scheduling (b) Task allocation

Fig. 1. The illustrative examples for the scheduling of *sampling interval* in the left panel and the allocation of tasks in the right panel.

example, acoustic data are continuously collected for a time interval to diagnose the state of the railway system while the discrete data does not help much [Cerullo et al. 2005]. As another example, the vibration in the volcano, earthquake [Suzuki et al. 2007] and structure monitoring systems [Xu et al. 2004] need to be continuously sampled for a period of time to analyze the anomaly while the exceptional data may be lost if the discrete sampling is employed. A further example is that continuously observed data for a time interval can be utilized to conduct scientific analysis on animals' behaviors [Szewczyk et al. 2004]. Such an interval sampling task, as shown in Fig. 1, has two properties, namely, a *time window* and a *sampling interval*. The *sampling interval* can be flexibly adjusted in the *time window*, which poses great opportunity to implement data sharing amongst multiple sampling tasks, e.g., the data sharing between tasks t_2 and t_3 in Fig. 1(a).

Recent literature has made some contributions on maximizing the data sharing amongst interval sampling tasks [Fang et al. 2013]. Fang et al. propose an effective task scheduling method for a task set which runs on a single node [Fang et al. 2013]. However, the scheduling strategy designed for a single node cannot be immediately extended to entire WSNs. It is thought to be invalid for entire WSNs due to an improper plan of task allocation amongst multiple nodes. Generally, actual deployed WSNs are k-coverage and r-redundant networks (Section 3.1) which make a balance between the reliability of entire network and the quality of communication [Huang and Tseng 2003], [Li and Liu 2007], [Mao et al. 2013]. The proposed data sharing strategies in [Fang et al. 2013] are not suitable to such a situation where data sharing amongst multiple sampling tasks becomes extremely complicated. The reason is that the methods proposed in [Fang et al. 2013] only consider how to maximize data sharing after sampling tasks have been allocated to sensor nodes, which ignore the potential gains of data sharing during task allocation. Since the strategy of task allocation has a great impact on the possible data sharing during task scheduling, only can the optimization of task scheduling be not enough. Instead, the allocation and scheduling of tasks should be taken into consideration together in order to maximize the data sharing for entire WSNs. For example, as illustrated in Fig. 1(b), the sink node s_0 accepts sampling tasks t_1, t_2 and t_3 from applications and then allocates them to other nodes, namely, s_1 and s_2. When s_1 and s_2 complete the tasks, all the sampled data will be sent to the sink node s_0 and then consumed by the applications. If the tasks t_1 and t_2 have been allocated to the nodes s_1 and s_2, respectively, then the task t_3, as illustrated in Fig. 1(a), should be allocated to s_2 due to more data sharing, that is, the overlapping time region between t_2 and t_3. The scheduling method proposed by [Fang et al. 2013], however, is not an effective solution for the entire network due to ignoring the overlap time region between tasks.

In a practical system, each node is usually assigned with more than one task during its life cycle. The conventional sampling strategy, i.e., First-In-First-Service (FIFS), does not help much to use data sharing amongst multiple sampling tasks, due to ignoring the overlap among sampling tasks. In fact, as illustrated in Fig. 1(a), the sampling intervals can be adjusted in the time windows to increase the overlap and lead to more data sharing. To be specific, it is the adjusting these intervals to achieve the minimal amount of sampled data that faces two challenging issues as follows.

— **Task allocation:** As demonstrated in Fig. 1(b), three sensor nodes s_0, s_1 and s_2 in WSN are considered. Here, s_0 is the sink node which is responsible for allocating sampling tasks to s_1 and s_2. A data-intensive application injects three sampling tasks t_1, t_2 and t_3 into the network, as shown in Fig. 1(a). If t_1 and t_2 have been allocated to s_1 and s_2, respectively, t_3 should be allocated to s_2. The reason is that the time window of t_3 and that of t_2 are overlapped. Therefore, sampling interval of t_3 and that of t_2 can be adjusted to achieve to maximal data sharing. The problem of task allocation is NP complete (See Theorem 3.7 in Section 3.2). Consider that extensive data-intensive applications generate a large set of sampling tasks. It is extremely difficult to allocate these tasks in order to maximize data sharing with a resource-constrained sensor node.

— **Task scheduling:** As illustrated in Fig. 1(a), three sampling tasks t_1, t_2 and t_3 which have been allocated to a sensor node. Since the time window of t_2 and t_3 have the maximal overlapping time region. Sampling intervals of t_2 and t_3 can be adjusted to achieve the maximal data sharing when data is sampled in the overlapping time region. The problem of scheduling sampling interval is NP complete (See Lemma 3.6 in Section 3.2). Consider that a sensor node may be allocated numerous sampling tasks in a monitoring system. It is very difficult to solve the scheduling problem in polynomial time with a resource-constrained sensor node.

In this paper, we aim at maximizing of the interval sampling data bearing the view of the entire WSN, rather than a single sensor node. The optimization problem is non-convex, and it is impossible to find an optimal solution for the resource-limited sensor nodes (Section 3.2). Therefore, we provide an effective solution by two steps, i.e., task scheduling and task allocation. Our basic idea, according to Fig. 1, is to allocate and schedule a task set by utilizing the overlap amongst multiple sampling tasks. To address the above issues, we first formulate the optimal problem in a k-coverage and r-redundant WSN. We then propose a 2-factor approximation scheduling method which aims to minimize the amount of sampled data for a sampling task set. Furthermore, a 2-factor approximation task allocation method is proposed to reduce the amount of unnecessary sampled data when tasks are allocated. To evaluate the correctness and the performance of our methods, we first conduct real implementations on a testbed containing 5×10 arrays of wireless sensor nodes. Simulations are further conducted in the widely-used simulation tool, i.e., TOSSIM. The evaluation results indicate that our methods significantly reduce the amount of sampled data and energy consumption and therefore improve the bandwidth and reliability of wireless communication in WSNs.

The rest of this paper is organized as follows. Section 2 summarizes related work. Section 3 defines the basic models of the network and sampling task, and then formalizes the optimization problem. Section 4 proposes a 2-factor approximation task scheduling method for a sampling task set. Section 5 focuses on the task allocation problem and proposes a 2-factor approximation allocation method. Section 6 evaluates the correctness and performance of our proposals with extensive experiments. Finally, Section 7 further discusses our solution and Section 8 concludes this paper.

2. RELATED WORK

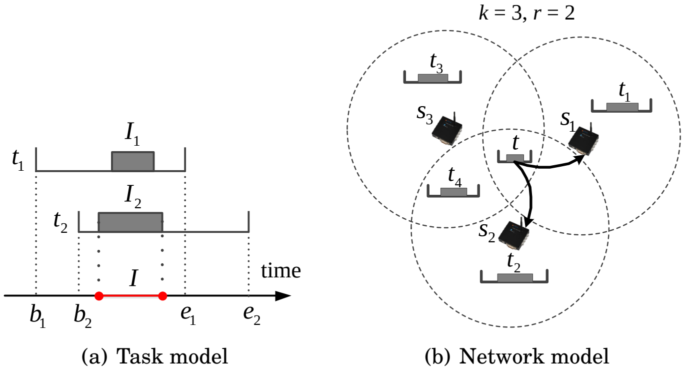
2.1. Energy-aware task allocation and scheduling mechanisms in WSNs

Xu et al. propose an energy-balanced method of task allocation which implements the maximal energy dissipation amongst all sensor nodes [Xu et al. 2010]. The tasks are executed during the beginning of each epoch and must be completed before the end of the epoch. That is, the epoch of a task is similar to the time window of the task model in the paper. It is noted that tasks in [Xu et al. 2010] is communication tasks, not the sampling tasks. The difference results in that the proposed method in [Xu et al. 2010] does not suit to our problem. Additionally, the solution for solving an integer linear programming problem in [Xu et al. 2010] costs much time. Packets in [Song et al. 2006] own the same properties of a sampling task we discuss in the paper. The release time and the deadline of a packet represent the beginning time and the end time, respectively. The difference is that a packet reports one unit of data, rather than the data which is sampled over a time interval continuously. Thus, the methods in [Song et al. 2006] work well for the discrete point sampling tasks, not the interval sampling tasks.

The most related work to ours are [Tavakoli et al. 2010] and [Fang et al. 2013]. Tavakoli et al. present an approach for task scheduling on a sensor node to minimize the network communication overhead [Tavakoli et al. 2010]. A task in [Tavakoli et al. 2010] is a discrete point sampling task and requires to be performed via once data sampling during its time window. Therefore, the method proposed in [Tavakoli et al. 2010] does not suit to our problem. Fang et al. propose an effective sampling approach for interval sampling tasks on a single sensor node [Fang et al. 2013]. The 2-factor approximation algorithm in [Fang et al. 2013] is the state-of-the-art method to maximize the data sharing amongst tasks on a single node. Unfortunately, there are two weak points in [Fang et al. 2013]. First, the proposed scheduling method schedules sampling tasks in the ascending order of the end time of tasks. As a result, such scheduling strategy neglects data sharing between overlapping tasks. Moreover, Fang et al. assume that all tasks have the same length of sampling interval, which is too ideal and not practical. By contrast, we observe that multiple tasks may be overlapping, and thereby data sharing exists. Our solution exploits this feature and thus achieves better performance on maximizing the data sharing than the scheduling method in [Fang et al. 2013]. Second, the solution in [Fang et al. 2013] only focuses on task scheduling on a single sensor node, and does not consider the process of task allocation in a WSN. As we have discussed, the performance of algorithms of task scheduling is sensitive to the strategy of task allocation. Only does optimizing the process of the former not achieve the final optimization for a deployed system. Our solution is more general and practical because of jointly optimizing the process of task allocation and that of scheduling sampling interval.

2.2. Multi-query optimization in WSNs

Recently, a WSN is treated as a database providing a good logical abstraction for sensor data management. The aim of multi-query optimization in such a database system studies how to efficiently process queries [Xiang et al. 2007], [Trigoni et al. 2005]. Xiang et al. adopt a two-tier multiple query optimization scheme to minimize the average transmission time in WSNs [Xiang et al. 2007]. The first-tier optimization is a cost-based approach which schedules queries as a whole and eliminates duplicate data requests from original queries. Since it is not the optimization of the volume of sampled data, such an approach cannot be used for our problem. Moreover, the second-tier optimization acquires and transmits sampled data by using the broadcast nature of the radio channel. Our solution aims to provide a general solution for maximizing



(a) Task model (b) Network model

Fig. 2. The task and network models for the interval data sampling.

data sharing amongst sampling tasks. The details of wireless communication is not involved in our method. Trigoni et al. consider multi-query optimization by using aggregation operations such as *sum* and *avg* to achieve the optimal communication cost [Trigoni et al. 2005]; while our method mainly concerns minimal energy consumption by reducing redundant sampled data.

2.3. Compression techniques based data collection

Compression-based data collection significantly reduces redundancy of sampled data for a sensor node [Arici et al. 2003; Pradhan et al. 2002]. Arici et al. propose an in-network compression scheme (PINCO) for a densely deployed WSN. PINCO compresses raw data by reducing redundancy existing in sensor readings in spatial, temporal and spatial-temporal domains [Arici et al. 2003]. Unfortunately, PINCO trades higher latency for lower energy consumption due to the process of data compression. Such weakness limits its effectiveness in some latency aware applications. By contrast, our solution does not cost time to analyze and compress the raw data. Moreover, PINCO only considers the single-valued data (humidity, temperature, etc.). Since single-valued data is produced by the discrete point sampling tasks, methods in [Arici et al. 2003] cannot be used to solve our problem directly. Using fast error-correcting coding algorithms, Pradban et al. present a framework on the distributed souring coding technique to reduce data redundancy [Pradhan et al. 2002]. However, method in [Pradhan et al. 2002] requires to get known of the sensor correlation structure. It is impossible for a randomly deployed system such as a battlefield monitoring system.

3. PRELIMINARIES

Before we further discuss the optimization problem of interval sampling for a task set in WSNs, the basic task and the network models used in this paper are formalized. Based on these models, we first define the optimization problem for a task set in WSNs by using the idea of interval data sharing. We then define a metric for measuring the data sharing amongst sampling tasks, which will be used to design the effective allocation and scheduling methods for a task set.

3.1. Task and network models

The task model and the WSN model are formalized by Definition 3.1 and Definition 3.2, respectively.

Definition 3.1 (*Interval sampling task*). Given a task t, it can be defined as a triple $<b, e, l>$ where b, e and l represents the begin time, the end time and the interval length of the task, respectively. Its time window is denoted by $[b,\ e]$, and the sampling interval can flexibly move in the time window.

In the view of a single sensor node, when a task is overlapping with other tasks, they can be scheduled to share the sampled data in the overlapping region to reduce the amount of sampled data. As illustrated in Fig. 2(a), there are two overlapping tasks t_1=$<b_1, e_1, l_1>$ and t_2=$<b_2, e_2, l_2>$. Here, b_i, e_i and l_i are the begin time, the end time and the corresponding sampling interval of the task t_i (i may be valued 1 or 2). The sampling interval I satisfies the requirements of two tasks, but its interval length is smaller than the sum of intervals I_1 and I_2 due to the utilization of data sharing between t_1 and t_2. For a single sensor node, the best scheduling solution is to minimize the amount of sampled data, decreases energy consumption, and prolongs its lifetime. However, the scheduling problem is NP complete (Lemma 3.6 in Section 3.2). We first discuss the scheduling problem and give a 2-factor approximation scheduling algorithm in Section 4.

Definition 3.2 (*k-coverage and r-redundant network*). Given a wireless sensor network N, it can be represented as $N= (S, T, k, r)$. Here, S and T represent the node set and the task set of the WSN, respectively. The notation k means a task in the network can be detected by k sensor nodes, but only r out of them are identified to execute the task. k is determined when a wireless sensor network is deployed. r is set by the requirement of communication quality or applications, and is universal to all the sampling tasks.

As illustrated in Fig. 2(b), the wireless sensor network consists of three sensor node s_1, s_2, and s_3. Each of them has a sensing range (indicated by the dotted circle). The tasks t_1, t_2, t_3 and t_4 have been allocated to a sensor node if which can be detected by the node. The task t is detected by three sensor nodes in WSNs, i.e., k=3. In fact, to guarantee the requirement of the communication quality and that of applications, t is eventually assigned to two sensor nodes, i.e., r=2. Thus, the allocation problem exists and will be discussed in Section 5.

We aim to minimize the amount of sampled data for the sampling tasks bearing the view of WSNs. The optimization problem contains two subproblems: allocating tasks amongst multiple candidate sensor nodes, and scheduling tasks for a sampling task set. Even though the optimization problem is extremely difficult, it is valuable and meaningful. A good allocation solution will reduce much unnecessary sampled data, which benefits the quality of communication so as to relieve delay and congestion of the entire network.

Table I. Symbols list

Symbol	Meaning	Symbol	Meaning
t	a task	n	the cardinality of T
T	a task set	m	the cardinality of S
s	a sensor node	b	the begin time
S	a sensor node set	e	the end time
I	an interval	$\lvert \cdot \rvert$	cardinality of a set
Φ	an interval set	x_{ij}	an indicator variable
l, $\lvert I \rvert$	the interval length	$d(t_i, t_j)$	value of data sharing
δ	the same length of interval used in a compact model		
φ	the number of compact tasks in a compact model		
t_{ij}	a new task generated by combining t_i and t_j		
k	a task can be executed by k sensor nodes		
r	a task is actually performed by r sensor nodes		

3.2. Interval data sharing based minimization of the amount of sampled data

We define the optimization problem in this section. To ease the description, Table I summarizes the frequently used symbols throughout the paper.

Definition 3.3. Given an interval I, and a $0-1$ indicator variable x_{ij}. If the task t_i is allocated to the node s_j, then x_{ij}=1 else x_{ij}=0. We define $x_{ij}\odot I$ as follows:

$$x_{ij}\odot I=\begin{cases} I & : \quad x_{ij}=1 \\ \emptyset & : \quad x_{ij}=0 \end{cases} \tag{1}$$

Definition 3.4. There are two intervals I_1 and I_2 with $I_1=[u_1,v_1]$ and $I_2=[u_2,v_2]$. As illustrated in Fig. 2(a), I_1 and I_2 are overlapping when they have the common area on the time axis. Assume $u=\min\{u_1,u_2\}$ and $v=\max\{v_1,v_2\}$, if I_1 and I_2 are overlapping, we define a new interval I as the union of I_1 and I_2, i.e., $I=I_1\uplus I_2=[u,v]$ and $|I|=v-u$. Here, '$\uplus$' stands for the union operation of two intervals.

Definition 3.5. Given a task set T with $|T|=n$, and a sensor node set S with $|S|=m$, each task t_i, $t_i\in T$, is notated as a triple $<b_i,e_i,l_i>$. Since r out of k candidate sensor nodes are identified to perform the task t_i, the optimization problem is to find an allocation solution for T so as to:

$$\min\sum_{j=1}^{m}|\biguplus_{i=1}^{n} x_{ij}\odot I_i| \tag{2}$$

$s.t.$:

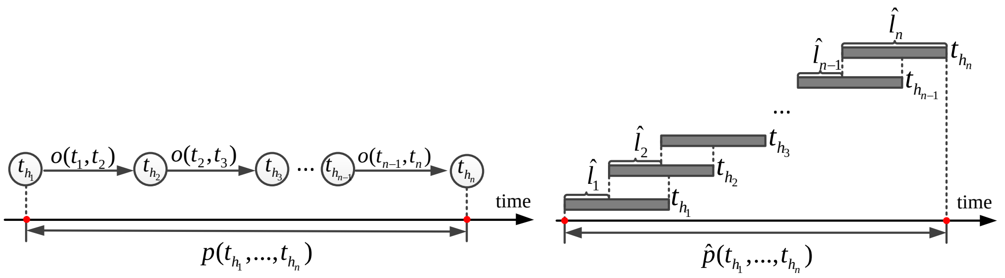
$$\begin{cases} \sum_{j=1}^{m} x_{ij}=r, i=1,...,n, 1\leq r\leq k; \\ x_{ij}=0 \ \text{ or } \ 1; \\ I_i\subseteq[b_i,e_i], i=1,...,n; \end{cases} \tag{3}$$

When $r=k$, we should allocate each task into all candidate sensor nodes. The allocation solution is determined solely. Generally, when $0<r<k$, the object function is non-linear, which makes the optimization problem become very difficult. This kind of non-linear programming problem has no universal efficient solution. Several methods, including branch and bound techniques, require high computational complexity and are not applied to a wireless sensor node. Considering the limitation of computing and memory resources of sensor nodes, the non-linear programming problem becomes very hard to resolve. To be clarified clearly, the scheduling problem, minimizing the amount of sampled data for a task set, is marked as "MIN-SA", and the allocation problem, minimizing the sum of sampled data for sampling tasks on multiple sensor nodes in WSNs, is marked as "MIN-SSA".

LEMMA 3.6. *The optimization problem of task scheduling, MIN-SA, is NP complete.*

PROOF. Assume there is a sampling interval set Φ satisfying the requirements of tasks in T. Intervals in Φ are sorted in the ascending order of the begin time and $i=1,2,...,|\Phi|$.

First, if Φ is the optimization solution, intervals in Φ must not be overlapping. Then, we check each interval whether it can be removed and Φ still satisfies the left tasks. If any interval in Φ can not be removed, the solution Φ can be proved to be optimal. The process of verifying can be completed in polynomial time. Hence the computation of the volume of sampled data is an NP problem.



(a) MAX-DHP in graph G.

(b) MAX-DHP is reduced to the scheduling problem MIN-SA.

Fig. 3. MAX-DHP can be reduced to the optimization problem of task scheduling.

Second, we construct a directional Hamilton graph which is used to reduce the Maximum Directed Hamiltonian Path (MAX-DHP) problem to the optimization problem of task scheduling. The construction procedure contains two steps:

- **Step** 1**:** For any pair of tasks $t_1{=}{<}b_1, e_1, l_1{>}$ and $t_2{=}{<}b_2, e_2, l_2{>}$, if they are overlapping and $b_1{<}b_2$, then a directed edge, pointing to t_2, is constructed. The weight of the edge equals the value of overlap, i.e., $d(t_1, t_2)$. If they are not overlapping, then the weight of edge is 0. It is obvious that the interval length l_i for t_i can be described as $l_i{=}d(t_i, \cdot){+}\widehat{l_i}$, where $d(t_i, \cdot)$ stands for the overlap between t_i and other tasks. As illustrated in Fig. 3(a), iteratively repeat the step until the directional graph G is constructed.
- **Step** 2**:** Given any Hamilton path t_{h_1}, t_{h_2}, ..., t_{h_n} in the graph G, the maximal length of a directed Hamilton path $p(t_{h_1}, ..., t_{h_n})$ in Fig. 3(a) is calculated as follows:

$$\begin{aligned}
&p(t_{h_1}, ..., t_{h_n})\\
&= \textstyle\sum_{i=1}^{n-1} d(t_{h_i}, t_{h_{i+1}})\\
&= p(t_{h_1}, ..., t_{h_{n-1}}) + (l_{t_{h_n}} - \widehat{l}_{t_{h_n}})\\
&= \cdots\\
&= \textstyle\sum_{i=1}^{n} l_{t_{h_i}} - \sum_{i=1}^{n} \widehat{l}_{t_{h_i}}\\
&= \textstyle\sum_{i=1}^{n} l_{t_{h_i}} - \widehat{p}(t_{h_1}, ..., t_{h_n}).
\end{aligned} \tag{4}$$

As illustrated in Fig. 3(b), $\widehat{p}(t_{h_1}, ..., t_{h_n})$ is the total length of intervals for the task set. Hence the MAX-DHP problem can be reduced to the optimization problem of task scheduling, MIN-SA.

As the entire process is completed in polynomial time, Lemma 3.6 is proved. □

THEOREM 3.7. *When $0{<}r{<}k$, the optimization problem of task allocation, MIN-SSA, is NP hard.*

PROOF. In a k-coverage and r-redundant sensor network, r out of k candidate nodes are used to execute a sampling task. Consider a special case, i.e., $r{=}1$, and $k{=}2$. In this situation, the optimization problem of task scheduling, MIN-SA, can be reduced to the optimization problem of task allocation, MIN-SSA.

Given a task set T with $T{\neq}\emptyset$, if there is an optimal solution to compute amount of the sampled data and returns an interval set Φ, it is easy to divide Φ into two subsets Φ_1, and Φ_2 such that each of which satisfies a task subset T_1 and T_2 $(T{=}T_1 \cup T_2)$, respectively. The procedure of partition can be finished in polynomial time. However, the computation of the minimal amount of sampled data, as proved in Lemma 3.6, is NP complete. The special case of allocation problem is NP hard. In general, the allocation problem is at least difficult as the special case. When $0<r<k$, the allocation problem is thus NP hard. □

3.3. Measurement of data sharing amongst tasks

Data sharing amongst tasks is the foundation of our scheduling method. We aim at minimizing the total length of sampling intervals via the data sharing amongst overlapping tasks. Before presenting our methods, we give out some definitions.

Definition 3.8 (*Satisfy*). Consider two overlapping tasks t_i=$<b_i, e_i, l_i>$ and t_j=$<b_j, e_j, l_j>$. We define two variables b and e such that b=$\max\{b_i, b_j\}$ and e=$\min\{e_i, e_j\}$. Given l_i^*=$\min\{l_i, e{-}b\}$ and l_j^*=$\min\{l_j, e{-}b\}$, t_i satisfies t_j if and only if $l_i^*{\geq}l_j$, and t_j satisfies t_i if and only if $l_j^*{\geq}l_i$.

Definition 3.9 (*Overlap*). Let $d(t_i, t_j)$ denote the maximal overlap value of two tasks t_i and t_j. Without loss of generality, assume $e_i{\leq}e_j$, then:

$$d(t_i,t_j)=\begin{cases} e_i - b_j & : \ t_i, t_j \text{ can not satisfy each other.} \\ l_j & : \ t_i \text{ satisfies } t_j. \\ l_i & : \ t_j \text{ satisfies } t_i. \end{cases} \quad (5)$$

Definition 3.10. Consider overlapping tasks t_i=$<b_i, e_i, l_i>$ and t_j=$<b_j, e_j, l_j>$. Sort those tasks in the descending order of end time. Without loss of generality, assume $e_i{\leq}e_j$. Then a time window $[b^*,\ e^*]$ can be constructed as follows:

— If t_i and t_j cannot satisfy each other, then:

$$\begin{cases} b^*{=}e_i{-}l_i \\ e^*{=}b_j{+}l_j \end{cases} \quad (6)$$

— If t_i satisfies t_j, then:

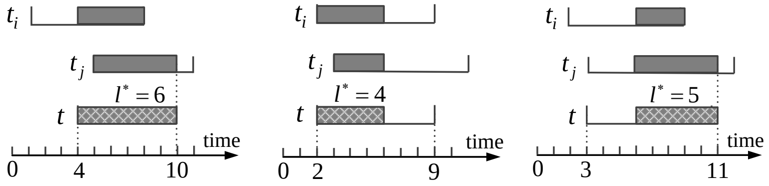
$$\begin{cases} b^*{=}\max\{b_i, b{-}(l_i{-}l_j)\} & b{=}\max\{b_i, b_j\} \\ e^*{=}\min\{e_i, e{+}(l_i{-}l_j)\} & e{=}\min\{e_i, e_j\} \end{cases} \quad (7)$$

— If t_j satisfies t_i, then:

$$\begin{cases} b^*{=}\max\{b_j, b{-}(l_j{-}l_i)\} & b{=}\max\{b_i, b_j\} \\ e^*{=}\min\{e_j, e{+}(l_j{-}l_i)\} & e{=}\min\{e_i, e_j\} \end{cases} \quad (8)$$

Definition 3.11 (*Combination*). Let t_{ij} denote the combination of two overlapping tasks t_i and t_j. The resultant task is denoted as t^* which is called the child task, while t_i and t_j are called father tasks. It is clear that if t^*=$<b^*, e^*, l^*>$, then l^*=l_i+l_j−$d(t_i, t_j)$, where both b and e are computed according to Definition 3.8.

As illustrated in Fig. 4(a), two tasks t_i=$<1, 8, 4>$ and t_j=$<5, 11, 5>$ do not satisfy with each other. Hence, it holds that $d(t_i, t_j)$=3, b^*=4 and e^*=10. After combination of them, a new task t_{ij} is generated and t=$<4, 10, 6>$. In Fig. 4(b), two tasks t_i=$<2, 9, 4>$ and t_j=$<3, 11, 3>$ are overlapping and t_i satisfies t_j. After combination of them, we get a new task t=$<2, 9, 4>$ where $d(t_i, t_j)$=3 and $l^* = 4$. Fig. 4(c) shows two overlapping tasks t_i=$<2, 9, 3>$ and t_j=$<3, 12, 5>$, and t_j satisfies t_i. After combination of them, we get a new task t_{ij} and t=$<3, 11, 5>$ where $d(t_i, t_j)$=3 and l^*=5. In the paper, we regard the value of data sharing of sampling tasks as a metric to measure data sharing.



(a) t_i and t_j do not satisfy with each other.

(b) t_i satisfies t_j.

(c) t_j satisfies t_i.

Fig. 4. Three cases: the combination of overlapping tasks.

```
ALGORITHM 1: Combine
Input: A task set $T$ and $T \neq \emptyset$.
Output: A task set where there are no overlapping tasks.
sort tasks in $T$ in the descending order of end time;
$Q \leftarrow$ Pre_processing ($T$);
sort quads in $Q$ in the descending order of value of data sharing;
repeat
    $t_i = Q(0).t_i$, $t_j = Q(0).t_j$, $t_{ij} = Q(0).t_{ij}$;
    insert $t_{ij}$ into $T$ in the ascending order of end time;
    remove $t_i$ and $t_j$ from $T$ and quads which include $t_i$ or $t_j$ from $Q$;
until $|T| > 1$ and $T$ contains overlapping tasks;
return $T$;
Function: Pre_processing ($T$)
repeat
    repeat
        if $t_i$ is overlapping with $t_j$ then
            a quad $q$ is generated with $q = <t_i, t_j, t_{ij}, d(t_i, t_j)>$;
            $Q \leftarrow Q \bigcup \{q\}$;
        end
    until each task $t_j \in T - \{t_i\}$;
until each task $t_i$ in $T$;
return $Q$;
```

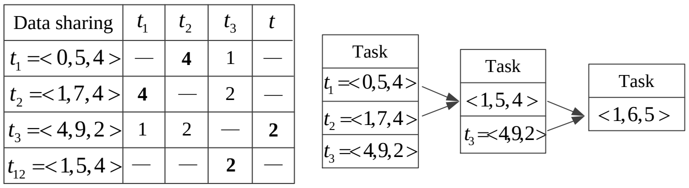
4. A 2-FACTOR APPROXIMATION SCHEDULING ALGORITHM DESIGNING FOR A SAMPLING TASK SET

In this section, we first propose a task scheduling algorithm for a task set via the data sharing amongst the overlapping tasks, and then prove it 2-factor approximation by the theoretical analysis.

Let T denote a non-empty task set at a sensor node. If $|T|=1$, the amount of sampled data is determined solely. If $|T|>1$, a quad $q = <t_i, t_j, t, d(t_i, t_j)>$ is maintained for the combination of the overlapping tasks, i.e., t_i and t_j. Here, t_i and t_j are father tasks, t is the combination result of t_i and t_j, and $d(t_i, t_j)$ is the maximal overlap value.

Our basic idea is to schedule all tasks in T by iteratively combining the task pair which has the maximal overlap value until there are no overlapping tasks in T. The whole scheduling procedure includes three major steps. First, it computes the combination of all overlapping tasks and gets a quad list Q. Second, the quad q, $q \in Q$, whose $d(t_i, t_j)$ is maximal is found. The original tasks t_i and t_j are replaced by the new task t in the task set T. Third, repeat above steps until no tasks are overlapping. Algorithm 1 presents the details and returns a task set which has no overlapping sampling tasks.

Algorithm 1 performs an iterative operation. Lines $1-3$ initialize a quad list Q, and sort the quad list in the descending order of $d(t_i, t_j)$. It consumes $O(n^2)$ memory to



Data sharing	t_1	t_2	t_3	t
t_1 =< 0,5,4 >	—	**4**	1	—
t_2 =<1,7,4 >	**4**	—	2	—
t_3 =< 4,9,2 >	1	2	—	**2**
t_{12} =<1,5,4 >	—	—	**2**	—

(a) Value of data sharing between overlapping tasks.

(b) Combination of tasks in the descending order of value of data sharing.

Fig. 5. The process of combination for overlapping tasks.

```
ALGORITHM 2: Combine_2
Input: A task set $T$ and $T \neq \emptyset$.
Output: A task set where there are no overlapping tasks.
sort tasks in $T$ in the ascending order of end time;
repeat
    find a task pair $<t_i, t_j>$ with the maximal overlap;
    insert the resultant task, i.e., $t = t_{ij}$ into $T$ in the ascending order of end time;
    remove $t_i$ and $t_j$ from $T$;
until $|T| > 1$ and $T$ contains overlapping tasks;
return $T$;
```

maintain the quad list. The time complexity is $O(n^2)$ due to the computation of maximal value of data sharing. Lines $4-7$ find a task pair which has the maximal value of data sharing in the task set T, and then update T and Q. As shown in Fig. 5, three tasks are notated by t_1=$<0, 5, 4>$, t_2=$<1, 7, 4>$ and t_3=$<4, 9, 2>$. First, we compute the data sharing of overlapping tasks and find that t_1 and t_2 have the maximal value of data sharing $d(t_1, t_2)$=4. Then, t_1 and t_2 are selected to construct a new task t_{12}=$<1, 5, 4>$. Second, remove t_1 and t_2 from T and add t_{12} into T. We get a new task, i.e., $<1, 6, 5>$ after combining t_{12} and t_3. Hence, the final sampling interval is $[1, 6]$, and the amount of sampled data is 5.

Note that both the space complexity and time complexity of Algorithm 1 are $O(n^2)$. Consider that the memory resource is very rare for a wireless sensor node. Algorithm 1 costs $O(n^2)$ space complexity because of maintaining a quad list. To address this problem, we further propose Algorithm 2 whose space complexity is $O(n)$ and time complexity is $O(n^2)$. It exhibits the same performance with Algorithm 1, but significantly reduces the memory consumption. Algorithm 1 is proved to be a 2-factor approximation task scheduling algorithm by Theorem 4.5. Before presenting the Theorem 4.5, we first present Lemma 4.1 and Property 4.2 which will be used in the proof of Theorem 4.5.

LEMMA 4.1. *For a non-empty task set T, if the selected two tasks from the task set by Algorithm 1 are t_i and t_j, the combination of such two tasks generates a new task t=t_{ij}. For any other task $t_k \in T$, we have $d(t_i, t_j) \geq \max\{d(t_i, t_k),\ d(t_j, t_k)\}$ and $d(t, t_k) \leq \min\{d(t_i, t_k),\ d(t_j, t_k)\}$.*

PROOF. Note that t_i and t_j are the selected candidate tasks by Algorithm 1. It means that for a task set T, the current maximal overlap value is $d(t_i, t_j)$. Without loss of generality, we assume $d(t_i, t_k) \geq d(t_j, t_k)$.

First, there exists a task notated as t_k. $t_k \in T$, $t_k \neq t_i$, $t_k \neq t_j$. If $d(t_i, t_j) < d(t_i, t_k)$, then the value of overlap between tasks t_i and t_j is not the maximal. It is a contradiction because we compute the current maximal overlap value by Algorithm 1. Thus, we have $d(t_i, t_j) \geq d(t_i, t_k)$.

Second, if a task t is the resultant task by the combination of tasks t_i and t_j, i.e., $t{=}t_{ij}$, then $d(t, t_k){=}d(t_i, t_j, t_k)$. Since $d(t_i, t_j, t_k) \leq d(t_j, t_k)$ is always satisfied, we have $d(t, t_k) \leq d(t_j, t_k)$. □

LEMMA 4.2. *The overlap value between two tasks, according to Algorithm 1, is monotone non-increasing.*

PROOF. Consider a non-empty task set T and the overlapping tasks t_i and t_j. For $\forall t_k \in T$, $d(t_i, t_j) \geq o(t_{ij}, t_k)$ always establishes according to Lemma 4.1. Then we can prove the Lemma 4.2 by using the method of mathematical induction. Since we identify a task pair with the maximal overlap value from the task set T in each combination step. If t_{ij} is the current choice, it means other task pairs cannot provide more data sharing than t_i and t_j in the current step. After the combination step, if a further task pair is t_x and t_y, we have $d(t_x, t_y) \leq d(t_i, t_j)$ according to Lemma 4.1. Thus Lemma 4.2 is proved. □

Definition 4.3 (*Compact model of a task*). For a task $t_i{=}{<}b_i, e_i, l_i{>}$, let $\widetilde{t_i}{=}{<}\widetilde{b}, \widetilde{e}, \widetilde{l}\ {>}$ denote its compact model such that:

$$\begin{cases} \widetilde{b} = b_i \\ \widetilde{e} = e_i \\ l_i = e_i - b_i \end{cases} \tag{9}$$

Definition 4.4 (*Compact model of a task set*). For a task set $T{=}\{t_1, ..., t_n\}$ where $t_i{=}{<}b_i, e_i, l_i{>}$ and $1 \leq i \leq n$, its compact model consists of φ compact tasks which are not overlapping with each other. These compact tasks $\widetilde{t_1}, \widetilde{t_2}, ..., \widetilde{t_\varphi}$ have the same interval length δ such that:

$$\begin{cases} \delta \leq \min\{l_1, l_2, ..., l_n\},\ \delta \textit{ is a positive constant.} \\ \varphi = \min\{x | x{\cdot}\delta \geq \sum_{i=1}^{n} l_i,\ x \textit{ is a positive integer.}\} \end{cases} \tag{10}$$

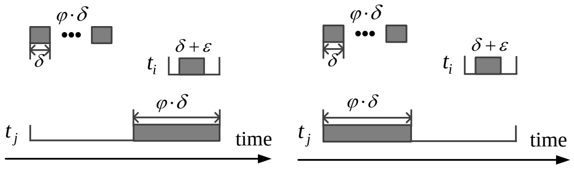
THEOREM 4.5. *Algorithm 1 is a 2-factor approximation scheduling algorithm.*

PROOF. For a non-empty task set T, if $|T|{=}1$, Algorithm 1 returns the only one sampling task which is the optimal result.

When $|T|{>}1$, assume I_i and I_j are the corresponding sampling intervals of tasks t_i and t_j, respectively. t_i and t_j are overlapping. I is the sampling interval of a task set notated as T' and $T'{=}T{-}\{t_i, t_j\}$. Considering the task t_i, in the worst case, Algorithm 1 returns the resultant interval length $|I_i \uplus I_j \uplus I|$ as illustrated in Fig. 6(a). However, there exists an optimal algorithm which derives the resultant interval length $|I_i \uplus I|$ as illustrated in Fig. 6(b). Therefore, we have:

$$\begin{aligned} \frac{GREEDY(T)}{OPT(T)} &= \frac{|I_i \uplus I_j \uplus I|}{|I_i \uplus I|} \\ &= \frac{\theta \cdot \delta + |I_i \uplus I_j|}{\theta \cdot \delta + |I_i|} \\ &\leq 1 + \frac{|I_j|}{\theta \cdot \delta + |I_i|} \\ &\leq 1 + \frac{\theta \cdot \delta}{\theta \cdot \delta + \delta} \\ &\leq 2. \end{aligned} \tag{11}$$

Here, $\delta \leq d(t_i, t_j)$. □



(a) Greedy result (b) Optimal result

Fig. 6. A typical example of the approximate result by using Algorithm 1 vs. the optimal result.

```
ALGORITHM 3: Allocation_Prune
Input: A task set $T$ and $T\neq\emptyset$. A sensor node set $S$ and $S\neq\emptyset$. $1\leq i\leq m$. $r\leq k$.
Output: A task set where there are no overlapping tasks.
repeat
    allocate a task $t$, $t\in T$ to $k$ candidate sensor nodes;
    compute the value of data sharing $d(t,\cdot)$ between $t$ and any other tasks on a sensor node;
    if no sampling task is overlapping with $t$ then
        randomly allocate $t$ to one candidate sensor node;
    else
        remove the task $t$ from a candidate sensor node when $t$ has the smallest value of data
        sharing with other tasks on the node;
    end
    $t.count=t.count-1$;
    if $t.count==r$ then
        remove $t$ from $T$;
    end
until $T\neq\emptyset$;
Combine for each task subset $T_i$;
```

A typical example is shown in Fig. 6. Algorithm 1 returns the interval length $2\theta\cdot\delta$ where $\delta<d(t_i,t_j)$, while the optimal result is $\theta\cdot\delta+\varepsilon$. Thus:

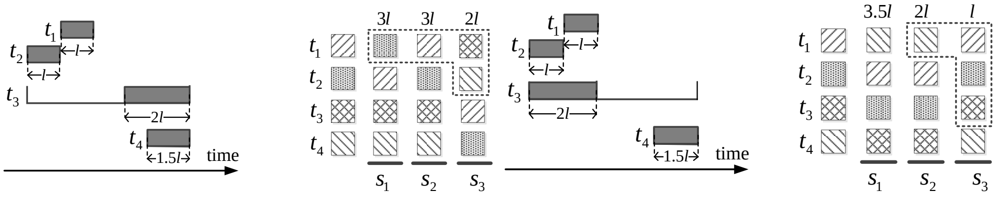
$$\lim_{\varepsilon\rightarrow 0,\theta\rightarrow\infty}\frac{\theta\cdot\delta+\theta\cdot\delta}{\theta\cdot\delta+\delta+\varepsilon}=2. \tag{12}$$

Our algorithm performs better when a task is overlapping with others tightly. Experimental results in Section 6 have verified this viewpoint.

5. A 2-FACTOR APPROXIMATION ALLOCATION ALGORITHM DESIGNING FOR WSNS

In this section, we provide three algorithms for the task allocation problem. The first is a random algorithm which allocates the sampling tasks randomly. The second is a pruning algorithm which first allocates a task to all candidate nodes and then removes it from some of the candidate sensor nodes by its overlap value with other tasks. The third is a 2-factor approximation algorithm which allocates sampling tasks by iteratively combining them until no overlapping tasks exist.

For any task set, the insight of the random allocation method is to randomly identify r out of k candidate sensor nodes. This method is simple and easy to be performed on a sensor node. However, it ignores the overlap amongst the tasks and brings much unnecessary sampled data which consumes much energy of sensor nodes and damages the quality of communication in WSNs as well.



(a) The approximate result of task allocation by using Algorithm 3.

(b) The optimal result of task allocation.

Fig. 7. A typical example of the approximate result by using Algorithm 3 vs. the optimal result. Tasks in the dotted polygon will be removed from nodes due to pruning operations.

.

5.1. Task allocation according to the pruning operation

We notice that the sampled data can be shared in the overlapping time region. Therefore, we consider allocating tasks via the pruning operation which removes unreasonable allocation choices repeatedly. The entire process contains three major steps. First, we allocate tasks to all candidate sensor nodes. That is, if a task is detected by k sensor nodes, it is allocated to all the k candidate sensor nodes. Second, compute the maximal overlap value between a task and other overlapping tasks. Then remove a task from a sensor node where the task has the smallest overlap value with other tasks until it is allocated to r sensor nodes. Finally, compute the total length of sampling intervals. Algorithm 3 describes more details.

In Algorithm 3, lines $2-3$ allocate a task to its k candidate sensor nodes. Lines $4-8$ check whether a task has been removed from $k-r$ candidate sensor nodes. Line 11 computes the final sampled data for each sensor node using Algorithm 1. In the worst case, if Algorithm 1 is used to compute the sampled data, the time complexity of Algorithm 3 is $O(n^2)$, and the space complexity is $O(n^2)$. If Algorithm 2 is used to compute the sampling time, then the space complexity of Algorithm 3 is $O(k \cdot n/m)$ because each task maintains its k maximal overlap value for all candidate sensor nodes.

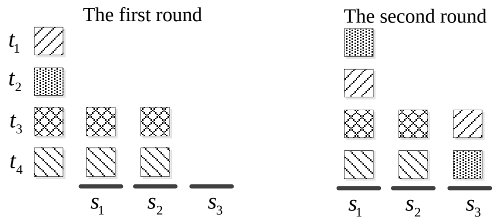
Algorithm 3 is a greedy algorithm. A typical example is shown in Fig. 7. Here, k=3 and r=2. The sensor node set is notated as S where S=$\{s_1, s_2, s_3\}$. The task set is notated as T and T=$\{t_1, t_2, t_3, t_4\}$. As illustrated in Fig. 7, we use four different types of rectangles in the right to stand for the tasks in the left. The pruning allocation method first allocates all the sampling tasks to the candidate nodes, and then removes the tasks which has smaller overlap with other tasks (indicated in dotted rectangular). Finally, Algorithm 3 returns the total length of sampling intervals $8l$ as shown in Fig. 7(a), while the optimal result is $6.5l$ as shown in Fig. 7(b). The reason is that our pruning procedure is not optimal in each round. This motivates us to propose Algorithm 4 which allocates tasks according to the combination operation and can be optimal for each round.

5.2. Tasks allocation according to the combination operation

The combination operation is utilized to schedule sampling tasks in Algorithm 1. Furthermore, the allocation problem can be solved by the combination operation. As illustrated in Fig. 2, a task t is detected by three sensor nodes, but only two of them are identified to perform it. If the task t is overlapping with three tasks t_1, t_2 and t_3 such that $d(t, t_1)$>$d(t, t_2)$>$o(t, t_3)$, we should allocate t to the sensor nodes s_1 and s_2. The task allocation method includes three major operations. First, maintain a global quad list in which each quad stands for a combination operation of two overlapping tasks. Second, iteratively identify a quad which has the maximal overlap value of overlap-

ALGORITHM 4: Allocation_Combine

```
Input: A task set $T$ and $T \neq \emptyset$. A sensor node set $S$ and $S \neq \emptyset$. $r \leq k$, $|S| = m$.
Output: A task set where there are no overlapping tasks.
maintain a task set $T_i$ for the sensor node $s_i$;
$Q_i$ = Pre_processing ($T_i$), and sort $Q_i$ in the descending order of the overlap value;
repeat
    identify a task pair $<t_1, t_2>$ from $T_i$ which has the maximal overlap value;
    combine $t_1$ and $t_2$ and notate the resultant task as $t$;
    remove $t_1$ and $t_2$ from $T_i$;
    add $t$ into $T_i$, and update $Q_i$;
until there exists overlapping tasks in $T_1, ..., T_m$;
repeat
    randomly allocate a task $t$ ($t \in T_i$) to a sensor node which is not allocated $t$ before;
    remove $t$ form $T_i$ if the task is allocated to $r$ out of $k$ sensor nodes;
until $T_i$ ($1 \leq i \leq k$) is not an empty set;
Combine for each task subset $T_i$;
```



(a) The first round of task combination.

(b) The second round of task combination.

Fig. 8. The illustrative example for the allocation of tasks by combining overlapping tasks in each round.

ping tasks in the global quad. Then allocate them to r out of k candidate sensor nodes. Finally, update the quad list and the task set. Algorithm 4 presents the method of task allocation in detail.

For example, let $T = \{t_1, t_2, t_3, t_4\}$ denote a task set where $t_1 = <0, 3, 3>$, $t_2 = <2, 6, 2>$, $t_3 = <3, 8, 4>$ and $t_4 = <4, 8, 4>$. There are three sensor nodes s_1, s_2, and s_3. Here $k=3$ and $r=2$. The task sets of these sensor nodes are marked as T_1, T_2, and T_3 with $T_1 = T_2 = T_3 = \{t_1, t_2, t_3, t_4\}$. We then maintain a quad list for each sensor node, i.e., $Q = \{q_{12}, q_{23}, q_{24}, q_{34}\}$ and sort Q in the descending order based on the value of data sharing. When Algorithm 4 is used to allocate tasks, the task pair $<t_3, t_4>$ is combined in the first round due to $d(t_3, t_4) = 4$. As illustrated in Fig. 8(a), tasks t_3 and t_4 should be allocated to sensor nodes. A new task $t_{34} = <4, 8, 4>$ is generated. We further remove tasks t_3 and t_4 from T_1, and add t_{34} into T_1. After t_3 and t_4 are allocated, all the quads containing tasks t_3 and t_4 will be removed from the quad list, i.e., $Q = \{q_{12}\}$. Thus, tasks t_1 and t_2 should be combined and allocated to sensor nodes. As indicated in Fig. 8(b), after two rounds of task combination, tasks in T are finally allocated to the sensor nodes in S, and the volume of sampled data has been computed.

In Algorithm 4, lines $1-2$ maintain a global quad list. Lines $3-7$ present the combination of the overlapping tasks. Lines $8-11$ ensure that all tasks are allocated even though some tasks are not overlapping with other sampling tasks. The time complexity is $O(n^2)$, and the space complexity is $O(n^2)$. If Algorithm 2 is used to compute to sampled data for a task set, then the space and time complexity of Algorithm 4 are $O(1)$ and $O(n^2)$, respectively.

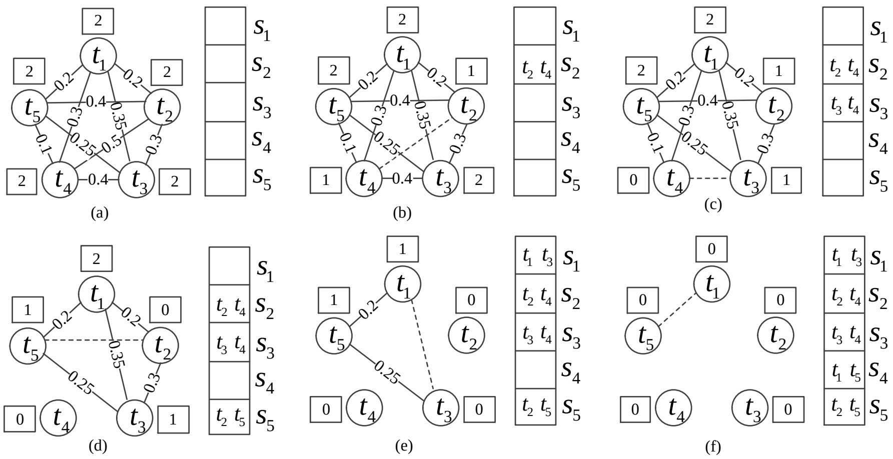


Fig. 9. The illustrative example of transformation from Algorithm 4 to Algorithm 1 by combining the overlapping tasks which have the maximal value of data sharing in each step. Here, k=3 and r=2.

THEOREM 5.1. *Algorithm 4 is a 2-factor approximation allocation algorithm.*

PROOF. When r=k, the allocation solution is deterministic. We can allocate tasks to all of its candidate sensor nodes. Algorithm 4 means to run Algorithm 1 on each sensor node for its allocated tasks. It always returns a 2-factor approximate result for each sensor node. Algorithm 4 is thus a 2-factor approximation algorithm.

When $1 \le r < k$, Algorithm 4 selects a task pair with the maximal value of data sharing from the task set repeatedly. Algorithm 4 can be transformed to Algorithm 1. The process of transformation can be described as follows:

- Compute the value of data sharing between overlapping tasks which may be allocated to a same node.
- For any two tasks which are not allocated to a same node, we set its the value of data sharing to negative infinity.
- If overlapping tasks have been allocated to a node, the data sharing between them will be adjusted to negative infinity.
- If a task has been allocated to r nodes, data sharing between it and other nodes will be set to negative infinity.

Then perform Algorithm 1 on the task set until there is no overlapping tasks. We get the minimal volume of sampled data. Meanwhile, we get a sampling interval set Φ in which none of intervals are overlapping. An interval I in Φ satisfies several sampling tasks. Thus, we get several sets of tasks. Since a task has been allocated to r nodes, these sets of tasks are the result of task allocation. Therefore, the computation of the volume of sampled data can be solved by running the Algorithm 1 repeatedly. The performance of Algorithm 1 has been proved by Theorem 4.5 and returns a 2-factor approximate result of the volume of sampled data. Thus, Algorithm 4 is a 2-factor approximation algorithm. □

To be clear, we take an example to illustrate the process of transformation. As shown in Fig. 9, assume there are five tasks $t_1, ..., t_5$ (as indicated by cycles) which should be allocated to five nodes $s_1, ..., s_5$. $\{t_1, t_2, t_3\}$ are detected by s_1. Similarly, $\{t_2, t_3, t_4\}$, $\{t_3, t_4, t_5\}$, $\{t_4, t_5, t_1\}$ and $\{t_5, t_1, t_2\}$ are detected by nodes s_2, s_3, s_4 and s_5, respectively. Here, k=3 and r=2. As illustrated in Fig. 9(a), we construct a weighted graph where

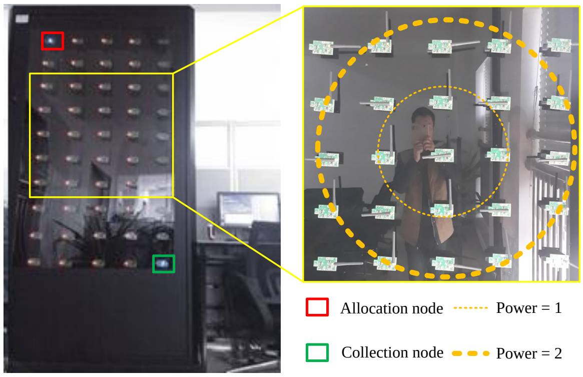


Fig. 10. The transmission power can be adjusted to realize the k-coverage network in the testbed.

a vertex stands for a task. If two tasks are overlapping, then a weighted edge exists. The weighted value represents the value of data sharing. The value in the rectangle indicates the number of nodes which the task has been allocated to. The overlapping tasks which have the maximal data sharing are allocated to a node. Then, the data sharing between them is set to negative infinity (indicated by the dotted line). If a task has been allocated to r sensor nodes, then the task is removed from the task set. Without loss of generality, the new tasks generated from the combination of original tasks are not demonstrated. For example, in Fig. 9(b), we find that tasks t_2 and t_4 have the maximal value of data sharing: 0.5. Then, allocate them to the node s_2, and set the data sharing between them to be negative infinity. As illustrated in Fig. 9(d), when the task t_4 has been allocated to s_2 and s_3, the edges between it and other nodes are removed. When the sampling interval of tasks are scheduled to achieve the maximal data sharing, these tasks are eventually allocated to and sampled by the nodes. Since the process of task combination will generate new tasks, new vertices will added to the graph, but this will not impact the performance of Algorithm 4.

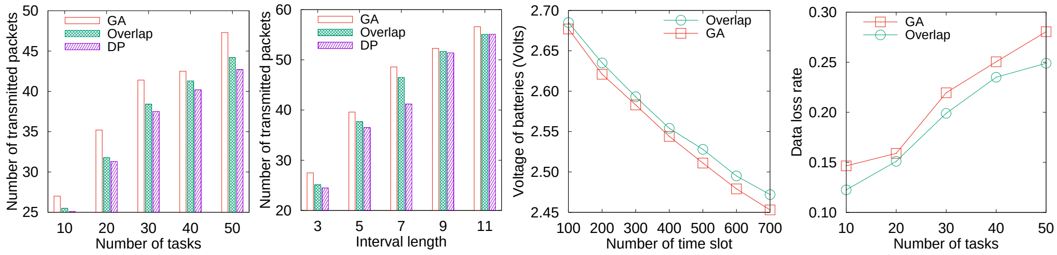
6. PERFORMANCE EVALUATION

In this section, we first introduce our experimental environment and settings. Then, we evaluate the effectiveness of our proposed algorithms by using a physical testbed containing 50 wireless sensor nodes. Finally, a widely-used simulation tool, i.e., TOSSIM, is used to verify the scalability of our proposals.

6.1. Experimental environment and settings

We evaluate the effectiveness of our proposed algorithms on a physical testbed of WSNs. As indicated in Fig. 10, this testbed contains 50 wireless sensor nodes. The distance between two adjacent sensor nodes is about 20cm. In the experiments, we construct different k-coverage networks by adjusting transmission power of nodes. The parameter k becomes large when the transmission power of nodes is increased. By default, the power is set to the lowest level, and the transmission range is only about tens of centimeters. All the nodes operate on the same channel.

With such settings, although a number of nodes locate in the same collision domain, the packet loss rate observed in the experiments is less than 0.1%. The reason is that the nodes in the same collision domain access the wireless channel for transmission based on the MAC protocol, and packet loss caused by severe interference will not happen [Demirkol et al. 2006], [Bharghavan et al. 1994].



(a) l=5 (b) n= p30 (c) $l = 5, n = 30$ (d) l=5

Fig. 11. A comparison of the number of transmitted packets by varying the number of tasks in (a), and the interval length in (b). Energy consumption is compared by varying the number of time slot in (c), and data loss rate in (d).

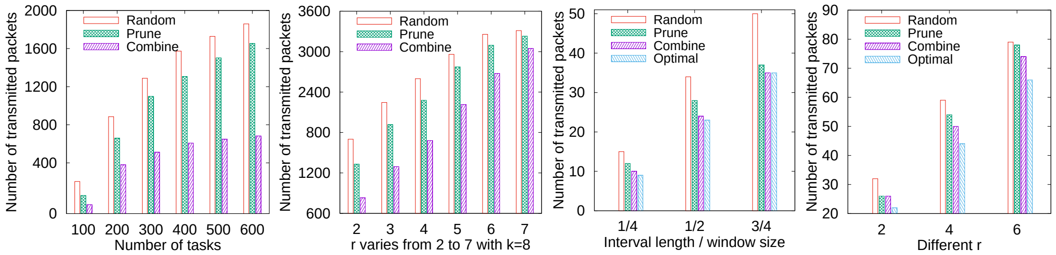
Since the WSN is densely deployed, any two nodes can set up a wireless link by at most three hops. The proposed algorithms are implemented in a centralized manner which is widely used in actual systems [Mo et al. 2009], [Jiang et al. 2009], [Sheng et al. 2007],[Xiang et al. 2011]. We use the left-top node (as highlighted by a red rectangular) as the sink node which allocates a task set to other nodes, and the right-bottom node as the collection node (as indicated by the green rectangular) which collects all sampled data. Transmitted packets are counted to indicate the required sampled data, while received packets are collected to demonstrate the actual collected data.

The value of k is varied from 2 to 8, which is reasonable and always used in the practical WSN systems [Mao et al. 2013], [Huang and Tseng 2003]. We construct a unit of continuous data by sampling temperature 5 times per second, and sent the data by using a packet. The interval length of a sampling task is randomly generated and not greater than 10. To be specific, the begin time of a sampling task is evenly distributed in time slot $[0,\ 50]$. We run each algorithm by 10 times and use the average value of these results as the final result. To be clear, m, n, and l represent the cardinality of the sensor node set and the task set, and the length of a sampling interval, respectively.

6.2. Performance of data sharing for a sampling task set

At first, we evaluate the performance of our Algorithm 1, denoted by '*Overlap*', which returns a reorganized sampling task set based on the idea of data sharing, against the state-of-the-art method [Fang et al. 2013], named '*GA*' here. *GA* is an approximation algorithm for optimizing the data sampling of a task set on a single sensor node. Meanwhile, *GA* can derive the optimal scheduling algorithm, named as '*DP*', using dynamic programming technique on the condition that all tasks have the same interval length. In this experiment, the interval length of each sampling task is consistently set to 5, and the window size of each task varies from 5 to 15.

Fig. 11(a) and Fig. 11(b) illustrate the number of transmitted packets of task scheduling methods, i.e., *Overlap*, *GA* and *DP*, by varying the cardinality of a sampling task set (Fig. 11(a)) or the interval length of a sampling task (Fig. 11(b)). Specifically, the interval length of a sampling task is set to 5 in Fig. 11(a) while the cardinality of each task set is fixed to 30 in Fig. 11(b). It is easy to observe from Fig. 11(a) and Fig 11(b) that the number of transmitted packets increases with the growth of both the number of tasks and the interval length. However, our *Overlap* method performs better and returns smaller number of transmitted packets than *GA*. This happens because our scheduling algorithm combines the maximal overlapping tasks each time. Thus, every scheduling step is the current optimal choice. Considering *GA* schedules tasks based on the end time of a task, it cannot make sure each scheduling choice is optimal. That is the reason why the *Overlap* outperforms the *GA*. Meanwhile, Fig. 11(a)



(a) k=5, r=2

(b) k=8, n=30

(c) k=5, r=2, n=5

(d) k=8, n=5

Fig. 12. A comparison of the number of transmitted packets by varying the number of tasks in (a) and the value of r in (b). When the scale of tasks is limited, the number of transmitted packets is compared by varying the interval length of tasks in (c) and the value of r in (d).

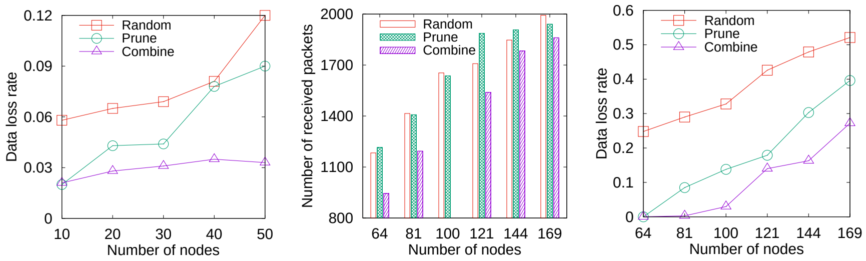
and Fig. 11(b) indicate *Overlap* achieves 2-factor approximation result vs. the optimal result. This verifies the correctness of the Theorem 4.5.

In Fig. 11(c), we run both *Overlap* and *GA* methods on two sensor nodes to test their energy usage. The terminal voltage of batteries equipped for a sensor node is measured every 100 time slots. The initiate value is 2.864V. It is apparent that while the terminal battery voltage decreases due to the energy consumption, our scheduling algorithm *Overlap* consumes less energy than that *GA* does. This is because that *Overlap* reduces more unnecessary sampled data than *GA*. It is well known that a sensor node uses up much more energy when listening, receiving and sending data. So our scheduling algorithm which greatly cuts down energy consumption prolongs the lifetime of the entire wireless sensor network. In Fig. 11(d), we test the data loss rate during data transmission of different methods by changing the number of tasks on a single sensor node. We observe that the data loss rate increases with the growth of the number of tasks. We are delighted to see that our *Overlap* method achieves the smaller data loss rate than *GA*, which improves the communication quality and the reliability of the whole network. To be specific, the great reduction of unnecessary amount of sampled data by using *Overlap* method relieves the workload of intra-network communication and decreases transmission delay and congestion.

6.3. Performance of data sharing amongst multiple sensor nodes

In this part, we verify the performance of the random allocation of tasks, Algorithm 3 and Algorithm 4 which are represented as '*Random*', '*Prune*' and '*Combine*', respectively hereinafter.

In Fig. 12(a), we vary the number of sampling tasks in WSNs to compare the number of transmitted packets. By default, k is set to 5 and r is set to 2. Fig. 12(a) shows the number of transmitted packets increases with the number of tasks. Moreover, methods *Prune* and *Combine* significantly decrease unnecessary sampled data than *Random*, especially when the number of tasks grows. Precisely, when the cardinality of a task set is larger than 600, the number of transmitted tasks produced by the *Combine* seems to be half of that brought by *Random*. In Fig. 12(b), we compare the number of transmitted packets by varying r from 2 to 6 under the setting of k=8. It is obvious that both *Prune* and *Combine* reduce more transmitted packets than *Random* does. Another observation is that the advantage of *Prune* and *Combine* becomes less significant when the growth of r. That is because when r increases, more candidate sensor nodes are involved to be identified to execute a task. The randomness of *Random* is weakened. Particularly, when r equals k, *Random* provides a deterministic allocation which shares the same performance with *Combine*.



(a) k=5, r=2, the testbed (b) k=5, r=2, TOSSIM (c) k=5, r=2, TOSSIM

Fig. 13. A comparison of data loss rate in the testbed is presented by varying the number of nodes in (a). The number of received packets is compared by varying the number of nodes in (b) and so is the data loss rate in (c) for large scale wireless sensor networks simulated in TOSSIM.

The task allocation problem amongst sensor nodes is difficult to derive an effective optimization solution. However, when the scale of tasks is small, we can find an optimal allocation solution by using a brute-force method. To evaluate the performance of our allocation rigorously, we compare the number of transmitted packets on each method by varying the number of tasks and the value of r. In Fig. 12(c), we display three groups of tasks and each of which has five tasks. The interval length of a task is set to $1/4$, $1/2$, and $3/4$ of its window size in the group 1, 2, and 3, respectively. These tasks appear in time slot $[0,\ 20]$ randomly. Here, k=5, r=2. It is clear that the number of transmitted packets increases with the expansion of the interval length. However, *Prune* and *Combine* perform better than Random and are closely to the optimal allocation solution, i.e., *Optimal*. This confirms the conclusion of Theorem 5.1 again which clarifies that our greedy allocation algorithm is a 2-factor approximation of the optimal solution. In Fig. 12(d), we set k=8, and modify the value of r from 2 to 6. A quick conclusion drawn from the figure illustrates that the number of transmitted packets increases with the growth of r. Under such condition, the *Combine* achieves greatly smaller number of transmitted packets than twice of that produced by the optimal solution. It verifies the conclusion of Theorem 5.1 again.

Large amount of sampled data definitely leads to server delay and congestion in WSNs, degrading the quality of data transmission as a result. In Fig. 13(a), we compare the data loss rate of every allocation method by changing the number of nodes in WSNs. It is obvious that the data loss rate becomes larger when the scale of network increases. But *Prune* and *Combine* significantly derive the smaller data loss rate than *Random* does. This verifies the benefits of our allocation solution in improving the quality of data transmission by cutting down the unnecessary sampled data.

We conduct simulations for evaluating the scalability of our proposed algorithms for a large scale wireless sensor network. These simulations are implemented with TOSSIM which is a widely-used simulation tool for wireless sensor networks [Levis et al. 2003]. We construct a grid network using the widely accepted settings in [Sohrabi et al. 1999]. As illustrated in Fig. 13(b), we compare the number of received data by varying the scale of the network. We apparently observe that *Combine* brings smaller amount of received data than that *Random* does. The reason is that *Combine* reduces more unnecessary sampled data than *Prune* and *Random*. As illustrated in Fig. 13(c), the data loss rate in both *Combine* and *Prune* is much smaller than that in *Random*. It happens when the number of transmitted data in *Combine* and *Prune* is smaller than that in *Random*. In summary, *Combine* is more suitable to be deployed for a large-scale wireless sensor network as it always performs much better than *Random*.

7. DISCUSSION

We have proposed methods of task allocation and scheduling of sampling interval and given much theoretical analysis about their performance. As the question is general, we aim to provide a universal solution which does not rely on the details of network. The sink node gets the strategy of task allocation in a wireless sensor network. It then allocates the sampling tasks to the sensor nodes. The allocation message will be added into packets and disseminated to sensor nodes with sampling tasks together. Comparing to the volume of sampled data of sampling tasks, such expenditure on the allocation solution is rather little. Meanwhile, as illustrated in Section 6.3, adopting the strategy of task allocation can reduce redundancy of sampling data by more than 30%. Therefore, it is worthy to trading little expenditure for appreciable data sharing. Besides, some other details such as protocols of communication and data routing do not impact performance of the proposed algorithms. We do not adopt optimization on such communication protocols or routing strategy.

The allocation of tasks involves assigning each task to r out of k candidate sensor nodes. The current allocation scheme seeks to fully exploit the benefits of data sharing amongst tasks. In reality, there still exists other important constraints that can be exploited to improve the proposed allocation schemes. Note that each sensor node is resource-constrained, i.e., it has limited computation and memory resources. If a sensor node has been allocated many sampling tasks, it may not handle all the tasks timely and exhaust the energy at early time. This problem cannot thus be simply addressed by limiting the number of tasks a sensor node carries. The reason is that although different nodes have the same amount of sampling tasks, the real work load of sensor nodes may have a considerable difference due to the data sharing strategy. Therefore, contemporary task allocation schemes can be more practical if we consider the load balance amongst sensor nodes. An effective method is to set a threshold for the constrained resource. Each allocation step makes sure that the threshold value is not exceeded. Algorithm 3 and Algorithm 4 are flexible to adjust for this tactics. When the limited resource changes dynamically, the allocation problem will be more difficult to solve. We leave it as our future work.

In this paper, we aim to minimize the sampled data for a sampling task set by cooperatively allocating tasks and scheduling sampling interval of tasks which are allocated to a sensor node. Our proposed algorithms do not rely on the topology of a network. In fact, the information of a network can be used to improve the performance of our algorithms on many respects. For example, even though many applications require to get the entire data of sampling interval, we can divide the sampling interval into several segments for a k-coverage network if these segments can be integrated into the complete task finally on the sink node. These sampling segments can be allocated to the sensor nodes, which will help to balance the sampling workload on sensor nodes in our proposals. However, embedding the network information into our solutions introduces arduous problems, including task dividing, data fusion and so forth. We leave it as our future work as well.

8. CONCLUSION

Many applications of WSNs pursue to perform a set of interval data sampling tasks for decision-making. In this paper, we focus on minimizing the interval data sampling in a k-coverage and r-redundant WSN. The solving of this optimal problem depends on the optimization of two subproblems. The task allocation problem, MIN-SSA, allocates a sampling task to r out of k candidate nodes targeting at minimizing the amount of sampled data in the whole network. The scheduling problem, MIN-SA, schedules the sampling tasks targeting at minimizing the amount of sampled data on a single sensor node. We design a 2-factor approximation scheduling and allocation method for each arduous problem. The effectiveness and scalability of our proposals is evaluated by employing a testbed and a widely-used simulation tool, i.e., TOSSIM, respectively. The evaluation results indicate that our methods significantly reduce the amount of sampled data, considerably save precious energy and apparently improve the quality of communication due to the decrease of the data loss rate.

ACKNOWLEDGMENTS

The work is partially supported by the National Natural Science Foundation of China (NSFC) under Grant 61402494, Grant 61402513, and Grant 61202487.

REFERENCES

T. Arici, B. Gedik, Y. Altunbasak, and L. Liu. 2003. PINCO: a Pipelined In-Network COmpression Scheme for Data Collection in Wireless Sensor Networks. In *IEEE Proceedings of 12th International Conference on Computer Communications and Networks*. 539–544.

Vaduvur Bharghavan, Alan Demers, Scott Shenker, and Lixia Zhang. 1994. MACAW: a media access protocol for wireless LAN's. In *ACM SIGCOMM*. London, UK, 212–225.

M. Cerullo, G. Fazio, M. Fabbri, F. Muzi, and G. Sacerdoti. 2005. Acoustic signal processing to diagnose transiting electric trains. *IEEE Transactions on Intelligent Transportation Systems* 6, 2 (2005), 238–243.

Ilker Demirkol, Cem Ersoy, and Fatih Alagoz. 2006. MAC protocols for wireless sensor networks: a survey. *IEEE Communications Magazine* 44, 4 (2006), 115–121.

Xiaolin Fang, Hong Gao, Jianzhong Li, and Yingshu Li. 2013. Application-aware data collection in Wireless Sensor Networks. In *Proc. IEEE INFOCOM*. Turin, Italy, 1645–1653.

Chi-Fu Huang and Yu-Chee Tseng. 2003. The Coverage Problem in a Wireless Sensor Network. In *Proc. ACM WSNA*. San Diego, California, USA, 115–121.

Mingxing Jiang, Zhongwen Guo, Feng Hong, Yutao Ma, and Hanjiang Luo. 2009. OceanSense: A practical wireless sensor network on the surface of the sea. In *Proc. IEEE PerCom*. Galveston, Texas, USA, 1–5.

Philip Levis, Nelson Lee, Matt Welsh, and David Culler. 2003. Tossim: accurate and scalable simulation of entire tinyos applications. In *Proc. ACM Sensys*. Los Angeles, California, USA, 126–137.

Mo Li and Yunhao Liu. 2007. Underground Structure Monitoring with Wireless Sensor Networks. In *Proc. IEEE IPSN*. Cambridge, Massachusetts, USA, 69–78.

Xufei Mao, Yunhao Liu, Shaojie Tang, Huafu Liu, Jiankang Han, and Xiang-Yang Li. 2013. Finding Best and Worst k-Coverage Paths in Multihop Wireless Sensor Networks. *IEEE Transactions on Parallel and Distributed Systems* 24, 12 (2013), 2396–2406.

Lufeng Mo, Yuan He, Yunhao Liu, Jizhong Zhao, Shao-Jie Tang, Xiang-Yang Li, and Guojun Dai. 2009. Canopy Closure Estimates with GreenOrbs: Sustainable Sensing in the Forest. In *Proc. ACM SenSys*. Berkeley, California, USA, 99–112.

S. S. Pradhan, J. Kusuma, and K. Ramchandran. 2002. Distributed compression in a dense microsensor network. *IEEE Signal Processing Magazine* 19, 2 (2002), 51–60.

Bo Sheng, Qun Li, Weizhen Mao, and Wen Jin. 2007. Outlier Detection in Sensor Networks. In *Proc. ACM MobiHoc*. Montreal, Quebec, Canada, 219–228.

K. Sohrabi, B. Manriquez, and G.J. Pottie. 1999. Near ground wideband channel measurement in 800-1000 MHz. In *Proc. IEEE Vehicular Technology Conference*. Houston, Texas, USA, 571–574.

Wen Zhan Song, Fenghua Yuan, and Richard LaHusen. 2006. Time-Optimum Packet Scheduling for Many-to-One Routing in Wireless Sensor Networks. In *Proc. IEEE MASS*. Vancouver, Canada, 81–90.

Makoto Suzuki, Shunsuke Saruwatari, Narito Kurata, and Hiroyuki Morikawa. 2007. A High-density Earthquake Monitoring System Using Wireless Sensor Networks. In *Proc. ACM SenSys*. Sydney, Australia, 373–374.

Robert Szewczyk, Eric Osterweil, Joseph Polastre, Michael Hamilton, Alan Mainwaring, and Deborah Estrin. 2004. Habitat Monitoring with Sensor Networks. *Communications of the ACM-Wireless sensor networks* 47, 6 (2004), 34–40.

Arsalan Tavakoli, Aman Kansal, and Suman Nath. 2010. On-line sensing task optimization for shared sensors. In *Proc. ACM/IEEE IPSN*. Stockholm, Sweden, 47–57.

Niki Trigoni, Yao Yong, Alan Demers, Johannes Gehrke, and Rajmohan Rajaraman. 2005. Multi-query Optimization for Sensor Networks. *Springer Lecture Notes in Computer Science* (2005), 307–321.

Liu Xiang, Jun Luo, and A. Vasilakos. 2011. Compressed data aggregation for energy efficient wireless sensor networks. In *Proc. IEEE SECON*. Salt Lake City, Utah, USA, 46–54.

Shili Xiang, Hock Beng Lim, K.-L. Tan, and Yongluan Zhou. 2007. Two-tier multiple query optimization for sensor networks. In *Proc. IEEE ICDCS*. 3–9.

Ning Xu, Sumit Rangwala, Krishna Kant Chintalapudi, Deepak Ganesan, Alan Broad, Ramesh Govindan, and Deborah Estrin. 2004. A Wireless Sensor Network For Structural Monitoring. In *Proc. ACM SenSys*. Baltimore, Maryland, USA, 13–24.

You Xu, Abusayeed Saifullah, Yixin Chen, Chenyang Lu, and Sangeeta Bhattacharya. 2010. Near Optimal Multi-application Allocation in Shared Sensor Networks. In *Proc. ACM MobiHoc*. 181–190.