CATS: Cooperative Allocation of Tasks and Scheduling of Sampling Intervals for Maximizing Data Sharing in WSNs

YAWEI ZHAO and DEKE GUO, National University of Defense Technology
JIA XU and PIN LV, Guangxi University
TAO CHEN and JIANPING YIN, National University of Defense Technology

Data sharing among multiple sampling tasks significantly reduces energy consumption and communication cost in low-power wireless sensor networks (WSNs). Conventional proposals have already scheduled the discrete point sampling tasks to decrease the amount of sampled data. However, less effort has been expended for applications that generate continuous interval sampling tasks. Moreover, most pioneering work limits its view to schedule sampling intervals of tasks on a single sensor node and neglects the process of task allocation in WSNs. Therefore, the gained efforts in prior work cannot benefit a large-scale WSN because the performance of a scheduling method is sensitive to the strategy of task allocation. Broadening the scope to an entire network, this article is the first work to maximize data sharing among continuous interval sampling tasks by jointly optimizing task allocation and scheduling of sampling intervals in WSNs. First, we formalize the joint optimization problem and prove it NP-hard. Second, we present the *COMBINE* operation, which is the crucial ingredient of our solution. *COMBINE* is a 2-factor approximate algorithm for maximizing data sharing among overlapping tasks. Furthermore, our heuristic named *CATS* is proposed. *CATS* is 2-factor approximate algorithm for jointly allocating tasks and scheduling sampling intervals so as to maximize data sharing in the entire network. Extensive empirical study is conducted on a testbed of 50 sensor nodes to evaluate the effectiveness of our methods. In addition, the scalability of our methods is verified by utilizing TOSSIM, a widely used simulation tool. The experimental results indicate that our methods successfully reduce the volume of sampled data and decrease energy consumption 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

ACM Reference Format:
Yawei Zhao, Deke Guo, Jia Xu, Pin Lv, Tao Chen, and Jianping Yin. 2016. CATS: Cooperative allocation of tasks and scheduling of sampling intervals for maximizing data sharing in WSNs. ACM Trans. Sen. Netw. 12, 4, Article 29 (September 2016), 26 pages.
DOI: http://dx.doi.org/10.1145/2955102

This work is supported in part by the National Basic Research Program (973 program) under Grant No. 2014CB347800, the Hunan Provincial Natural Science Fund for Distinguished Young Scholars under Grant No. 2016JJ1002, the National Natural Science Foundation of China under Grants No.61422214, No. 61170287, No. 61232016, No. 61402494, No. 61402513, No. 61202487, the Program for New Century Excellent Talents in University, and the Preliminary Research Funding of NUDT under Grants No. JC10-05-02 and No. ZDYYJCYJ20140601.
Authors' addresses: Y. Zhao and J. Yin, College of Computer, National University of Defense Technology; emails: {zhaoyawei, jpyin}@nudt.edu.cn; D. Guo and T. Chen, Science and Technology on Information System Engineering Laboratory, National University of Defense Technology; emails: {dekeguo, chentao}@nudt.edu.cn; J. Xu and P. Lv, School of Computer, Electronics and Information, Guangxi University; emails: {xujia, lvpin}@gxu.edu.cn.

DOI: http://dx.doi.org/10.1145/2955102

1. INTRODUCTION

The successful applications of low-power wireless sensor networks (WSNs) under different scenarios, such as earthquake monitoring [Suzuki et al. 2007], railway diagnosis [Cerullo et al. 2005], wildlife protection [Szewczyk et al. 2004], and structure monitoring [Xu et al. 2004], are time consuming and labor intensive. They are all constrained by scarce resources, such as power, memory, and computation. Moreover, the experience of some real projects [Mo et al. 2009; Jiang et al. 2009; Mao et al. 2012] shows that the lifetime of a deployed network is constrained for several months without recharging. This phenomenon is extremely severe when data-intensive applications generate a large set of sampling tasks and need extensive sampled data.

Such data-intensive applications widely exist in monitoring systems. For example, acoustic data is collected to diagnose the state of a railway system [Cerullo et al. 2005]. The vibration data, sampled by volcano, earthquake, and structure monitoring systems [Tan et al. 2013; Suzuki et al. 2007; Xu et al. 2004], needs to be sampled to detect the anomaly. A further example is that observed data in a wildlife monitoring system can be used to conduct scientific analysis on animals' behavior [Szewczyk et al. 2004]. As illustrated in Figure 1(b), these applications first generate a large amount of sampling tasks. Then, those sampling tasks will be allocated to certain sensor nodes by a sink node (as indicated by the solid line). Finally, the collected sensor readings are transmitted to the sink node (as indicated by the dotted line) and then are fed into aforementioned applications.

Generally, such a sampling task can be defined as a task model by using two properties: time window and sampling interval, meaning a period of time. As illustrated in Figure 1(a), the time window relates to the lifetime of a sampling task. The sampling interval indicates that this task should be performed to sample data over the time interval continuously. Universally, sampling tasks can be classified into two cases: discrete point sampling tasks and continuous interval sampling tasks. The former is the special case of the task model. Such a discrete point sampling task requires a wireless sensor node to sample data once during the period. The latter is a general case. In other words, a continuous interval sampling task must be performed over a time interval continuously. Moreover, the sampling interval of a continuous interval sampling task can be adjusted to move in its time window flexibly. In this article, we focus on the latter (i.e., continuous interval sampling tasks).[1]

Note that the time window of more than one sampling task may have overlapping time regions. The sampling intervals of those tasks can be adjusted in the overlapping regions of time. Data, which is sampled in such overlapping regions once, can be shared by multiple tasks simultaneously. As illustrated in Figure 1(a), consider three sampling tasks t_1, t_2, and t_3. Since the time window of t_2 and t_3 are overlapping, the sampling intervals of t_2 and t_3 can be adjusted in the overlapping time region so that data sampled in the region of data sharing can be shared by t_2 and t_3. Such a strategy of data sharing is very practical to reduce redundant data sampling and improve the efficiency of WSNs. Moreover, this strategy is extremely important for interval sampling tasks. First, an interval sampling task usually produces a large amount of sampled data each time. Without such a data sharing strategy, delivering those superfluous sampled data to the sink node will consume more network resources. Second, the execution of unnecessary data sampling and the transmission of redundant sampled data consume nontrivial energy of the sensor node.

[1]For simplicity, the terms *interval sampling tasks*, *sampling tasks*, and *tasks* are hereafter termed *continous interval sampling tasks* without difference.

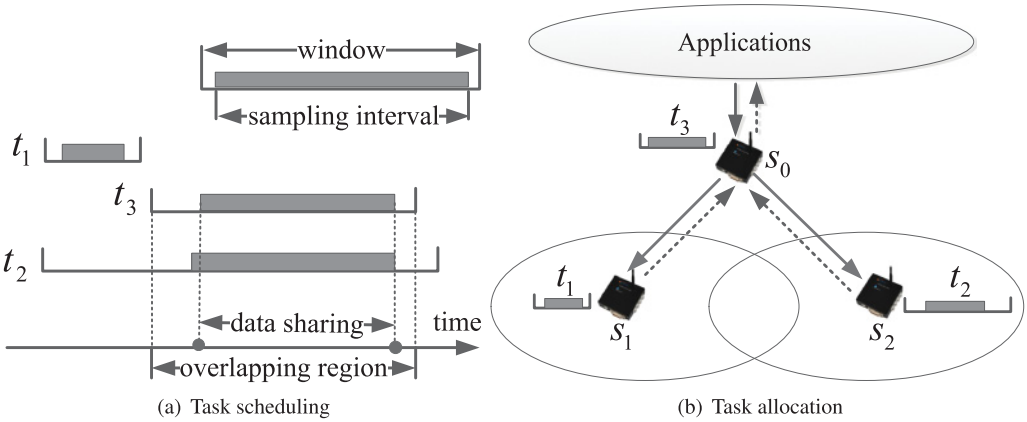


Fig. 1. Illustrative examples for the schedule of sampling intervals in the left panel and the allocation of tasks in the right panel.

Generally, given a set of tasks, the benefits of data sharing among those tasks are dominated by the task allocation as well as the schedule of sampling intervals. The time window of tasks may overlap when more than one task is allocated to a same sensor node. Accordingly, the sampling intervals of those tasks should be scheduled into the overlapping regions so as to maximize the gain of data sharing. Both the task allocation and the scheduling method of sampling intervals are tightly coupled. The strategy of task allocation has a great impact on the performance of the scheduling method sampling intervals. If the time window of sampling tasks allocated to a sensor node are not overlapping, the method of scheduling sampling intervals cannot exploit data sharing to reduce redundant data sampling. If the scheduling method sampling intervals is not optimal, each sensor node will still produce redundant sampled data. The goal of maximizing data sharing for the entire network cannot be realized. Therefore, exploiting the benefits of data sharing involves two challenges for WSNs. The first is the allocation strategy of sampling tasks over the entire WSN. The second is the schedule of sampling intervals for a set of tasks that have been allocated to a sensor node. To ease the presentation, we introduce Figure 1 as an illustrative example for the two subproblems:

—*Task allocation*: As demonstrated in Figure 1(b), consider three sensor nodes s_0, s_1, and s_2 in a WSN. Here, s_0 is the sink node 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 Figure 1(a). If t_1 and t_2 have been allocated to s_1 and s_2, respectively, then t_3 should be allocated to s_2, because the time window of t_3 and that of t_2 are overlapping and the sampling interval of t_3 and that of t_2 can be adjusted to implement maximal data sharing. Consider that extensive data-intensive applications generate a large set of sampling tasks. It is extremely difficult to allocate tasks so as to maximize data sharing with a resource-constrained sensor node.
—*Task scheduling*: As illustrated in Figure 1(a), consider three sampling tasks t_1, t_2, and t_3 that have been allocated to a same sensor node. Since the time window of t_2 and t_3 has a maximal overlapping time region, sampling intervals of t_2 and t_3 can be adjusted to achieve maximal data sharing when data is sampled in the overlapping time region. The problem of scheduling sampling intervals is NP-complete (see Corollary 3.14 in Section 3.4). Consider that a sensor node may be allocated

numerous sampling tasks in a monitoring system. It is difficult to solve the scheduling problem in polynomial time with a resource-constrained sensor node.

Although the scheduling problem of sampling intervals has been studied in Dalvi [2014], Tavakoli et al. [2010], and Fang et al. [2013], prior methods face at least two weaknesses. First, most existing work focuses on the discrete point sampling tasks. The continuous interval sampling tasks cannot benefit from these achievements directly. Second, existing work about the interval sampling tasks focuses only on the scheduling problem on a single node. The proposed effective scheduling method does not work well in WSNs due to lack of a task allocation procedure, primarly because the performance of a scheduling method is dominated by a strategy of task allocation at the scope of the entire network. We consider and formulate the two problems as a whole and provide a general and practical solution for a deployed system. To the best of our knowledge, this article is the first work that jointly optimizes both allocation and scheduling problems in a large-scale WSN.

In this article, we focus on maximizing the benefits of data sharing among sampling tasks, bearing the view of an entire WSN rather than a single sensor node. Our contributions are outlined as follows:

—We formulate the allocation and scheduling problems as a joint optimization problem under the k-coverage and r-redundant network model. The optimization problem is proved to be NP-hard.
—We present a crucial ingredient of our solution *COMBINE*, which aims to compute the minimal volume of sampled data for a set of overlapping tasks. The rigorous approximation bounds of *COMBINE* are presented theoretically.
—We propose our solution named *CATS*, which jointly allocates tasks and schedules the sampling intervals of the tasks on a sensor node to achieve maximal data sharing for WSNs. *CATS* is proved to be a 2-factor approximate algorithm theoretically.
—To evaluate the performance of our method, we conduct empirical study using a real testbed containing 50 nodes in the form of a 5×10 array. Large-scale simulations are further conducted to evaluate our methods via TOSSIM, a widely used simulation tool for WSNs. The evaluation results indicate that our method significantly reduces the amount of sampled data and energy consumption, and thereby improves the quality of communication in WSNs.

The rest of the article is organized as follows. Section 2 outlines related work and points out the difference between our method and previous work. Section 3 defines the basic models of a network and a sampling task, and then formalizes the optimization problem. Section 4 presents the crucial operation (i.e., *COMBINE*) and gives a rigorous approximation bounds theoretically. Section 5 proposes the solution (i.e., *CATS*) for the joint optimization problem. Section 6 evaluates the performance of our proposal through extensive experiments. Finally, Section 7 further discusses our solution, and Section 8 concludes this work.

2. RELATED WORK

2.1. Energy-Aware Task Allocation and Scheduling Mechanisms in WSNs

Xu et al. [2010] propose an energy-balanced method of task allocation that implements the maximal energy dissipation among all sensor nodes. The tasks are executed during the beginning of each epoch and must be completed before the end of the epoch (i.e., the epoch of a task is same as the time window of the task model in the article). It is noted that tasks in Xu et al. [2010] are communication tasks, not ampling tasks. The difference dominates that the proposed method in Xu et al. [2010] does not suit our

problem. Additionally, the solution for solving an integer linear programming problem in Xu et al. [2010] costs a great deal of time, which is prohibitive for a large system. Packets in Song et al. [2006] own the same properties of a sampling task that we discuss in this article. The release time and the deadline of a packet represent the begin and end time, respectively. The difference is that a packet reports one unit of data, not the data sampled over a time interval continuously. Thus, the methods in Song et al. [2006] work well for discrete point sampling tasks, not interval sampling tasks.

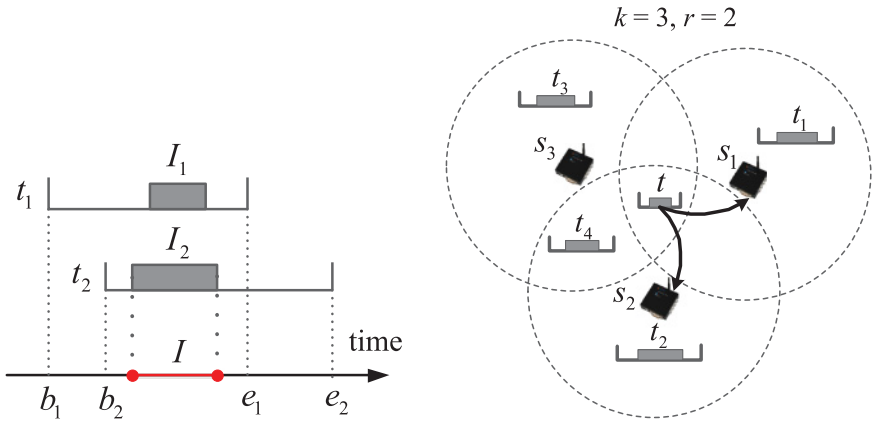
The most related works to ours are that of Tavakoli et al. [2010] and Fang et al. [2013]. Tavakoli et al. [2013] present an approach for task scheduling on a sensor node to minimize network communication overhead. A task in their work is a discrete point sampling task, that is, such a task requires a node to sample data once during its time window. However, the task defined in our problem requires a node to continuously sample data during an time interval. Therefore, their method does not suit our problem. Fang et al. [2013] propose an effective sampling approach for interval sampling tasks on a single sensor node. The 2-factor approximation algorithm in their work is a state-of-the-art method to maximize data sharing among tasks on a single node. Unfortunately, there are two weak points in this approach. First, the proposed scheduling method schedules sampling tasks in the descending order of the end time of a task, which, as a result, neglects data sharing between overlapping tasks. Moreover, the authors assume that all sampling intervals of tasks have a same length, 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 information by designing a crucial operation, namely *COMBINE*, which achieves better performance in maximizing 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 discussed, the performance of algorithms of task scheduling is sensitive to the strategy of task allocation. Our solution is more general and practical because of jointly optimizing the process of task allocation and that of scheduling sampling intervals.

2.2. Multiquery Optimization in a WSN

Recently, a WSN was treated as a database providing a good logical abstraction for sensor data management. Multiquery optimization in such a database system studies how to efficiently process queries [Xiang et al. 2007; Trigoni et al. 2005]. Xiang et al. [2007] adopts a two-tier multiple query optimization scheme to minimize the average transmission time in WSNs. The first-tier optimization is a cost-based approach that 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 data sharing among sampling tasks. The details of wireless communication is not involved in our method. Trigoni et al. [2005] consider multiquery optimization by using aggregation operations such as *sum* and *avg* to achieve optimal communication cost, whereas our method mainly concerns minimal energy consumption by reducing redundant sampled data.

2.3. Compression Technique–Based Data Collection

Compression technique–based data collection significantly reduces redundancy of sampled data for a sensor node [Arici et al. 2003; Pradhan et al. 2002]. Arici et al. [2002] propose an in-network compression scheme (PINCO) for a densely deployed WSN.



(a) Task model (b) Network model

Fig. 2. Task and network models for the interval data sampling.

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, Pradhan et al. [2002] present a framework on the distributed source coding technique to reduce data redundancy. However, the method of Pradhan et al. [2002] requires one to know the sensor correlation structure. Doing so was impossible for a randomly deployed system such as a battlefield monitoring system.

3. PRELIMINARIES

In this section, the basic task and network models are first formalized. We characterize the task allocation and the schedule of sampling intervals as a joint optimization problem. Then, we define a dedicated metric for measuring the data sharing. Finally, we prove that the joint optimization problem is NP-hard.

3.1. The Joint Optimization Problem

Definition 3.1 (*Interval Sampling Task*). An interval sampling task t is defined as a triple $<b, e, l>$, where b, e, and l represent begin time, end time, and length of the sampling interval of the task, respectively. The time window is denoted by $[b,\ e]$. The sampling interval can flexibly move in the time window. In other words, the sampling interval can select its beginning time in the time window as long as its end time does not exceed e. The point sampling tasks are the special case when such a point sampling task can be performed by sampling data only once during its time window.

As illustrated in Figure 2(a), there are two overlapping tasks $t_1 = <b_1, e_1, l_1>$ and $t_2 = <b_2, e_2, l_2>$. The sampling interval I satisfies the requirements of two tasks, but its interval length is smaller than the sum of the length of I_1 and I_2 due to the data sharing between t_1 and t_2.

Table I. Symbols List

Symbol	Notation	Symbol	Notation
t	Task	n	Cardinality of T
T	Task set	m	Cardinality of S
s	Sensor node	b	Begin time
S	Sensor node set	e	End time
I	Interval	$\lvert\cdot\rvert$	Cardinality of a set
Φ	Interval set	x_{ij}	Indicator variable
l, $\lvert I\rvert$	Interval length	$d(t_i, t_j)$	Value of data sharing
δ	Same length of interval used in a compact model		
φ	Number of compact tasks in a compact model		
t_{ij}	Novel task generated by combining t_i and t_j		
k	Task can be executed by k sensor nodes		
r	Task is actually performed by r sensor nodes		

Definition 3.2 (*k-Coverage and r-Redundant Network*). A WSN can be represented by $<S, T, k, r>$. Here, S and T denote the node set and the task set, respectively. k means that a sampling task in the network is detected by k sensor nodes, whereas only r of them are identified to execute the task.

For a k-coverage and r-redundant network, k is determined during the deployment of a WSN. Additionally, the setting of r can be adjusted according to the requirement of an application. In this article, we consider the case that k and r are determined after the deployment of a WSN. As illustrated in Figure 2(b), consider a WSN that consists of three sensor nodes s_1, s_2, and s_3. Each of them has a sensing range (indicated by the dotted circle). Although the task t_0 can be detected by three sensor nodes (i.e., $k = 3$), it is eventually assigned to two sensor nodes (i.e., $r = 2$) to guarantee the requirement of the application.

Before presenting the joint optimization problem, we first formalize the problem of task allocation and the problem of scheduling of sampling intervals. To ease the description, Table I outlines symbols frequently used throughout the article.

Definition 3.3 (*Overlap*). There are two sampling intervals I_1 and I_2. They are overlapping if and only if $I_1 \cap I_2 \neq \emptyset$. Similarly, tasks $t_1 = <b_1, e_1, l_1>$ and $t_2 = <b_2, e_2, l_2>$ are overlapping if and only if their time windows are overlapping. In other words, their time windows $[b_1, e_1]$ and $[b_2, e_2]$ have a common time region.

As illustrated in Figure 2(a), the tasks t_1 and t_2 are overlapping due to the common time region $[b_2, e_1]$. Similarly, their sampling intervals $I_1 = [u_1, v_1]$ and $I_2 = [u_2, v_2]$ are overlapping because of the common time region $[u_1, v_1]$. Assume that $u = \min\{u_1, u_2\}$, $v = \max\{v_1, v_2\}$, and $v_1 \geq u_2$; we define a novel interval I as the union of I_1 and I_2 (i.e., $I = I_1 \uplus I_2 = [u, v]$) and its length $l = v - u$. Here, "$\uplus$" stands for the union operation of two intervals.

Definition 3.4 (*The Allocation of Tasks*). Given a task t and a sensor node set S, the task set that has been allocated to the sensor node s, $s \in S$ is denoted by $T^s_{s \in S}$. The optimization problem of allocating t to one of sensor node in S can be defined as follows:

$$\min \left| \biguplus_{t_i \in \{t\} \bigcup T^s_{s \in S}} I_i \right| - \left| \biguplus_{t_i \in T^s_{s \in S}} I_i \right| \tag{1}$$

such that: $I_i \subseteq [b_i, e_i], i = 1, \ldots, n$;

Definition 3.5 (*The Schedule of Sampling Intervals*). Given a task set T with $|T| = n$, the optimization problem of scheduling sampling intervals of tasks in T is defined as follows:

$$\min \left| \biguplus_{t_i \in T} I_i \right| \tag{2}$$

such that: $I_i \subseteq [b_i, e_i], i = 1, \ldots, n$;

Definition 3.6. Given a sampling interval I and a $0-1$ indicator variable x_{ij}. If a task t_i is allocated to a node s_j, then $x_{ij} = 1$, else $x_{ij} = 0$. We define $x_{ij} \odot I$ as follows:

$$x_{ij} \odot I = \begin{cases} I & : \quad x_{ij} = 1 \\ \emptyset & : \quad x_{ij} = 0 \end{cases}. \tag{3}$$

Definition 3.7 (*The Joint Optimization Problem*). Given a task set T with $|T| = n$ and a sensor node set S with $|S| = m$, a task $t_i, t_i \in T$, is notated by 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 defined as follows:

$$\min \sum_{j=1}^{m} \left| \biguplus_{i=1}^{n} x_{ij} \odot I_{ij} \right| \tag{4}$$

such that:

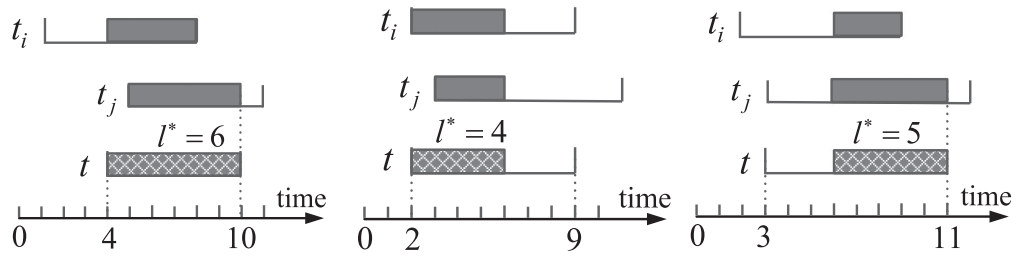
$$\begin{cases} \sum_{j=1}^{m} x_{ij} = r, i = 1, \ldots, n, 1 \le r \le k; \\ x_{ij} = 0 \text{ or } 1; \\ I_{ij} \subseteq [b_i, e_i], i = 1, \ldots, n; \end{cases} \tag{5}$$

When $r = k$, we should allocate each task to all candidate sensor nodes. The solution of task allocation is determined solely. Generally, when $0 < r < k$, the object function is nonlinear, which makes the optimization problem become difficult. This kind of nonlinear programming problem has no universal efficient solution. Several methods, including branch and bound techniques, require high computational complexity and are not applied to a WSN. Considering the limitation of computing and memory resources of sensor nodes, the nonlinear programming problem becomes difficult to solve. To ease the description, Table I summarizes the major symbols in this paper.

3.2. Measurement of Data Sharing Between Overlapping Tasks

Data sharing among tasks is the foundation of our solution. We aim to minimize the total amount of sampled data by exploiting data sharing among them. Before presenting our method, we first demonstrate some basic definitions that are used to measure data sharing among overlapping tasks.

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 $l_i^* \ge l_j$, and t_j satisfies t_i if and only if $l_j^* \ge l_i$.



(a) t_i and t_j do not satisfy each other (b) t_i satisfies t_j (c) t_j satisfies t_i

Fig. 3. Three cases: the combination of overlapping tasks.

Definition 3.9 (*Data Sharing*). Let $d(t_i, t_j)$ denote the maximal value of data sharing of two tasks t_i and t_j. Without loss of generality, assume that $e_i \leq e_j$, then

$$d(t_i, t_j) = \begin{cases} e_i - b_j & : \ t_i, t_j \text{ cannot satisfy each other.} \\ l_j & : \ t_i \text{ satisfies } t_j. \\ l_i & : \ t_j \text{ satisfies } t_i. \end{cases} \tag{6}$$

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 that $e_i \leq e_j$. Then a time window $[b^*, \ e^*]$ can be constructed as follows:

—If t_i and t_j cannot satisfy each other, then

$$\begin{cases} b^* = e_i - l_i \\ e^* = b_j + l_j \end{cases}. \tag{7}$$

—If t_i satisfies t_j, then

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

—If t_j satisfies t_i, then

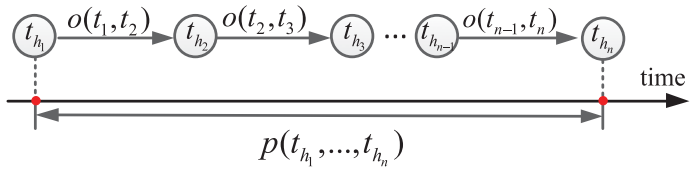
$$\begin{cases} b^* = \max\{b_j, b - (l_j - l_i)\} & b = \max\{b_i, b_j\} \\ e^* = \min\{e_j, e + (l_j - l_i)\} & e = \min\{e_i, e_j\} \end{cases}. \tag{9}$$

Definition 3.11 (*Combination*). A novel task t_{ij} is the combination of overlapping tasks t_i and t_j. For simplicity, t_{ij} is denoted by $t^* = <b^*, e^*, l^*>$, which is called the *child task*, whereas t_i and t_j are called *father tasks*. Here, both b^* and e^* are computed according to Definition 3.10. $l^* = l_i + l_j - d(t_i, t_j)$.

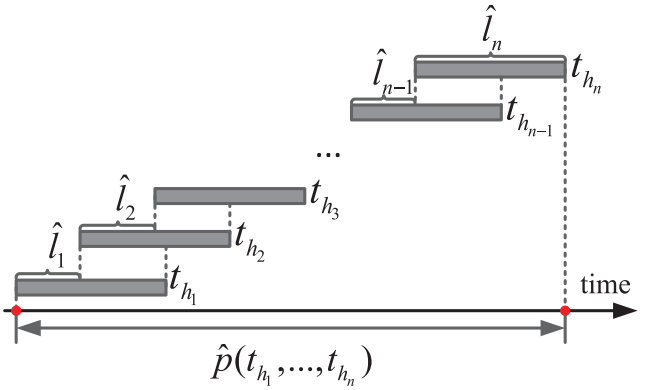
As illustrated in Figure 3(a), two tasks $t_i = <1, 8, 4>$ and $t_j = <5, 11, 5>$ do not satisfy each other. $d(t_i, t_j) = 3$. $b^* = 4$, $e^* = 10$. After a combination of them, a novel task t_{ij} is generated and $t = <4, 10, 6>$. In Figure 3(b), two tasks $t_i = <2, 9, 4>$ and $t_j = <3, 11, 3>$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

3.3. The Complexity Analysis of the Joint Optimization Problem

To make it clear, we first introduce the definition of maximum directed Hamilton path (MAX-DHP), which has been proved to be an NP-complete problem [Du et al. 2012].



(a) Construct a graph G in which a vertex stands for a task and weight of an edge represents the value of data sharing



(b) The computation of MAX-DHP in graph G

Fig. 4. MAX-DHP is transformed to the computation of a minimal volume of sampled data.

We then prove that the optimization problem is NP-hard by the transformation from MAX-DHP.

Definition 3.12 (*MAX-DHP*). Given a complete directed graph and a distance function, find a Hamiltonian path with the maximum total distance. (*A Hamiltonian path is a simple path that passes through each vertex exactly once.*)

LEMMA 3.13. *Given a sampling task set T with $|T| > 2$, the computation of the minimal volume of sampled data is NP-complete.*

PROOF. Assume that 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 $i = 1, 2, \ldots, |\Phi|$.

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

Second, we construct a directed Hamilton graph, which is used to transform MAX-DHP to the optimization problem. 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 data sharing (i.e., $d(t_1, t_2)$). If they are not overlapping, then the weight of the 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 data sharing between t_i and other tasks. As illustrated in Figure 4(a), iteratively repeat the step until all tasks have been checked and the directional graph G is constructed.

—*Step 2*: Given a maximal Hamilton path $t_{h_1}, t_{h_2}, \ldots, t_{h_n}$ in graph G, the length of the directed Hamilton path $p(t_{h_1}, \ldots, t_{h_n})$ in Figure 4(a) is calculated as follows:

$$\begin{aligned} & p(t_{h_1}, \ldots, t_{h_n}) \\ & = \sum_{i=1}^{n-1} d(t_{h_i}, t_{h_{i+1}}) \\ & = p(t_{h_1}, \ldots, t_{h_{n-1}}) + (l_{t_{h_n}} - \widehat{l}_{t_{h_n}}) \\ & = \cdots \\ & = \sum_{i=1}^{n} l_{t_{h_i}} - \sum_{i=1}^{n} \widehat{l}_{t_{h_i}} \\ & = \sum_{i=1}^{n} l_{t_{h_i}} - \widehat{p}(t_{h_1}, \ldots, t_{h_n}). \end{aligned} \tag{10}$$

As illustrated in Figure 4(b), $\widehat{p}(t_{h_1}, \ldots, t_{h_n})$ is the total amount of sampled data for the task set. So the MAX-DHP problem can be transformed to the optimization problem.

As the entire process is completed in polynomial time, Lemma 3.13 is proved. □

COROLLARY 3.14. *Given a sampling task set, the problem of scheduling sampling intervals of tasks is NP-complete.*

THEOREM 3.15. *When $0 < r < k$, the joint optimization problem 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—for example, $r = 1$, and $k = 2$. In this situation, the problem of computing minimal volume of sampled data can be transformed to the joint optimization problem.

Given a task set T with $T \neq \emptyset$, if there is an optimal solution to compute the volume of sampled data and return an interval set Φ, then divide Φ into two subsets Φ_1 and Φ_2 such that each satisfies T_1 and T_2 ($T = T_1 \cup T_2$), respectively. The procedure can be finished in polynomial time. However, the computation of the minimal volume of sampled data, as proved in Lemma 3.13, is NP-complete. Thus, the special case of the joint optimization problem is NP-hard. In general, this joint optimization problem is at least as difficult as the special case. Therefore, when $0 < r < k$, the joint optimization problem is NP-hard. □

4. COMBINE: MAXIMIZING DATA SHARING BETWEEN OVERLAPPING TASKS

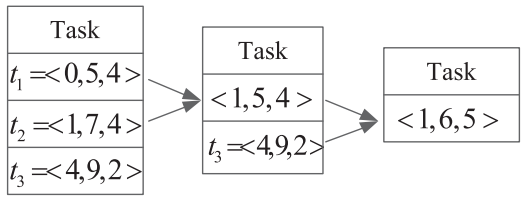
In this section, we first present the *COMBINE* algorithm, which maximizes data sharing among overlapping tasks. We then prove the bound of the algorithm by theoretical analysis rigorously. *COMBINE* is the crucial ingredient of our solution.

Let T denote a nonempty task set. If $|T| = 1$, the amount of sampled data is determined solely. If $|T| > 1$, a quad $q = <t_i, t_j, t_{ij}, d(t_i, t_j)>$ is maintained for the combination of the overlapping tasks t_i and t_j. Here, t_i and t_j are father tasks, and the child task is t_{ij}. $d(t_i, t_j)$ is the value of data sharing between t_i and t_j.

Our basic idea is to schedule all tasks in T by iteratively combining the task pair that has the maximal value of data sharing until there are no overlapping tasks in T. The whole procedure includes three major steps. First, it computes the combination of all overlapping tasks and gets a quad list Q. Second, a quad $q \in Q$, $q = <t_i, t_j, t_{ij}, d(t_i, t_j)>$ where $d(t_i, t_j)$ is maximal is found. Third, original tasks t_i and t_j are replaced by the novel task t_{ij} in the task set T. Fourth, repeat the preceding steps until no overlapping tasks exist. Algorithm 1 presents details and returns a task set that has no overlapping sampling tasks.

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. Process of *COMBINE* for overlapping tasks.

ALGORITHM 1: COMBINE(T)

Require: A task set T and $T \neq \emptyset$. A quad list Q and $Q = \emptyset$.
1: sort tasks in T in the descending order of end time.
2: $Q \leftarrow$ PRE_PROCESSING(T).
3: sort quads in Q in the descending order of value of data sharing.
4: **while** $|T| > 1$ and T contains overlapping tasks **do**
5: $t_i = Q(0).t_i$, $t_j = Q(0).t_j$, $t_{ij} = Q(0).t_{ij}$.
6: insert t_{ij} into T in the order of end time.
7: remove t_i and t_j from T and quads that include t_i or t_j from Q.
return T.
8: **function** PRE_PROCESSING(T)
9: **for** each task t_i in T **do**
10: **for** each task $t_j \in T - \{t_i\}$ **do**
11: **if** t_i is overlapping with t_j **then**
12: a quad q is generated with $q = <t_i, t_j, t_{ij}, d(t_i, t_j)>$.
13: $Q \leftarrow Q \bigcup \{q\}$.
return Q.

Algorithm 1 performs an iterative operation. Lines 1 through 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 maintain the quad list. The time complexity is $O(n^2)$ due to the computation of maximal value of data sharing. Lines 4 through 7 find a task pair that has the maximal value of data sharing in the task set T and then update T and Q. As shown in Figure 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 length of the overlapping intervals 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 novel task $t_{12} = <1, 5, 4>$. Second, remove t_1 and t_2 from T and add t_{12} to T. We get a novel task (i.e., $<1, 6, 5>$) after combining t_{12} and t_3. Thus, 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 rare for a WSN. 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 as Algorithm 1 but significantly reduces the memory consumption. Algorithm 1 is proved to be a 2-factor approximation algorithm by Theorem 4.5. Before presenting Theorem 4.5, we first present Lemma 4.1 and Property 4.2, which will be used in the proof of Theorem 4.5.

ALGORITHM 2: COMBINE_2(T)

Require: A task set T and $T \neq \emptyset$.
1: sort tasks in T in the descending order of end time.
2: **while** $|T| > 1$ **do**
3: find a task pair $<t_i, t_j>$ that has the maximal value of data sharing.
4: **if** $d(t_i, t_j)==0$ **then**
5: return T.
6: insert a novel task t_{ij} into T in the descending order of end time.
7: remove t_i and t_j from T.
8: return T.

LEMMA 4.1. *For a nonempty task set T, if we select two tasks t_i and t_j by using Algorithm 1, a novel task t_{ij} is generated. For another task t_k and $t_k \in T$, we have $d(t_i, t_j) \geq \max\{d(t_i, t_k), d(t_j, t_k)\}$ and $d(t_{ij}, t_k) \leq \min\{d(t_i, t_k), d(t_j, t_k)\}$.*

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

First, there exists a task notated by 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 data sharing between tasks t_i and t_j is not the maximal. It is a contradiction because we compute the current maximal value of data sharing by Algorithm 1. Thus, we have $d(t_i, t_j) \geq d(t_i, t_k)$.

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

LEMMA 4.2. *The value of data sharing between two overlapping tasks, according to Algorithm 1, is monotone nonincreasing.*

PROOF. Consider a nonempty task set T and the overlapping tasks t_i and t_j. $\forall t_k \in T$, $d(t_i, t_j) \geq d(t_{ij}, t_k)$ is always established according to Lemma 4.1. Then we can prove Property 4.2 by using the method of mathematical induction, as we identify a task pair with the maximal value of data sharing from the task set T in each combination step. If t_i and t_j are the current choice, it means that other task pairs cannot provide more data sharing than t_i and t_j. 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, Property 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

$$\begin{cases} \widetilde{b} = b_i \\ \widetilde{e} = e_i \\ \widetilde{l} = e_i - b_i \end{cases}. \tag{11}$$

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

$$\begin{cases} \delta \leq \min\{l_1, l_2, \ldots, l_n\},\ \delta \textit{ is a positive constant}. \\ \varphi = \min\{x | x \cdot \delta \geq \sum_{i=1}^{n} l_i, x \textit{ is a positive integer}\}. \end{cases} \tag{12}$$

THEOREM 4.5. *Algorithm 1 is a 2-factor approximation algorithm to compute the minimal volume of sampled data.*

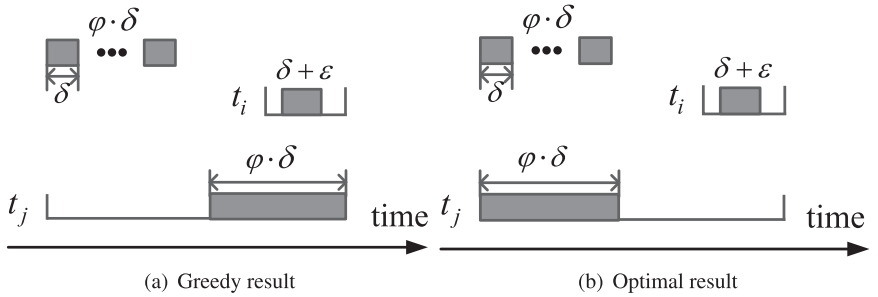


Fig. 6. Typical example of the approximate result by using Algorithm 1 versus the optimal result.

PROOF. For a nonempty task set T, if $|T| = 1$, Algorithm 1 returns the only one sampling task that is the optimal result.

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

$$\begin{aligned} \frac{GREEDY(T)}{OPT(T)} &= \frac{|I_i \uplus I_j \uplus I^{T'}|}{|I_i \uplus I^{T'}|} \\ &= \frac{\varphi \cdot \delta + |I_i \uplus I_j|}{\varphi \cdot \delta + |I_i|} \\ &\leq 1 + \frac{|I_j|}{\varphi \cdot \delta + |I_i|} \\ &\leq \frac{1}{2} + \frac{\varphi \cdot \delta}{\varphi \cdot \delta + \delta} \end{aligned} \tag{13}$$

Here, $\delta \leq d(t_i, t_j)$. □

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

$$\lim_{\varepsilon \to 0, \varphi \to \infty} \frac{\varphi \cdot \delta + \varphi \cdot \delta}{\varphi \cdot \delta + \delta + \varepsilon} = 2. \tag{14}$$

Our algorithm performs better when a task is overlapping with others tightly. Empirical study in Section 6 has verified this conclusion.

5. APPROXIMATION ALGORITHMS FOR THE JOINT OPTIMIZATION PROBLEM IN WSNS

In this section, three algorithms are presented for the joint optimization problem. The first is a random algorithm that allocates the sampling tasks randomly. The second is a pruning algorithm that first allocates a task to all candidate nodes and then removes it from some of the candidate sensor nodes by the value of data sharing. The third is a 2-factor approximation algorithm that computes the volume of sampled data by iteratively combining overlapping tasks.

For a task set, the insight of the random method is to randomly identify r out of k candidate sensor nodes. This method is simple and easily performed on a sensor node. However, it neglects data sharing among tasks and brings a wealth of unnecessary

ALGORITHM 3: PRUNE(T, S, k, r)

Require: A task set T and $T \neq \emptyset$. A sensor node set S and $S \neq \emptyset$. $1 \leq i \leq m$. $r \leq k$.

1: **while** $T \neq \emptyset$ **do**
2: allocate a task t, $t \in T$ to k candidate sensor nodes.
3: compute the value of data sharing $d(t, \cdot)$ between t and any other tasks on a sensor node.
4: **if** no sampling task is overlapping with t **then**
5: randomly allocate t to one candidate sensor node.
6: **else**
7: remove the task t from a candidate sensor node when t has the smallest value of data sharing with other tasks on the node.
8: $t.count = t.count - 1$.
9: **if** $t.count == r$ **then**
10: remove t from T.
11: **COMBINE**(T_i) for each task subset T_i.

sampled data. This method consumes much energy of a sensor node and damages the quality of communication in WSNs as well.

5.1. Maximizing Data Sharing of Tasks According to the Pruning Method

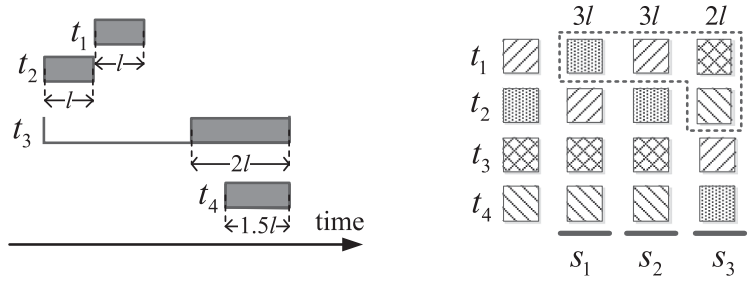
We notice that the sampled data can be shared in the overlapping time region. Therefore, we consider allocating tasks via the pruning operation, which removes unreasonable allocation choices repeatedly. The entire process contains three major steps. First, we allocate tasks to all candidate sensor nodes. In other words, if a task is detected by k sensor nodes, it is allocated to the k candidate sensor nodes. Second, compute the maximal value of data sharing between a task and other overlapping tasks. Then remove a task from a sensor node where the task has the smallest value of data sharing 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 1 through 3 allocate a task to its k candidate sensor nodes. Lines 4 through 10 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 volume of 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 $O(n)$ because each node does not have to maintain the data sharing between any pair of overlapping tasks.

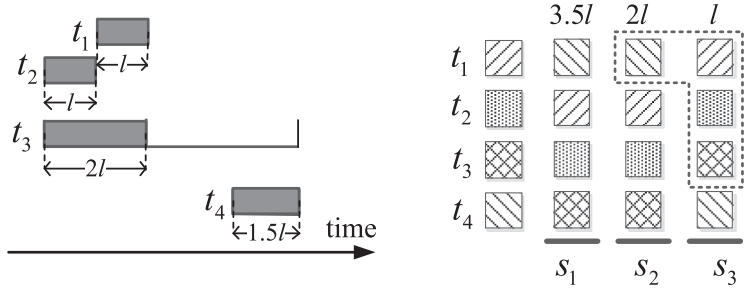
Algorithm 3 is a greedy algorithm. A typical example is shown in Figure 7. Here, $k = 3$ and $r = 2$. The sensor node set is notated by S, and $S = \{s_1, s_2, s_3\}$. The task set is notated by T, and $T = \{t_1, t_2, t_3, t_4\}$. As illustrated in Figure 7, the left panel demonstrates the computation of data sharing by Algorithm 3 and the optimal method. Four rectangles in the right panel stand for the tasks in the left panel. The pruning method first allocates all sampling tasks to candidate nodes and then removes the tasks that have the smallest volume of data sharing with other tasks (indicated in the dotted polygon). Finally, Algorithm 3 returns the total length of the sampling intervals, $8l$ as shown in Figure 7(a), whereas the optimal result is $6.5l$ as shown in Figure 7(b), because our pruning procedure is not optimal in each round. This motivates us to propose Algorithm 4, which allocates tasks according to the *COMBINE* operation and can be optimal in each round.

5.2. CATS: Maximizing Data Sharing of Tasks According to the *COMBINE* Operation

The *COMBINE* operation schedules sampling tasks for maximizing data sharing in each round. Since it has a good performance, we provide a solution that allocates



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



(b) The optimal result of task allocation

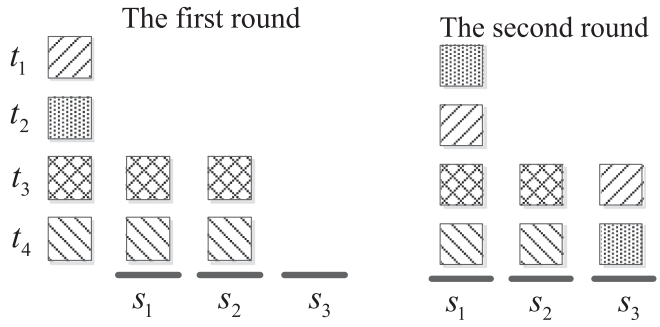
Fig. 7. Typical example of the approximate result by using Algorithm 3 versus the optimal result. Four tasks t_1, t_2, t_3, and t_4 in the left panel are represented by four rectangles in the right panel, respectively. Tasks in the dotted polygon will be removed from nodes due to the pruning process.

ALGORITHM 4: CATS(T, S, k, r)

Require: A task set T and $T \neq \emptyset$. A sensor node set S and $S \neq \emptyset$. $r \leq k$.

```
1: while there exists overlapping tasks in T do
2:     identify a task pair <t1, t2> from T that has the maximal value of data sharing.
3:     combine t1 and t2 and notate the resultant task by t12.
4:     remove t1 and t2 from T.
5:     add t12 into T.
6: while T is not an empty set do
7:     randomly identify a task t, t ∈ T.
8:     if t is not an original task then
9:         identify the original tasks that generate t after the process of COMBINE, and allocate
       them to r sensor nodes.
10:    else
11:        allocate t to r sensor nodes.
```

tasks and computes the volume of sampled data by using the *COMBINE* operation. As illustrated in Figure 2, a task t_0 is detected by three sensor nodes, but only two of them are identified to perform it. Assume that task t_0 is overlapping with three tasks t_1, t_2, and t_3 such that $d(t_0, t_1) > d(t_0, t_2) > d(t_0, t_3)$, and t_1, t_2, and t_3 have already been allocated to s_1, s_2, and s_3, respectively. When $k = 3$ and $r = 2$, we should allocate t_0 to the sensor nodes s_1 and s_2. The method includes three major steps. First, maintain a global quad list in which each quad stands for a *COMBINE* operation of two overlapping tasks. Second, iteratively identify a quad that has the maximal value of data sharing of overlapping tasks in the global quad list. Then allocate the overlapping tasks to r out of k candidate sensor nodes. Finally, update the quad list and the task set. Algorithm 4 presents the method in detail. In Algorithm 4, lines 1 through 5 present the combination



(a) The first round of task combination by using Algorithm 4

(b) The second round of task combination by using Algorithm 4

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

of the overlapping tasks. Lines 6 through 11 ensure that all tasks are allocated to r different sensor nodes even though some tasks are not overlapping with others. The time complexity is $O(n^2)$, and the space complexity is $O(n)$.

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 of 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 Figure 8(a), tasks t_3 and t_4 should be allocated to sensor nodes. A novel task $t_{34} = <4, 8, 4>$ is generated. We further remove tasks t_3 and t_4 from T_1, then add t_{34} into T_1. After t_3 and t_4 are allocated, all 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 Figure 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.

THEOREM 5.1. *Algorithm 4 is a 2-factor approximation algorithm for computation of the volume of sampled data.*

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 $1 \leq r < k$, 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 that may be allocated to a same node.
—For any two tasks that are not allocated to a same node, we set its 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.

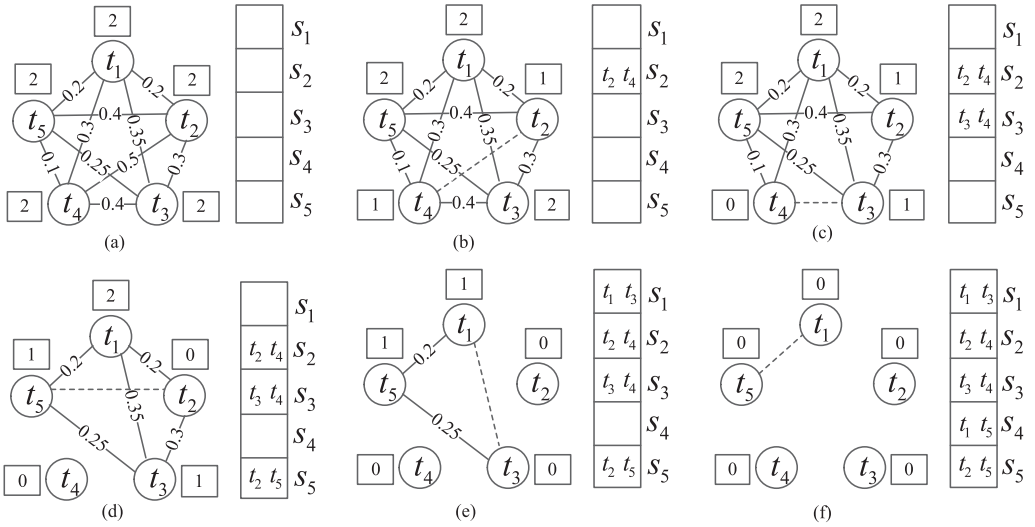


Fig. 9. Illustrative example of transformation from Algorithm 1 to Algorithm 4 by combining the overlapping tasks that have the maximal value of data sharing in each step. Here, $k = 3$ and $r = 2$.

Algorithm 1 is then performed on the task set until there are 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 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 provide an example to illustrate the process of transformation. As shown in Figure 9, assume that there are five tasks $t_1, \ldots, t_5$ (as indicated by cycle) that should be allocated to five nodes $s_1, \ldots, s_5$. $\{t_1, t_2, t_3\}$ 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 Figure 9(a), we construct a weighted graph where 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 to which the task has been allocated. The overlapping tasks that 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, the task is removed from the task set. Without loss of generality, the novel tasks generated from the *COMBINE* operation are not demonstrated. For example, in Figure 9(b), we find that tasks t_2 and t_4 have the maximal value of data sharing: 0.5. Then, they are allocated to node s_2, and the data sharing between them is set to negative infinity. As illustrated in Figure 9(d), when task t_4 has been allocated to s_2 and s_3, the edges between it and other nodes are removed. When the sampling intervals 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 novel tasks, novel vertices will added to the graph, but this will not impact the performance of Algorithm 4.

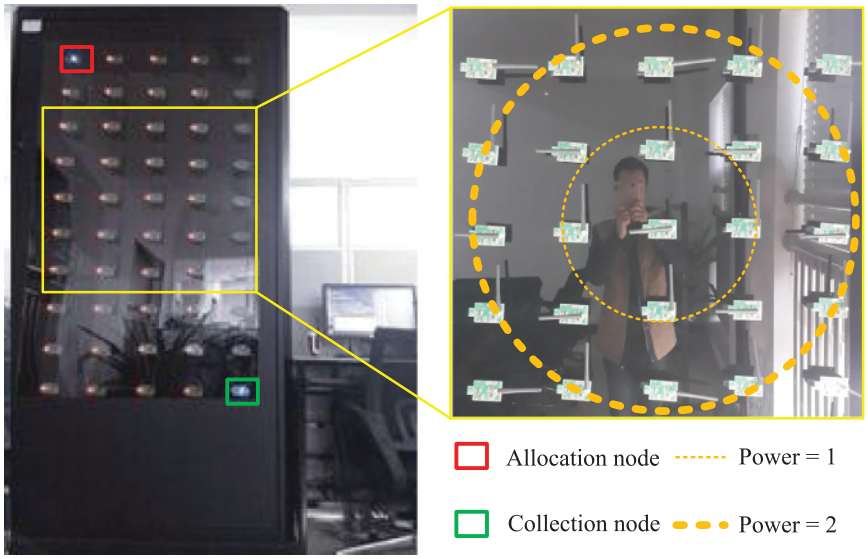


Fig. 10. Output power can be adjusted to realize the k-coverage network in the testbed.

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 WSNs. Finally, TOSSIM, the widely used simulation tool, is used to verify the scalability of our method.

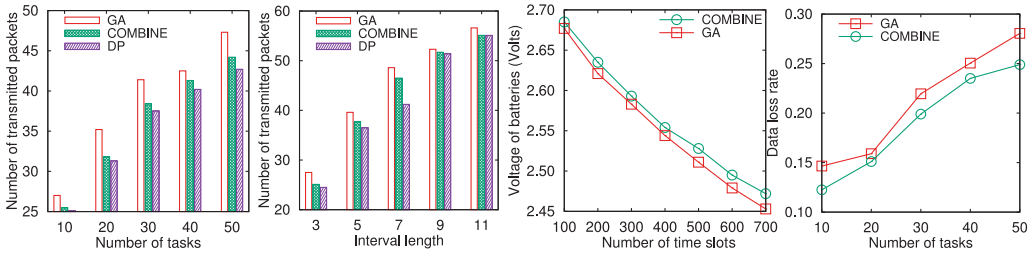
6.1. Experimental Environment and Settings

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

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 that 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 rectangle in Figure 10) as the sink node that allocates a task set to other nodes and the right-bottom node as the collection node (as indicated by the green rectangle in Figure 10) that collects all sampled data. Transmitted packets are counted to indicate the required sampled data, whereas received packets are collected to demonstrate the actual collected data.

The value of k varies 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



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

Fig. 11. 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 slots in (c) and data loss rate in (d).

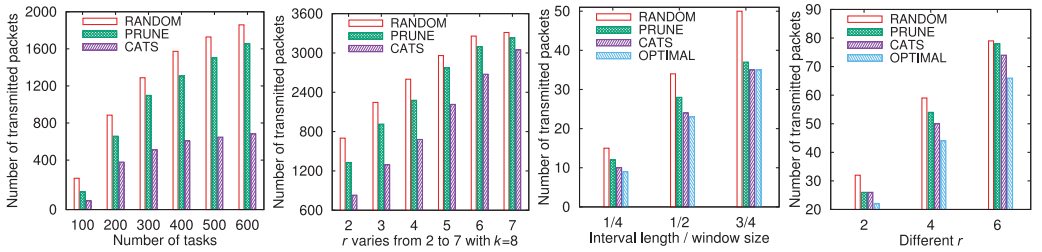
continuous data by sampling temperature 5 times per second and send the data by using a packet. The interval length of a sampling task is randomly generated and not greater than 10. To be specific, the begin time of a sampling task is evenly distributed in the time slot [0, 50]. We run each algorithm 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 on a Sensor Node

First, we evaluate the performance of our Algorithm 1, denoted by *COMBINE*, against the state-of-the-art method [Fang et al. 2013], named *GA* here. *GA* is an approximation algorithm for computing the amount of sampled data on a single sensor node. Meanwhile, *GA* can derive the optimal scheduling algorithm, denoted by *DP*, using the 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.

Figure 11(a) and (b) illustrate the number of transmitted packets when using *COMBINE*, *GA*, and *DP* by varying the cardinality of a sampling task set (Figure 11(a)) or the interval length of a sampling task (Figure 11(b)). Specifically, the interval length of a sampling task is set to 5 in Figure 11(a), whereas the cardinality of each task set is fixed to 30 in Figure 11(b). It is easy to observe from Figure 11(a) and (b) that the number of transmitted packets increases with both the growth of the number of tasks and the interval length. However, *COMBINE* performs better and returns a smaller number of transmitted packets than *GA*. This happens because our algorithm combines overlapping tasks that have the maximal value of data sharing for each step. Thus, each step is the current optimal choice. Considering that *GA* schedules the sampling intervals of tasks based on the end time of a task, it cannot ensure that each scheduling choice is optimal. This is the reason *COMBINE* outperforms *GA*. Meanwhile, Figure 11(a) and (b) indicate that *COMBINE* achieves a 2-factor approximation result versus the optimal result. This verifies the correctness of Theorem 4.5.

In Figure 11(c), we run the *COMBINE* 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 initial value is 2.864V. It is apparent that while the terminal battery voltage decreases due to the energy consumption, *COMBINE* consumes less energy than *GA*. This is because *COMBINE* reduces more unnecessary sampled data than *GA*. Since a sensor node uses up much more energy when listening, receiving, and sending data, *COMBINE* greatly cuts down energy consumption and prolongs the lifetime of the entire WSN. Precisely, *COMBINE* decreases energy consumption by 4.85% per slot on average. In Figure 11(d), we test the data loss rate



(a) k=5, r=2 (b) k=8, n=30 (c) k=5, r=2, n=5 (d) k=8, n=5

Fig. 12. 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).

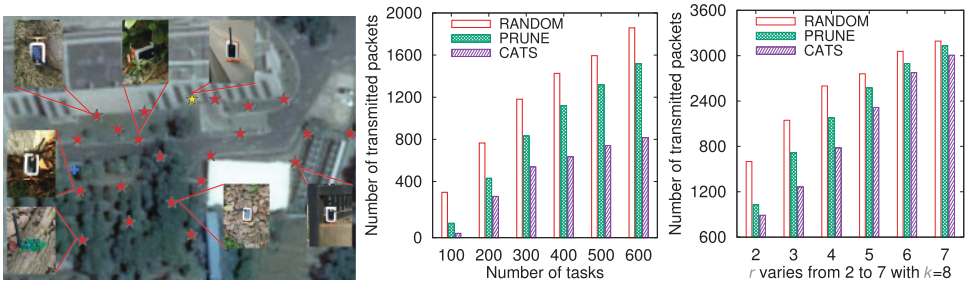
of different methods during data transmission by changing the number of tasks on a single sensor node. We observe that the data loss rate increases with the growth of the number of tasks. We are delighted to see that *COMBINE* achieves a smaller data loss rate than *GA*. Precisely, *COMBINE* decreases energy consumption by 4% per slot on average. Here, a time slot is a period of 50 seconds. Since a wireless sensor node works for several months, such improvements are appreciable and worthy of being exploited. Moreover, *GA* is specifically designed for the schedule of sampling intervals. It is unknown how to use *GA* for task allocation. Adopting a random strategy of task allocation for 300 tasks in a k-coverage and r-redundant network with $k = 5$ and $r = 2$, *GA* consumes more energy and leads to a higher data loss rate than our solution by more than 35% and 30%, respectively, to poorly exploiting data sharing among tasks. Such benefit becomes more obvious when the number of tasks increases. The reduction of redundant sampled data relieves the workload of intranetwork communication and decreases transmission delay and congestion by using *COMBINE*.

6.3. Performance of Data Sharing Across WSNs

In this section, we verify the performance of the naive method which allocates tasks randomly, Algorithm 3 and Algorithm 4 which are represented by *RANDOM*, *PRUNE*, and *CATS*, respectively hereinafter.

In Figure 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. Figure 12(a) shows that the number of transmitted packets increases with the number of tasks. Moreover, *PRUNE* and *COMBINE* significantly decrease unnecessary sampled data more so 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 *COMBINE* seems to be half of that brought about by *RANDOM*. In Figure 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 *CATS* reduce more transmitted packets than *RANDOM*. Another observation is that the advantage of *PRUNE* and *CATS* becomes less significant with the growth of r, as 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 solution that shares the same performance with *PRUNE* and *CATS*.

It is difficult to derive an effective optimal solution for the joint optimization problem. However, when the scale of tasks is small, we can find the optimal result by using a brute-force method. To evaluate the performance of our method rigorously, we compare the number of transmitted packets on each method by varying the number of tasks and the value of r. In Figure 12(c), we display three groups of tasks, each of which has



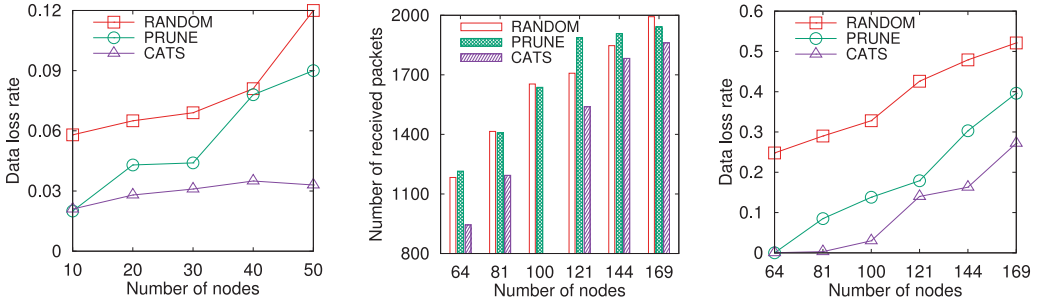
(a) The network built in a general topology (b) k=5, r=2, general topology (c) k=5, n=30, general topology

Fig. 13. WSN with a general topology is deployed by using 22 nodes. The yellow star represents the collection node, whereas the red stars represent other nodes in the network; some of their enlarged pictures are shown as examples. The comparison of sampled data is shown by adopting different strategies of data sampling when the number of tasks varies in (a) and r varies in (b).

five tasks. The interval length of a task is set to 1/4, 1/2, and 3/4 of its window size in groups 1, 2, and 3, respectively. These tasks appear in time slot [0, 20] randomly. Here, $k = 5, r = 2$. The optimal method is denoted by *OPTIMAL*. It is clear that the number of transmitted packets increases with the expansion of the interval length. *PRUNE* and *CATS* perform better than *RANDOM* and are close 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 Figure 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 a condition, *CATS* achieves a greatly smaller number of transmitted packets than twice of that produced by the optimal solution. This verifies the conclusion of Theorem 5.1 again.

Additionally, we deploy a real WSN that consists of 22 nodes and is built in a general topology. As illustrated in Figure 13(a), the yellow star represents the collection node, and the red stars represent other nodes. Some enlarged pictures of those red stars are shown as examples. The sampling tasks are produced under the settings of Section 6.1. Moreover, the output power of each a sensor node is adjusted to be a high level to build a coverage network. As illustrated in Figure 13(b) and (c), we compare the amount of sampled data to evaluate the performance of different strategies of data sampling. It is obvious that *CATS* and *PRUNE* perform better than *RANDOM* when we vary the number of sampling tasks or the length of a sampling interval for a task. This is noted primarily because they reduce unnecessary sampled data by adopting effective data sharing strategies. It is noting that the real WSN is built in a more general topology than the grid topology we use in the testbed. The result thus shows that our strategies of task allocation and scheduling of sampling intervals (i.e., *CATS* and *PRUNE*) are effective to reduce the unnecessary sampled data.

In addition, a large amount of sampled data definitely leads to server delay and congestion in WSNs, degrading the quality of data transmission as a result. In Figure 14(a), we compare the data loss rate of different methods 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 *CATS* significantly derive the smaller data loss rate than *RANDOM*. This verifies the benefits of our solution on the improvement of the quality of data transmission by cutting down the unnecessary sampled data. As well, we conduct simulations for evaluating the scalability of our proposed algorithms for a large-scale WSN. These simulations are implemented with TOSSIM, which is a



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

Fig. 14. Comparison of the data loss rate in the testbed is presented by varying the number of nodes in (a). In addition, the number of received packets is compared by varying the number of nodes in (a), as well as the data loss rate in (b), for large-scale WSNs simulated in TOSSIM.

Parameter	Value	Notation
PATH_LOSS_EXPONENT	4.7	Rate at which signal decays
SHADOWING_STANDARD_DEVIATION	3.2	Randomness of received signal due to multipath
D0	1.0	Minimum distance allowed between any pair of nodes
PL_D0	55.4	Power decay in dB for the reference distance *D0*
NOISE_FLOOR	−105.0	Radio noise floor in dBm
S11	0	Variance of noise floor
S22	0	Variance of output power
WHITE_GAUSSIAN_NOISE	4	Standard deviation of additive white Gaussian noise
TOPOLOGY	1	Nodes are placed on a square topology
GRID_UNIT	2.0	Internode distance of the grid
NUMBER_OF_NODES	100	Number of nodes in the grid WSNs

Fig. 15. Parameter settings in the simulations by using TOSSIM. The left column shows the parameters that can be used to configure a grid WSN, the middle column is the value, and the right column is the notions of these parameters.

widely used simulation tool for WSNs [Levis et al. 2003]. We construct a grid network, and the settings of the simulated network are listed in Figure 15, which are widely accepted [Sohrabi et al. 1999]. As illustrated in Figure 14(b), we compare the number of received data by varying the scale of the network. We observe that *CATS* brings the smallest amount of received data when the scale of the network grows, as *CATS* reduces more unnecessary sampled data than *PRUNE* and *RANDOM*. We note that *PRUNE* brings more received data than *RANDOM* in a large wireless sensor network, which includes more than 100 wireless sensor nodes, as *RANDOM* causes more severe loss of data than *PRUNE* when the scale of the network is large. Although *RANDOM* produces the largest volume of sampled data, the received data by using *RANDOM* may be smaller than *PRUNE* because of data loss. As illustrated in Figure 14(c), the data loss rate in both *CATS* and *PRUNE* is much smaller than that in *RANDOM*. This happens when the number of transmitted data in *CATS* and *PRUNE* is smaller than that in *RANDOM*. In summary, *CATS* is more suitable to be deployed for a large-scale WSN, as it always performs much better than *RANDOM*.

7. DISCUSSION

We have proposed methods of task allocation and scheduling of sampling intervals and have given much theoretical analysis about their performance. As the question is general, we aim to provide a universal solution that does not rely on the details of the network. The sink node runs *CATS* and gets the strategy of task allocation in a WSN. 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. Compared to the volume of sampled data of sampling tasks, such expenditure on the allocation solution is small. 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 trade few expenditures for appreciable data sharing [Liu et al. 2012, 2013]. Additionally, 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. We aim to propose an effective solution for the joint optimization problem.

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 among tasks. In reality, there still exist 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 tasks in a timely manner and may exhaust the energy early. This problem cannot thus be simply addressed by limiting the number of tasks a sensor node is carrying, because although different nodes have the same amount of sampling tasks, the real workload 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 among sensor nodes. An effective method is to set a threshold for the constrained resource. Each allocation step ensures that the threshold value is not exceeded. Algorithms 3 and 4 are flexible to adjust for this tactic. When the limited resource changes dynamically, the allocation problem will be more difficult to solve. We leave this as our future work.

In this article, we aim to minimize the sampled data for a sampling task set by cooperatively allocating tasks and scheduling sampling intervals of tasks that are allocated to a sensor node. Our proposed algorithms are orthogonal to the topology of a network. In fact, the information of a network can be utilized to improve the performance of our algorithms in many respects. For example, we can divide the sampling intervals 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, such as task dividing and data fusion. We leave this as our future work as well.

8. CONCLUSION

Many applications of WSNs pursue to perform a set of interval sampling tasks for decision making. In this article, we focus on minimizing the volume of sampled data in a k-coverage and r-redundant WSN. The solving of this optimization problem depends on the optimization of two subproblems: the problem of task allocation among candidate sensor nodes and the problem of scheduling sampling intervals of sampling tasks that are allocated to a sensor node. Since the strategy of task allocation dominates the performance of the schedule of sampling intervals of sampling tasks,

we jointly optimize both subproblems. Specifically, we first propose a crucial operation for the problem, namely *COMBINE*, and give a rigorous bound of its performance. Furthermore, we present our method, which allocates tasks and computes the amount of sampled data by using *CATS*. The effectiveness and scalability of our proposals are evaluated by employing a testbed and TOSSIM, respectively. The extensive empirical study indicates that our method reduces the amount of sampled data significantly, decreases the energy consumption of a wireless sensor node, and improves the quality of communication apparently due to the decrease of data loss rate.

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 *Proceedings of the 12th IEEE 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 *Proceedings of the Conference on Communications Architectures, Protocols, and APplications (SIGCOMM'94)*. 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, 238–243.

Rashmi Dalvi. 2014. *Energy Efficient Scheduling and Allocation of Tasks in Sensor Cloud*. Master's Thesis. Missouri University of Science and Technology, Rolla, MO.

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

Ding-Zhu Du, Ker-I Ko, and Xiaodong Hu. 2012. *Design and Analysis of Approximation Algorithms*. Springer-Verlag, New York, NY.

Xiaolin Fang, Hong Gao, Jianzhong Li, and Yingshu Li. 2013. Application-aware data collection in wireless sensor networks. In *Proceedings of the 32nd IEEE International Conference on Computer Communications (IEEE INFOCOM'13)*. 1645–1653.

Chi-Fu Huang and Yu-Chee Tseng. 2003. The coverage problem in a wireless sensor network. In *Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications (WSNA'03)*. 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 *Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom'09)*. 1–5.

Philip Levis, Nelson Lee, Matt Welsh, and David Culler. 2003. TOSSIM: Accurate and scalable simulation of entire TinyOS applications. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys'03)*. 126–137.

Qiang Liu, Jianping Yin, Victor C. M. Leung, and Zhiping Cai. 2012. ISAR: Improved situation-aware routing method for wireless mesh backbones. *IEEE Communications Letters* 16, 9, 1404–1407.

Qiang Liu, Jianping Yin, Victor C. M. Leung, and Zhiping Cai. 2013. FADE: Forwarding assessment based detection of collaborative grey hole attacks in WMNs. *IEEE Transactions on Wireless Communications* 12, 10, 5124–5137.

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, 2396–2406.

Xufei Mao, Xin Miao, Yuan He, Xiang Yang Li, and Yunhao Liu. 2012. CitySee: Urban CO2 monitoring with sensors. In *Proceedings of the 31st Annual International Conference on Computer Communications (IEEE INFOCOM'12)*. 1611–1619.

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 *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys'09)*. 99–112.

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

Bo Sheng, Qun Li, Weizhen Mao, and Wen Jin. 2007. Outlier detection in sensor networks. In *Proceedings of the 8th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'07)*. 219–228.

K. Sohrabi, B. Manriquez, and G. J. Pottie. 1999. Near ground wideband channel measurement in 800-1000 MHz. In *Proceedings of the IEEE Vehicular Technology Conference*. 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 *Proceedings of the 2006 IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS'06)*. 81–90.

Makoto Suzuki, Shunsuke Saruwatari, Narito Kurata, and Hiroyuki Morikawa. 2007. A high-density earthquake monitoring system using wireless sensor networks. In *Proceedings of the 5th ACM Conference on Embedded Networked Sensor Systems (SenSys'07)*. 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* 47, 6, 34–40.

Rui Tan, Guoliang Xing, Jinzhu Chen, Wen Zhan Song, and Renjie Huang. 2013. Fusion-based volcanic earthquake detection and timing in wireless sensor networks. *ACM Transactions on Sensor Networks* 9, 2, 53–55.

Arsalan Tavakoli, Aman Kansal, and Suman Nath. 2010. On-line sensing task optimization for shared sensors. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN'10)*. 47–57.

Niki Trigoni, Yao Yong, Alan Demers, Johannes Gehrke, and Rajmohan Rajaraman. 2005. Multi-query optimization for sensor networks. In *Distributed Computing in Sensor Systems*. Lecture Notes in Computer Science, Vol. 3560. Springer, 307–321.

Liu Xiang, Jun Luo, and A. Vasilakos. 2011. Compressed data aggregation for energy efficient wireless sensor networks. In *Proceedings of the 8th Annual IEEE Communications Society Conference on Sensor, Mesh, and Ad Hoc Communications and Networks (SECON'11)*. 46–54.

S. Xiang, H. B. Lim, K.-L. Tan, and Y. Zhou. 2007. Two-tier multiple query optimization for sensor networks. In *Proceedings of the 2007 27th IEEE International Conference on Distributed Computing Systems (ICDCS'07)*. 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 *Proceedings of the 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys'04)*. 13–24.

You Xu, Abusayeed Saifullah, Yixin Chen, Chenyang Lu, and Sangeeta Bhattacharya. 2010. Near optimal multi-application allocation in shared sensor networks. In *Proceedings of the 11th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'10)*. 181–190.

Received January 2016; revised April 2016; accepted June 2016