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

Yawei Zhao, Honghui Chen, Deke Guo *Member, IEEE,* Jia Xu, Tao Chen, Jianping Yin

Abstract—Data sharing amongst multiple sampling tasks significantly reduces energy consumption and communication cost in low-power WSNs. Conventional 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 which generate continuous interval sampling tasks. Moreover, most of pioneering work limits its view to schedule *sampling interval* of tasks on a single sensor node, and neglects the process of task allocation in WSNs. Therefore, the gained efforts in the prior work can not benefit a large-scale wireless sensor network because the performance of a scheduling method is sensitive to the strategy of task allocation. Broadening the scope to an entire network, this paper is the first work to maximize data sharing amongst continuous interval sampling tasks by jointly optimizing task allocation and scheduling of *sampling interval* in WSNs. First, we formalize the joint optimization problem and prove it NP-hard. Second, we present *COMBINE* operation which is the crucial ingredient of our solution. *COMBINE* is a 2-factor approximate algorithm for maximizing data sharing amongst overlapping tasks. Furthermore, our heuristic named *CATS* is proposed. *CATS* is 2-factor approximate algorithm for jointly allocating tasks and scheduling *sampling interval* 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. Besides, the scalability of our methods is verified by using the widely-used simulation tool, i.e., TOSSIM. The experimental results indicate that our methods successfully reduce the volume of sampled data and decrease the energy consumption significantly.

Index Terms—Data sharing, continuous interval sampling task, task scheduling, task allocation, WSNs

1 Introduction

The successful applications of low-power wireless sensor networks under different scenarios, e.g., earthquake monitoring [1], railway diagnosis [2], wild-life protection [3], and structure monitoring [4], 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 [5], [6], [7] 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 [1]. The vibration data, sampled by volcano, earthquake and structure monitoring systems [2], [4], [8], 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 [3]. As illustrated in Fig. 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*. Either *time window* or *sampling interval* stands for a period of time. As illustrated in Fig. 1(a), *time window* means lifetime of a sampling task. *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. The length of *sampling interval* of a discrete point sampling task can be regarded aggressively small so that the task requires to be performed via once data sampling. The latter is a general case. That is, a continuous interval sampling task requires to 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 the paper, we focus on the latter, i.e., the continuous interval sampling tasks. [1]

Note that *time window* of more than one sampling task may have overlapping time regions. The *sampling interval* 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 Fig. 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 interval* of t_2 and t_3 can be adjusted in the overlapping time region so that data which is sampled in region of data

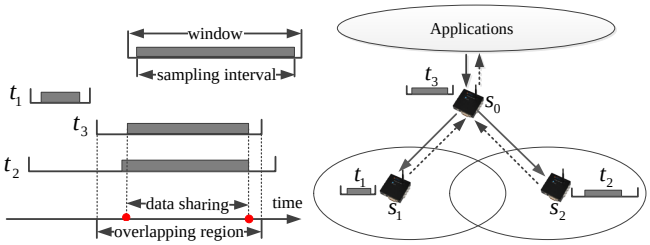
- *Y. Zhao, J. Yin are with the State Key Laboratory of High Performance Computing, National University of Defense Technology, Changsha 410073, P.R. China. E-mail: zhaoyawei09@gmail.com, jpyin@nudt.edu.cn.*
- *H. Chen, D. Guo and T. Chen, are with the Science and Technology on Information System Engineering Laboratory, National University of Defense Technology, Changsha 410073, P.R. China. E-mail: {chh0808, guodeke}@gmail.com, chentao@nudt.edu.cn.*
- *J. Xu is with the School of Computer, Electronics and Information, Guangxi University, Nanning 530004, P.R. China. E-mail: xujia@gxu.edu.cn.*

[1] For simplicity, "interval sampling tasks" or "sampling tasks" or "tasks" is hereafter referred to as "continuous interval sampling tasks" without difference.

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 those unnecessary data sampling and the transmission of redundant sampled data consume non-trivial energy of the sensor node.

Generally, given a set of tasks, data sharing exists in the processes of allocation of task and scheduling of *sampling interval*. *time window* of tasks may be overlapping, when more than one task is allocated to a same sensor node. Accordingly, the *sampling interval* of those tasks should be scheduled into the overlapping regions so as to maximize the gain of data sharing. Both the process of task allocation and the process of scheduling of *sampling interval* are tightly coupled. The strategy of task allocation has a great impact on the performance of the method of scheduling *sampling interval*. If the *time window* of sampling tasks which are allocated to a sensor node are not overlapping, the method of scheduling *sampling interval* cannot exploit data sharing to reduce redundant data sampling. If the method of scheduling *sampling interval* is not optimal, each sensor node will still produce redundant sampled data. The goal of maximizing data sharing for the entire network cannot be implemented. Therefore, exploiting the benefits of data sharing involves two challenges for WSNs. The first is the allocation strategy of sampling tasks over the entire wireless sensor network. The second is the scheduling of *sampling interval* for a set of tasks which have been allocated to a sensor node. To ease the presentation, we introduce Fig. 1 as an illustrative example for the two sub-problems.

- **Task allocation:** As demonstrated in Fig. 1(b), consider three sensor nodes s_0, s_1 and s_2 in a wireless sensor network. Here, s_0 is the sink node which is responsible to allocate 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 overlapping and *sampling interval* of t_3 and that of t_2 can be adjusted to implement maximal data sharing. The problem of task allocation is NP complete (See Corollary 1 in Section 3.3). 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 Fig. 1(a), consider three sampling tasks t_1, t_2 and t_3 which have been allocated to a same sensor node. Since the *time window* of t_2 and t_3 have 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 interval* is NP complete (See Corollary 2 in Section 3.3). 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.



(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.

Although the scheduling problem of *sampling interval* has been studied in [9], [10], [11], prior methods face at least two weaknesses. First, most of the 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 just focus on the scheduling problem on a single node. The proposed effective scheduling method do not work well in WSNs due to neglecting of task allocation procedure. The main reason is that performance of a scheduling method is dominated by the strategy of task allocation at the level of entire network. We consider and formulate the two problems as a whole, and provide a general and practical solution for a deployed system. To best of our knowledge, this paper is the first work that jointly optimizes both of the allocation and scheduling problems in a large-scale wireless sensor network.

In this paper, we focus on maximizing the benefits of data sharing amongst sampling tasks, bearing the view of an entire wireless sensor network rather than a single sensor node. Our contribution is 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* is presented theoretically.
- We propose our solution named *CATS* which jointly allocates tasks and schedules the *sampling interval* 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 a widely-used simulation tool for WSNs, i.e., TOSSIM. The evaluation results indicate that our method significantly reduces

the amount of sampled data and the energy consumption, and thereby improve the quality of communication in WSNs.

The rest of this paper 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 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 [12]. 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 same with the time window of the task model in the paper. It is noting that that tasks in [12] is communication tasks, not the sampling tasks. The difference dominates that the proposed method in [12] does not suit to our problem. Additionally, the solution for solving an integer linear programming problem in [12] costs much time, which is prohibitive for a large system. Packets in [13] owns the same properties of a sampling task we discuss in the paper. Release time and deadline of a packet represents begin time and end time, respectively. The difference is that a packet reports one unit of data, not the data which is sampled over a time interval continuously. Thus, the methods in [13] work well for the discrete point sampling tasks, not the interval sampling tasks.

The most related work to ours are [10] and [11]. Tavakoli et al. present an approach for task scheduling on a sensor node to minimize network communication overhead [10]. A task in [10] is a discrete point sampling task and requires to be performed via once data sampling during its time window. Therefore, method proposed in [10] does not suit to our problem. Fang et al. propose an effective sampling approach for interval sampling tasks on a single sensor node [11]. The 2-factor approximation algorithm in [11] is the state-of-the-art method to maximize data sharing amongst tasks on a single node. Unfortunately, there are two weak points in [11]. 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, Fang et al. assume that all tasks have a 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 information by designing a crucial operation, namely, *COMBINE*. *COMBINE* achieves better performance on maximizing data sharing than the scheduling method in [11]. Second, the solution in [11] only focuses on task scheduling on a single sensor node, and does not consider the process of task allocation in a wireless sensor network. 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 a wireless sensor network

Recently, a wireless sensor network is treated as a database providing a good logical abstraction for sensor data management. Multi-query optimization in such a database system studies how to efficiently process queries [14], [15]. Xiang et al. adopts a two-tier multiple query optimization scheme to minimize the average transmission time in WSNs [14]. The first-tier optimization is a cost-based approach which schedules queries as a whole and eliminates duplicate data requests form 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 acquire and transmit sampled data by using the broadcast nature of the radio channel. Our solution aims to provide a general solution for maximizing 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 optimal communication cost [15]; while our method mainly concerns minimal energy consumption by reducing redundant sampled data.

2.3 Compression techniques based data collection

Compression techniques based data collection significantly reduce redundancy of sampled data for a sensor node [16], [17]. Arici et al. propose an in-network compression scheme (PINCO) for a densely deployed wireless sensor network. PINCO compresses raw data by reducing redundancy existing in sensor readings in spatial, temporal and spatial-temporal domains [16]. 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 do 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 [16] 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 [17]. However, method in [17] requires to get known of the sensor correlation structure. It was impossible for a randomly deployed system such as a battlefield monitoring system.

3 Overview and Preliminaries

In this section, the basic task and network models are first formalized. We characterize the task allocation and the scheduling

of *sampling interval* 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 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. That is, the *sampling interval* can select its beginning time in the time window, as long as its end time does not exceed e.

As illustrated in Fig. 2(a), there are two overlapping tasks t_1=$<b_1,e_1,l_1>$ and t_2=$<b_2,e_2,l_2>$. 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.

Definition 2 (k-coverage and r-redundant network): A wireless sensor network 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, while only are r out of them identified to execute the task.

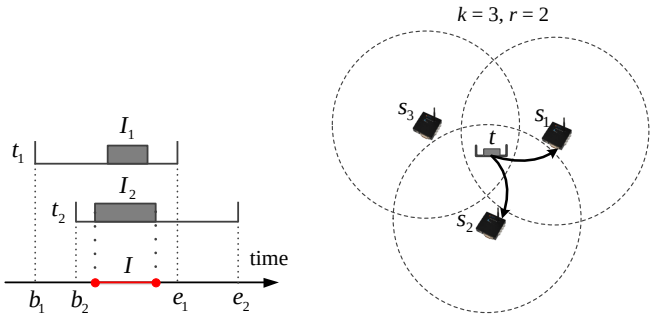
For a k-coverage and r-redundant network, k is determined during the deployment of a wireless sensor network. Additionally, the setting of r can be adjusted according to the requirement of an application. In this paper, we consider the case that k and r are determined after the deployment of a wireless sensor network. As illustrated in Fig. 2(b), consider a wireless sensor network which 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 so as 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 interval*. To ease the description, Table 1 outlines frequently used symbols throughout the paper.

Definition 3: 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

TABLE 1
Symbols list

Symbol	Notation	Symbol	Notation
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 novel 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		



(a) Task model (b) Network model

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

are overlapping. Assume u=$\min\{u_1,u_2\}$ and v=$\max\{v_1,v_2\}$, 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 4 (The allocation of tasks): Given a task t and a sensor node set S, the task set which has been allocated to the sensor node s, s∈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:

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

$s.t.:\ I_i \subseteq [b_i,e_i], i=1,...,n;$

Definition 5 (The scheduling of *sampling interval*): Given a task set T with $|T|$=n, the optimization problem of scheduling *sampling interval* of tasks in T is defined as follows:

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

$s.t.:\ I_i \subseteq [b_i,e_i], i=1,...,n;$

Definition 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:

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

Definition 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∈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:

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

$s.t.:$

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

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 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 pretty hard to be solved.

3.2 Measurement of data sharing between overlapping tasks

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

Definition 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 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 $e_i \leq e_j$, then:

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

Definition 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:

$$\begin{cases} b^*=e_i-l_i \\ e^*=b_j+l_j \end{cases} \quad (7)$$

- If t_i satisfies t_j, then:

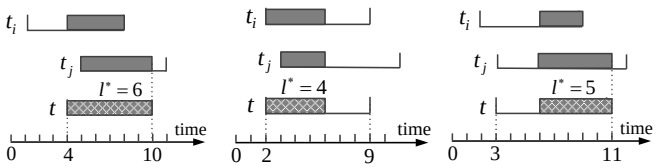
$$\begin{cases} b^*=\max\{b_i,b-(l_i-l_j)\} & b=\max\{b_i,b_j\} \\ e^*=\min\{e_i,e+(l_i-l_j)\} & e=\min\{e_i,e_j\} \end{cases} \quad (8)$$

- If t_j satisfies t_i, then:

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

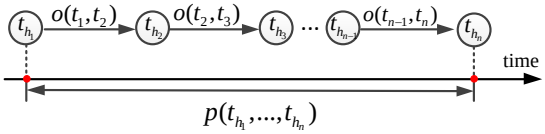
Definition 11 (Combination): A novel task t_{ij} is the combination of overlapping tasks t_i and t_j. For simplicity, t_{ij} is denoted by t^*=<b^*,e^*,l^*> which is called the child task, while t_i and t_j are called father tasks. Here, both b^* and e^* are computed according to Definition 10. $l^*=l_i+l_j-d(t_i,t_j)$.

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

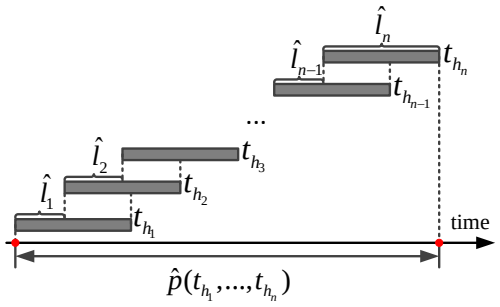


(a) t_i and t_j do not satisfy with each other. (b) t_i satisfies t_j. (c) t_j satisfies t_i.

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



(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 minimal volume of sampled data.

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 a NP complete problem [18]. We then prove that the optimization problem is NP-hard by the transformation from MAX-DHP.

Definition 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 1: Given a sampling task set T with $|T|>2$, then the computation of the minimal volume of sampled data 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 optimal solution, intervals in Φ must not be overlapping. Then, we check each an interval whether it can be removed and Φ still satisfies the left tasks. If any an 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. So the computation of the volume of sampled data is a 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 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 Fig. 4(a), iteratively repeat the step until all the tasks have been checked and the directional graph G is constructed.
- **Step 2:** Given a maximal Hamilton path t_{h_1}, t_{h_2}, ..., t_{h_n} in the graph G, the length of the directed Hamilton path $p(t_{h_1},...,t_{h_n})$ in Fig. 4(a) is calculated as follows:

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

  As illustrated in Fig. 4(b), $\widehat{p}(t_{h_1},...,t_{h_n})$ 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 1 is proved. □

Corollary 1: Given a sampling task set and a sensor node set, the problem of task allocation is NP complete.

Corollary 2: Given a sampling task set, the problem of scheduling *sampling interval* of tasks is NP complete.

Theorem 1: 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 of which 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 1, is NP complete. So the special case of the joint optimization problem is NP-hard. In general, this joint optimization problem is at least 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 algorithm, i.e., *COMBINE* which maximizes data sharing amongst overlapping tasks. We then prove the bound of the algorithm by theoretical analysis rigorously. *COMBINE* is the crucial ingredient of our solution.

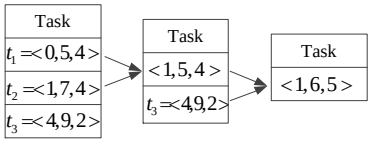
Algorithm 1 COMBINE(T)

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 which 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.

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 *COMBINE* for overlapping tasks.

Let T denote a non-empty 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 which 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. Second, original tasks t_i and t_j are replaced by the novel task t_{ij} in the task set T. Third, repeat above steps until no overlapping tasks exist. Algorithm 1 presents 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 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 showed 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 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

Algorithm 2 COMBINE_2(T)

Require: A task set T and $T \neq \emptyset$.

```
sort tasks in T in the descending order of end time.
while |T|>1 do
    find a task pair <t_i,t_j> which has the maximal value of data sharing.
    if d(t_i,t_j)==0 then
        return T.
    insert a novel task t_ij into T in the descending order of end time. Here, t_ij ← t_i ⊗ t_j .
    remove t_i and t_j from T.
return T.
```

task t_{12}=<1,5,4>. Second, remove t_1 and t_2 from T and add t_{12} into T. We get a novel task, i.e., <1,6,5> after combining t_{12} and t_3. So 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 algorithm by Theorem 2. Before presenting Theorem 2, we first present Lemma 2 and Property 1 which will be used in the proof of Theorem 2.

Lemma 2: For a non-empty 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 an 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 $d(t_i,t_k) \geq d(t_j,t_k)$.

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 $d(t_{ij},t_k) \leq d(t_j,t_k)$. □

Property 1: The value of data sharing between two overlapping 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. $\forall t_k \in T$, $d(t_i,t_j) \geq d(t_{ij},t_k)$ always establishes according to Lemma 2. Then we can prove the Property 1 by using the method of mathematical induction. Since 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 can not 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 2. Thus Property 1 is proved. □

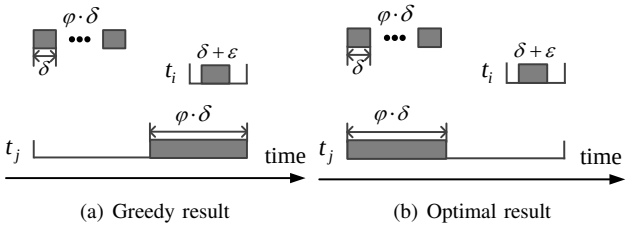


Fig. 6. A typical example of the approximate result by using Algorithm 1 vs. the optimal result.

Definition 13 (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 14 (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:

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

Theorem 2: Algorithm 1 is a 2-factor approximation algorithm to compute the minimal volume of sampled data.

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. $I^{T'}$ is the *sampling interval* of a task set which 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 Fig. 6(a). However, there exists an optimal algorithm which derives an interval length $|I_i \uplus I^{T'}|$ as illustrated in Fig. 6(b). Therefore, we have:

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

Here, $\delta \leq d(t_i,t_j)$. □

A typical example is showed in Fig. 6. Algorithm 1 returns the interval length $2\varphi\cdot\delta$ where $\delta < d(t_i,t_j)$, while 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.

Algorithm 3 RANDOM(T,S,k,r)

Require: A task set T and $T\neq\emptyset$. A sensor node set S and $S\neq\emptyset$. The task subset of a sensor node t_i is denoted by T_i with $T_i=\emptyset$. $1\leq i\leq m$, $r\leq k$.

1: **while** $T\neq\emptyset$ **do**
2: randomly allocate a task t, $t\in T$ to one of k candidate sensor nodes s_i which do not have been allocated t before.
3: $T_i \leftarrow T_i \bigcup \{t\}$.
4: $t.count=t.count+1$.
5: **if** $t.count==r$ **then**
6: remove t form T.
7: **COMBINE**(T_i) for each task subset T_i.

Algorithm 4 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.

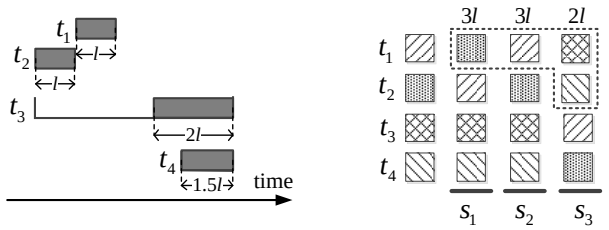
5 Approximation Algorithms for the Joint Optimization Problem in WSNs

In this section, three algorithms are presented for the joint optimization problem. The first is a random algorithm 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 the value of data sharing. The third is a 2-factor approximation algorithm which computes the volume of sampled data by iteratively combining overlapping tasks.

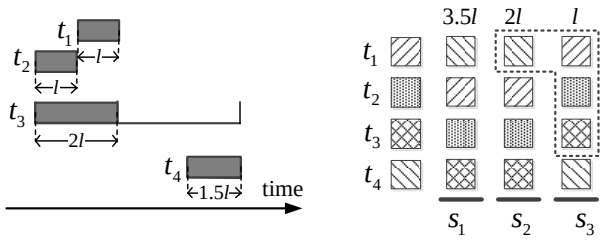
For a task set, the insight of the random method is to randomly identify r out of k candidate sensor nodes. This method is simple and easy to be performed on a sensor node. However, it neglects data sharing amongst tasks and brings a wealth of unnecessary sampled data. This method consumes much energy of a sensor node and damages the quality of communication in WSNs as well. The random algorithm is outlined in Algorithm 3.

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 pruning operation which removes unreasonable allocation choices repeatedly. The entire process contains three major steps. First, we allocate tasks to all candidate sensor nodes.



(a) The approximate result of task allocation by using Algorithm 4.



(b) The optimal result of task allocation.

Fig. 7. A typical example of the approximate result by using Algorithm 4 vs. 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 pruning process.

That is, 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 4 describes more details.

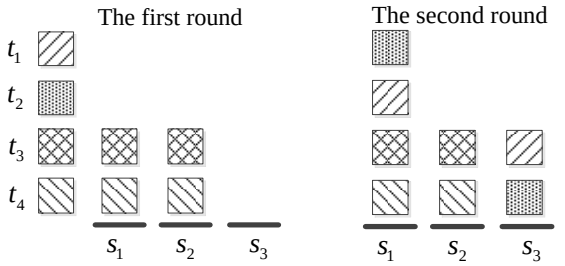
In Algorithm 4, lines $1-3$ allocate a task to its k candidate sensor nodes. Lines $4-10$ check whether a task has been removed from $k-r$ candidate sensor nodes. Line 11 computes the final sampled data for each sensor node using Algorithm 1. In the worst case, if Algorithm 1 is used to compute the volume of sampled data, the time complexity of Algorithm 4 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 4 is $O(n)$ because each a task maintains a list of value of data sharing with other tasks.

Algorithm 4 is a greedy algorithm. A typical example is showed in Fig. 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 Fig. 7, the left panel demonstrates the computation of data sharing by Algorithm 4 and the optimal method. Four rectangles in the right panel stand for the tasks in the left panel. The pruning method first allocates all the sampling tasks to candidate nodes, and then removes the tasks which have smallest volume of data sharing with other tasks (indicated in dotted polygon). Finally, Algorithm 4 returns the total length of sampling intervals: $8l$ as showed in Fig. 7(a), while the optimal result is $6.5l$ as showed in Fig. 7(b). The reason is that our pruning procedure is not optimal in each round. This motivates us to propose Algorithm 5 which allocates tasks according to the *COMBINE* operation and can be optimal in each round.

Algorithm 5 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 $<t_1,t_2>$ from T which has the maximal value of data sharing.
3: combine t_1 and t_2 and notate the resultant task by t_{12}.
4: remove t_1 and t_2 from T.
5: add t_{12} into T.
6: **while** T is not an empty set **do**
7: randomly identify a task $t,t\in T$.
8: **if** t is not an original task **then**
9: identify the original tasks which generate t after process of *COMBINE*, and allocate them to r sensor nodes.
10: **else**
11: allocate t to r sensor nodes.



(a) The first round of task combination by using Algorithm 5.

(b) The second round of task combination by using Algorithm 5.

Fig. 8. The illustrative example for the allocation of tasks by combining overlapping tasks 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 which allocates tasks and computes the volume of sampled data by using the *COMBINE* operation. As illustrated in Fig. 2, a task t_0 is detected by three sensor nodes, but only two of them are identified to perform it. If the task t_0 is overlapping with three tasks t_1, t_2 and t_3 such that $d(t_0,t_1)>d(t_0,t_2)>d(t_0,t_3)$. If 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 which 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 5 presents the method in detail. In Algorithm 5, Lines 1−5 present the combination of the overlapping tasks. Lines 6−11 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 5 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 novel 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.

Theorem 3: Algorithm 5 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 5 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 5 is thus a 2-factor approximation algorithm.

When $1\leq r<k$, Algorithm 5 selects a task pair with the maximal value of data sharing from the task set repeatedly. Algorithm 5 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 2 and returns a 2-factor approximate result of the volume of sampled data. Thus, Algorithm 5 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 cycle) 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 a vertex stands for a task. If two tasks are overlapping, then a weighted edge exists. The weighted value represents the

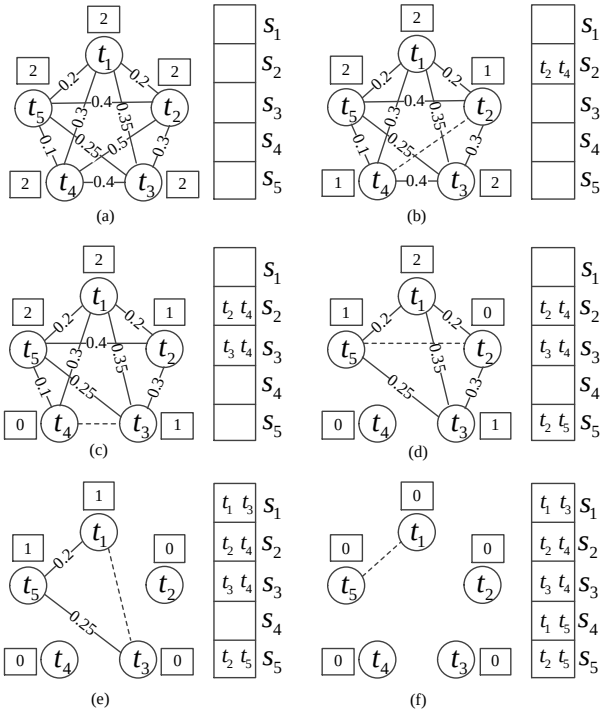


Fig. 9. The illustrative example of transformation from Algorithm 1 to Algorithm 5 by combining the overlapping tasks which have the maximal value of data sharing in each step. Here, k=3 and r=2.

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 novel tasks generated from *COMBINE* operation 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 novel tasks, novel vertices will added to the graph, but this will not impact the performance of Algorithm 5.

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, the widely-used simulation tool, i.e., TOSSIM, is used to verify the scalability of our method.

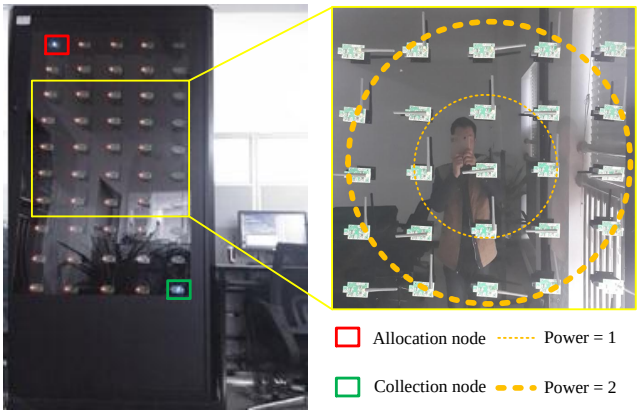


Fig. 10. The output power can be adjusted to realize the k-coverage network in the testbed.

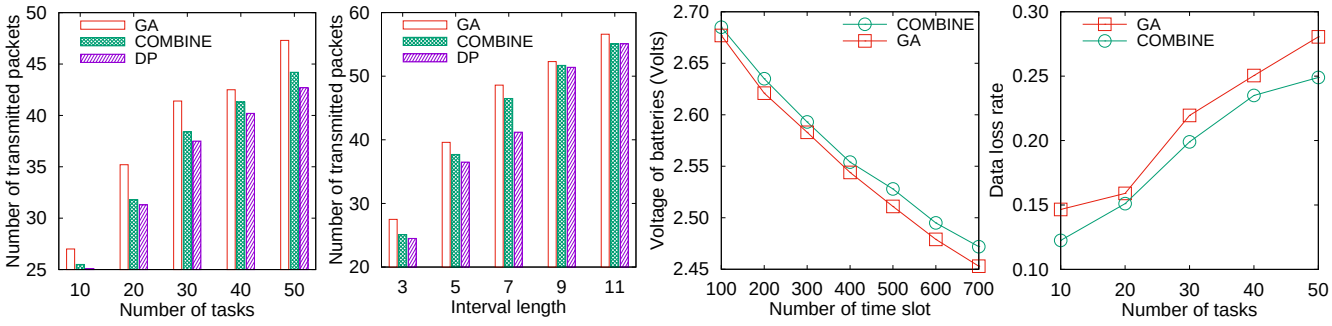
6.1 Experimental environment and settings

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 experiments, we construct different k-coverage networks by adjusting 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 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 [19], [20].

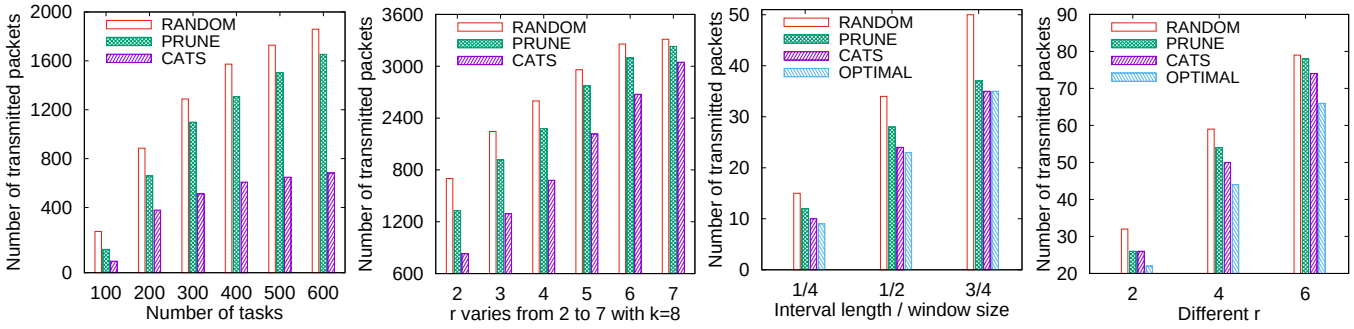
Since the wireless sensor network is densely deployed, any two nodes can set up a wireless link by at most three hops. The proposed algorithms are implemented in a centralized manner which is widely used in actual systems [5], [6], [21], [22]. We use the left-top node (as highlighted by a red rectangular) 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 varies from 2 to 8, which is reasonable and always used in the practical wireless sensor network systems [23], [24]. 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 the 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.



(a) l=5 (b) n=30 (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).



(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).

6.2 Performance of data sharing on a sensor node

At first, we evaluate the performance of our Algorithm 1, denoted by *COMBINE*, against the state-of-the-art method [11], 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 dynamic programming technique on 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 when using *COMBINE*, *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 both the growth of the number of tasks and the interval length. However, *COMBINE* performs better and returns smaller number of transmitted packets than *GA*. This happens because our algorithm combines overlapping tasks which have the maximal value of data sharing for each step. Thus, each step is the current optimal choice. Consider that *GA* schedules the *sampling interval* of tasks based on the end time of a task, it cannot make sure each scheduling choice is optimal. That is the reason why *COMBINE* outperforms *GA*. Meanwhile, Fig. 11(a) and Fig. 11(b) indicate *COMBINE* achieves 2-factor approximation result vs. the optimal result. This verifies the correctness of the Theorem 2.

In Fig. 11(c), we run both *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 that *GA* does. This is because that *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 wireless sensor network. Precisely, *COMBINE* decreases energy consumption by 4.85% per slot on average. In Fig. 11(d), we test the data loss rate 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 the 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 over decades of days, such improvements is appreciable and worthy

to being exploited. Moreover, *GA* is specifically designed for the scheduling of *sampling interval*. 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 more data loss rate than our solution by over 35% and 30% due 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 intra-network communication and decreases transmission delay and congestion by using *COMBINE*.

6.3 Performance of data sharing across WSNs

In this part, we verify the performance of Algorithm 3, 4 andp 5 which are represented by *RANDOM*, *PRUNE* and *CATS*, 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, *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 *CATS* reduce more transmitted packets than *RANDOM* does. Another observation is that the advantage of *PRUNE* and *CATS* 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 solution which shares the same performance with *PRUNE* and *CATS*.

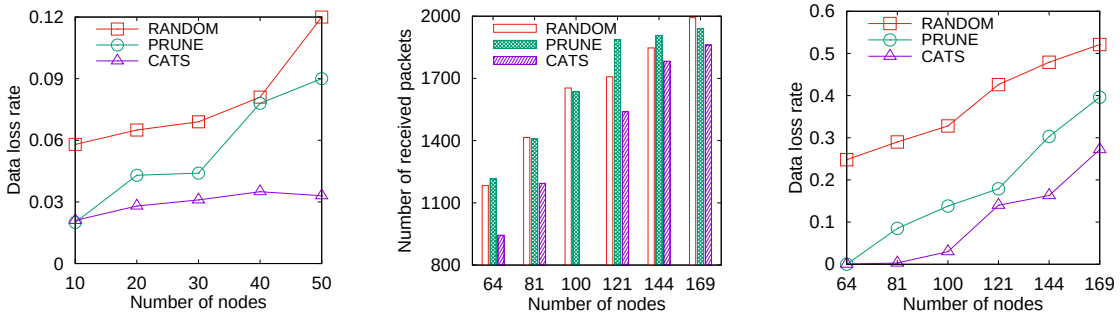
It is difficult to derive an effective optimal solution for the joint optimization problem. However, when the scale of tasks is small, we can find the optimal result by using a brute-force method. To evaluate the performance of our method rigorously, we compare the number of transmitted packets on each method by varying the number of tasks and the value of r. In 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. 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 closely to the optimal allocation solution, i.e., *OPTIMAL*. This confirms the conclusion of Theorem 3 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 *CATS* achieves greatly smaller number of transmitted packets than twice of that produced by the optimal solution. It verifies the conclusion of Theorem 3 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 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* does. This verifies the benefits of our solution on the improvement of 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 [25]. We construct a grid network and the settings of the simulate network are listed in Fig. 14 which are widely accepted [26]. As illustrated in Fig. 13(b), we compare the number of received data by varying the scale of the network. We apparently observe that *CATS* brings the smallest amount of received data when the scale of the network grows. The reason is that *CATS* reduces more unnecessary sampled data than *PRUNE* and *RANDOM*. Noting that *PRUNE* brings more received data than *RANDOM* when the number of sensor nodes in the network is 121 and 144. The reason is that *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 Fig. 13(c), the data loss rate in both *CATS* and *PRUNE* is much smaller than that in *RANDOM*. It 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 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 runs CATS and 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 few. 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 few 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. We aim



(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.

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	The internode distance of the grid
NUMBER_OF_NODES	100	Number of nodes in the grid WSNs

Fig. 14. The parameter settings in the simulations by using TOSSIM. The left column is the parameters which can be used to configure a grid wireless sensor network, the middle column is the value, and the right column is the notions of these parameters.

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 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 carrying. 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 4 and Algorithm 5 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 relay on the topology of a network. In fact, the information of a network can be utilized 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 wireless sensor networks pursue to perform a set of interval sampling tasks for decision-making. In this paper, we focus on minimizing the volume of sampled data in a k-coverage and r-redundant wireless sensor network. The solving of this optimization problem depends on the optimization of two sub-problems: the problem of task allocation amongst candidate sensor nodes and the problem of scheduling *sampling interval* of sampling tasks which are allocated to a sensor node. Since the strategy of task allocation dominates the performance of the scheduling of *sampling interval* of sampling tasks. We jointly optimize both sub-problems. To be specific, 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 is evaluated by employing a testbed and TOSSIM, respectively. The extensive empirical study indicates that our method reduces the amount of sampled data significantly, saves rare energy considerably, and improves the quality of communication apparently due to the decrease of data loss rate.

Acknowledgments

The work is partially supported by the National Natural Science Foundation of China (NSFC) under Grants NO. 61402494, NO. 61402513, NO. 61202487, NO. 61170287, NO. 61232016. Besides, the project is sponsored by the Scientific Research Foundation of GuangXi University under Grant No. XGZ141182.

References

[1] M. Cerullo, G. Fazio, M. Fabbri, F. Muzi, and G. Sacerdoti, "Acoustic signal processing to diagnose transiting electric trains," *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 2, pp. 238–243, 2005.

[2] M. Suzuki, S. Saruwatari, N. Kurata, and H. Morikawa, "A high-density earthquake monitoring system using wireless sensor networks," in *Proc. ACM SenSys*, Sydney, Australia, 2007, pp. 373–374.

[3] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin, "Habitat monitoring with sensor networks," *Communications of the ACM-Wireless sensor networks*, vol. 47, no. 6, pp. 34–40, 2004.

[4] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, "A wireless sensor network for structural monitoring," in *Proc. ACM SenSys*, Baltimore, Maryland, USA, 2004, pp. 13–24.

[5] L. Mo, Y. He, Y. Liu, J. Zhao, S.-J. Tang, X.-Y. Li, and G. Dai, "Canopy closure estimates with greenorbs: Sustainable sensing in the forest," in *Proc. ACM SenSys*, Berkeley, California, USA, 2009, pp. 99–112.

[6] M. Jiang, Z. Guo, F. Hong, Y. Ma, and H. Luo, "Oceansense: A practical wireless sensor network on the surface of the sea," in *Proc. IEEE PerCom*, Galveston, Texas, USA, 2009, pp. 1–5.

[7] X. Mao, X. Miao, Y. He, X. Y. Li, and Y. Liu, "Citysee: Urban co2 monitoring with sensors," in *Proc. IEEE INFOCOM*, 2012, pp. 1611–1619.

[8] R. Tan, G. Xing, J. Chen, W. Z. Song, and R. Huang, "Fusion-based volcanic earthquake detection and timing in wireless sensor networks," *Acm Transactions on Sensor Networks*, vol. 9, no. 2, pp. 53–55, 2013.

[9] R. Dalvi, "Energy efficient scheduling and allocation of tasks in sensor cloud," *Dissertations & Theses - Gradworks*, 2014.

[10] A. Tavakoli, A. Kansal, and S. Nath, "On-line sensing task optimization for shared sensors," in *Proc. ACM/IEEE IPSN*, Stockholm, Sweden, 2010, pp. 47–57.

[11] X. Fang, H. Gao, J. Li, and Y. Li, "Application-aware data collection in wireless sensor networks," in *Proc. IEEE INFOCOM*, Turin, Italy, 2013, pp. 1645–1653.

[12] Y. Xu, A. Saifullah, Y. Chen, C. Lu, and S. Bhattacharya, "Near optimal multi-application allocation in shared sensor networks," in *Proc. ACM MobiHoc*, 2010, pp. 181–190.

[13] W. Z. Song, F. Yuan, and R. LaHusen, "Time-optimum packet scheduling for many-to-one routing in wireless sensor networks," in *Proc. IEEE MASS*, Vancouver, Canada, 2006, pp. 81–90.

[14] S. Xiang, H. B. Lim, K.-L. Tan, and Y. Zhou, "Two-tier multiple query optimization for sensor networks," in *Proc. IEEE ICDCS*, 2007, pp. 3–9.

[15] N. Trigoni, Y. Yong, A. Demers, J. Gehrke, and R. Rajaraman, "Multi-query optimization for sensor networks," *Springer Lecture Notes in Computer Science*, pp. 307–321, 2005.

[16] T. Arici, B. Gedik, Y. Altunbasak, and L. Liu, "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*, 2003, pp. 539–544.

[17] S. S. Pradhan, J. Kusuma, and K. Ramchandran, "Distributed compression in a dense microsensor network," *IEEE Signal Processing Magazine*, vol. 19, no. 2, pp. 51–60, 2002.

[18] X. H. Ding-Zhu Du, Ker-I Ko, *Design and Analysis of Approximation Algorithms*. SpringerVerlag New York, 2012.

[19] I. Demirkol, C. Ersoy, and F. Alagoz, "Mac protocols for wireless sensor networks: a survey," *IEEE Communications Magazine*, vol. 44, no. 4, pp. 115–121, 2006.

[20] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "Macaw: a media access protocol for wireless lan's," in *ACM SIGCOMM*, London, UK, 1994, pp. 212–225.

[21] B. Sheng, Q. Li, W. Mao, and W. Jin, "Outlier detection in sensor networks," in *Proc. ACM MobiHoc*, Montreal, Quebec, Canada, 2007, pp. 219–228.

[22] L. Xiang, J. Luo, and A. Vasilakos, "Compressed data aggregation for energy efficient wireless sensor networks," in *Proc. IEEE SECON*, Salt Lake City, Utah, USA, 2011, pp. 46–54.

[23] X. Mao, Y. Liu, S. Tang, H. Liu, J. Han, and X.-Y. Li, "Finding best and worst k-coverage paths in multihop wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 12, pp. 2396–2406, 2013.

[24] C.-F. Huang and Y.-C. Tseng, "The coverage problem in a wireless sensor network," in *Proc. ACM WSNA*, San Diego, California, USA, 2003, pp. 115–121.

[25] P. Levis, N. Lee, M. Welsh, and D. Culler, "Tossim: accurate and scalable simulation of entire tinyos applications," in *Proc. ACM Sensys*, Los Angeles, California, USA, 2003, pp. 126–137.

[26] K. Sohrabi, B. Manriquez, and G. Pottie, "Near ground wideband channel measurement in 800-1000 mhz," in *Proc. IEEE Vehicular Technology Conference*, Houston, Texas, USA, 1999, pp. 571–574.