Energy Efficient Routing with Adaptive Data Fusion in Sensor Networks

Hong Luo*
College of Computer Science and Technology
Beijing Univ. of Posts and Telecommunications
100876, Beijing, China
luohongmm@163.com

Jun Luo, Yonghe Liu†, and Sajal K. Das
Dept. of Computer Science and Engineering
The University of Texas at Arlington
Arlington, TX 76019
{juluo, yonghe, das}@cse.uta.edu

ABSTRACT

While in-network data fusion can reduce data redundancy and hence curtail network load, the fusion process itself may introduce significant energy consumption for emerging wireless sensor networks with vectorial data. Therefore, fusion-driven routing protocols for sensor networks cannot optimize over communication cost only – fusion cost must also be accounted for. Towards this end, we design a novel routing algorithm, called *Adaptive Fusion Steiner Tree (AFST)*, for energy efficient data gathering in sensor networks that jointly optimizes over the costs for data transmission and data fusion. Furthermore, AFST evaluates the benefit and cost of data fusion along information routes and adaptively adjusts whether fusion shall be performed. Analytically and experimentally, we show that AFST achieves better performance than existing algorithms including SLT, MFST, and SPT.

Categories and Subject Descriptors

C.2.2 [**Computer-communication networks**]: Network Protocols—*Routing Protocols*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Routing and layout*; G.2.2 [**Discrete Mathematics**]: Graph Theory—*Network problems*

General Terms

Algorithms, Theory

Keywords

Sensor networks, data gathering, data fusion, routing

1. INTRODUCTION

Energy efficient routing algorithms for data gathering is a major concern in wireless sensor networks [5, 8, 17, 13, 6, 7, 3, 2, 15, 19]. By exploring data correlation and employing in-network processing, redundancy among sensed data can be curtailed and hence the network load can be reduced [8]. The objective of sensor routing algorithms is then to jointly explore the data structure and network topology to provide the optimal strategy for data gathering with as minimum energy as possible.

Regardless of the techniques employed, existing strategies miss one key dimension in the optimization space for routing correlated data, namely the *data aggregation cost*. Indeed, the cost for data aggregation may not be negligible for certain applications. For example, sensor networks monitoring field temperature may use simple average, max, or min functions which essentially are of insignificant cost. However, other networks may require complex operations for data fusion[1]. Energy consumption of beamforming algorithm for acoustic signal fusion has been shown to be on the same order of that for data transmission [18]. In our own experimental study described in [10], we show that aggregation processes such as image fusion cost tens of nJ per bit, which is on the same order as the communication cost reported in the literature [5, 18].

Different from transmission cost that depends on the output of the fusion function, the fusion cost is mainly determined by the inputs of the fusion function. Therefore, in addition to transmission cost, the fusion cost can significantly affect routing decisions when involving data aggregation. In our prior work [9], we presented a randomized algorithm termed *Minimum Fusion Steiner Tree (MFST)* that jointly optimizes over both the fusion and transmission costs to minimize overall energy consumption. MFST is proved to achieve a routing tree that exhibits $\frac{5}{4}log(n+1)$ approximation ratio to the optimal solution, where n denotes the number of source nodes.

While MFST has been shown to outperform other routing algorithms including *Shortest Path Tree* (SPT), *Minimum Spanning Tree* (MST), and *Shallow Light Tree* (SLT) in various system settings, it assumes that aggregation is performed at the intersection nodes whenever data streams encounter. However, as we shall show below, such a strategy may introduce unnecessary energy consumption. Specifically, performing fusion at certain nodes may be less efficient than simply relaying the data directly. This observation motivates us to design an adaptive fusion strategy that not only optimizes information routes, but also embeds the decisions as to when and where fusion shall be performed in order to minimize the total network energy consumption.

*The work is done while the author was at the Department of Computer Science and Engineering at The University of Texas at Arlington.

†Corresponding author.

[1]In this paper we will consider "aggregation" and "fusion" interchangeable, denoting the data reduction process on intermediate sensor nodes.

DIALM-POMC'05, September 2, 2005, Cologne, Germany.

1.1 Motivation

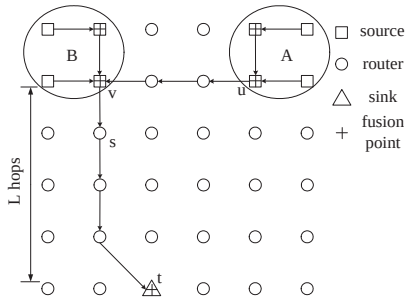


Figure 1: Fusion benefit/disadvantage in sensor networks.

Fig. 1 depicts a sensor network where sensor nodes are deployed on grid and sensed information of the source nodes is to be routed to sink t. Each hop has identical unit transmission cost c_0. Assume the fusion cost is linear to the total amount of incoming data, and the unit fusion cost is q_0. According to MFST, nodes u and v initially aggregate data of areas A and B, respectively. As the sink is far away, u and v shall further aggregate their data and then send one fused data to the sink. Assume that node v is selected as the aggregation point and let $w(u)$ and $\widetilde{w}(v)$ respectively denote the amount of data at u and v before the aggregation between u and v. Let us also assume that the amount of resultant aggregated data at v can be expressed as $w(v) = (w(u) + \widetilde{w}(v))(1 - \sigma_{uv})$, where σ_{uv} represents the data reduction ratio owing to aggregation. In this scenario, the total energy for v to aggregate and deliver the data to the next hop, node s, is $c_0(w(u)+\widetilde{w}(v))(1-\sigma_{uv})+q_0(w(u)+\widetilde{w}(v))$. On the contrary, if v forwards both u and v's data directly to the next hop without performing aggregation, the total energy consumption at v is $c_0(w(u) + \widetilde{w}(v))$. From v's point of view, to save energy, it shall not perform data fusion but simply relay u's data as long as $\sigma_{uv} < \frac{q_0}{c_0}$.

However, the above conclusion is only valid if node v is selfish and noncooperative, as the decision of whether to fuse or not at v will also impact the energy consumption of the whole path from v to sink t. If the data is not fused at v, succeeding relaying nodes will incur additional communication cost due to higher payload. Therefore, in order to optimize the whole network energy consumption, node v's decision has to be based on a network-centric point of view, i.e., the effect on the total network energy consumption.

In this example, if v performs data fusion, the total energy consumption of the route from v to t, assuming there are L hops in between, is $Lc_0(w(u) + \widetilde{w}(v))(1 - \sigma_{uv}) + q_0(w(u) + \widetilde{w}(v))$. On the contrary, if v does not perform data fusion, the total energy consumption of the same route is simply the total relaying cost, $Lc_0(w(u) + \widetilde{w}(v))$. To minimize the *total energy consumption of the network*, v should perform data fusion as long as $\sigma_{uv} < \frac{q_0}{Lc_0}$.

This simple example reveals that to minimize total network energy consumption, the decision at an individual node has to be based on data reduction ratio due to aggregation, its related cost, and its effect on the communication costs at the succeeding nodes. Although the criteria can be easily obtained for this simple example, a sensor network confronting various aggregation/communication costs, and data/topology structures, undoubtedly will dramatically augment the difficulty of the fusion decisions. Hence, the motivation of this paper is to design an energy efficient routing algorithm that not only jointly explores the data correlation structure and network topology, but also adaptively determines if aggregation shall occur on the routing nodes.

1.2 Related Work

If the complete knowledge of all data correlations is available in advance at each source, theoretically the best routing strategy is to use a distributed source coding typified by Slepian-Wolf coding [14]. An optimal rate allocation algorithm for nodes in the network is proposed in [2] and SPT is employed as the routing scheme. However, implementation of distributed source coding in a practical setting is still an open problem and likely to incur significant additional cost because of the requirement on the knowledge of network wise correlation.

Routing with data aggregation can be generally classified into two categories: routing-driven and aggregation-driven. Routing-driven algorithms [5, 8, 17, 6, 7] emphasize source compression at each individual node and aggregation occurs opportunistically when routes intersect. On the contrary, routing paths in aggregation-driven algorithms [3, 2, 15] are heavily dependent on data correlation in order to fully benefit from information reduction resulting from data aggregation. In [2], the authors proved that the minimum-energy data gathering problem is NP-complete by applying reduction set-cover problem and claimed that the optimal result is between SPT and the travelling salesman path. In [3], a hierarchical matching algorithm is proposed resulting in an aggregation tree with a logarithmic approximation ratio to the optimal for all concave aggregation functions. In this model, each node can theoretically obtain the joint entropy of its subtree to receive the maximal aggregation ratio. However, aggregation only depends on the number of nodes in the subtree rooted at the aggregation node regardless of the correlation among the data.

Indeed, the idea of embedding fusion decisions in routing has been implicitly explored in the literature. For example, LEACH [5] is a cluster-based protocol, in which sensors directly send data to cluster heads where data fusion is performed. Aggregated data is then delivered to the sink through multi-hop paths. The authors of [15] proposed an optimal algorithm MEGA for foreign-coding and an approximating algorithm LEGA for self-coding. In MEGA, each node sends raw data to its encoding point using directed MST, and encoded data is then transmitted to the sink through SPT. LEGA uses SLT [4, 16] as the data gathering topology, and achieves $2(1 + \sqrt{2})$ approximation ratio for self-coding. LEGA and MEGA implicitly assume that fusion stops after first aggregation as encoded data cannot be recoded again. However, the decision regarding fusions in these schemes are rather static and cannot adapt to network/data structure changes. As demonstrated earlier, this decision shall be based on various conditions of the networks in order to minimize energy consumption.

1.3 Our Contribution

In this paper, we propose *Adaptive Fusion Steiner Tree* (AFST), a routing scheme that not only optimizes over both transmission and fusion costs, but also adaptively adjusts its fusion decisions for sensor nodes. By evaluating whether fusion is beneficial to the network based on fusion/transmission costs and network/data structures, AFST dynamically assigns fusion decisions to routing nodes during the route construction process. Analytically we prove that AFST outperforms MFST. Through an extensive set of simulations, we demonstrate that AFST provides significant energy saving over MFST (up to 70%) and other routing algorithms under a wide range of system setups. By adapting both the routing tree and fusion decisions to various network conditions, including fusion cost, transmission cost, and data structure, AFST provides a routing algorithm suitable for a broad range of applications.

In particular, we prove that the routing tree resulting from AFST is partitioned into two parts: a lower part where aggregation is always performed, and an upper part where no aggregation occurs.

The result can be readily applied in designing clustering algorithms in sensor networks: based on where fusion stops, the network can be partitioned into clusters where data aggregation is confined to be within the clusters only.

The remainder of this paper is organized as follows. In Section 2, we describe the system model and formulate the routing problem. Section 3 gives an overview of the randomized approximation algorithm MFST. Section 4 presents in detail the design and analysis of the proposed algorithm AFST. In Section 5 we experimentally study the performance of AFST. Section 6 concludes the paper.

2. SYSTEM MODEL AND PROBLEM FORMULATION

2.1 Network Model

We model a sensor network as an undirected graph $G = (V, E)$ where V denotes the node set and E the edge set representing the communication links between node-pairs. We assume a set $S \subset V$ of n nodes, are data sources of interests and the sensed data needs to be gathered at a special sink node $t \in V$.

We refer to the process of gathering information during a certain time interval from each sensor node in S to sink t as a *round*. Therefore, at each round, data from all nodes in S has to be sent to t, where it is further processed.

For a node $v \in S$, we define node weight $w(v) : S \rightarrow \Re^+$, denoting the amount of information outgoing from v in every round. Evidently, various data fusion algorithms will result in different weights and therefore a node's weight is *dynamic* in the process of data gathering.

An edge $e \in E$ is denoted by $e = (u, v)$, where u is the start node and v is the end node. The weight of edge e is equivalent to the weight of its start node, i.e., $w(e) = w(u)$. Two metrics, $t(e)$ and $f(e)$, are associated with each edge, describing the transmission cost and fusion cost on the edge, respectively.

2.2 Transmission and Fusion Costs

Transmission cost, $t(e) : E \rightarrow \Re^+$ denotes the cost for transmitting $w(e)$ amount of data from u to v. We abstract the unit transmission cost on edge e as $c(e)$ and thus the transmission cost $t(e)$ of edge e is given by

$$t(e) = w(e)c(e). \tag{1}$$

Notice that $c(e)$ is edge-dependent and hence can accommodate various conditions per link, for example, different distances between nodes and local congestion situations.

Fusion cost, $f(e) : E \rightarrow \Re^+$ denotes energy consumption for fusion process at the *end* node v. $f(e)$ depends on the amount of data to be fused as well as the algorithms utilized. In this paper, we use a variable q to abstract the unit fusion cost. Then the cost for fusing the data of nodes u and v is $f(e) = q \cdot \Big(w(u) + w(v)\Big)$. Notice q is variable from edge to edge and dependent on the amount of data to be fused. Since data fusion is performed by intermediate nodes to aggregate their own data with their children's, in order to avoid confusion, we use $\widetilde{w}(\cdot)$ to denote the weight of a node *before current data fusion*. Thus, the above fusion cost can be rewritten as

$$f(e) = q \cdot \Big(w(u) + \widetilde{w}(v)\Big), \tag{2}$$

if node v is the fusion point.

Although both transmission and fusion costs are link-based, we remark that they cannot be simply combined together and hence rely on existing techniques solely based on the transmission cost to solve this problem. The reason is that the fusion cost on an edge is determined by the inputs of the fusion function. The inputs include both the incoming data from other nodes and the data produced by the fusion point itself. On the contrary, the transmission cost on an edge is only determined by the weight of the start point of the edge. In other words, for a fusion point, the transmission cost is only determined by the output of the fusion function. More evidently, this can be seen from Equations (1) and (2).

2.3 Correlation and Data Aggregation

In-network data aggregation captures the redundancy among data collected by different sensors and consequently aims at load reduction over the network. We assume that data aggregation can potentially take place at *any* intermediate node along the route: an intermediate node can explore the redundance among multiple child-nodes' data and aggregate all into one compressed data stream.

Key to a sensor data routing protocol is the data aggregation ratio. Unfortunately, this ratio is heavily dependent on application scenarios. Here, we use an abstract parameter σ to denote the data reduction ratio due to aggregation. To be more specific, if node v is responsible for fusing node u's data (denoted by $w(u)$) with its own, we have $w(v) = (w(u) + \widetilde{w}(v))(1 - \sigma_{uv})$, where $\widetilde{w}(v)$ and $w(v)$ denote the data amount of node v before and after fusion. Evidently, σ_{uv} is dependent on the correlation between the sensed data of u and v: larger correlation will result in more data reduction and hence larger σ_{uv}.

Due to aggregation cost, node v may choose not to perform data aggregation in order to realize maximum energy saving. Instead, it will simply relay the incoming data of node u. In this case, the new weight of node v is simply $w(v) = (w(u) + \widetilde{w}(v))$.

Jointly considering both cases described above, we can summarize the aggregation function at node v as

$$w(v) = (w(u) + \widetilde{w}(v))(1 - \sigma_{uv}x_{uv}), \tag{3}$$

where $x_{uv} \in \{0, 1\}$ denotes whether fusion occurs on edge $e = (u, v)$.

2.4 Problem Formulation

Given the source node set S and sink t, our objective is to design a routing algorithm that minimizes the energy consumption when delivering data from all source nodes in S to the sink t. Not only do we need to design routing paths back hauling sensed information driven by information aggregation, but also we have to optimize over the decisions as to whether aggregation shall occur or not on a particular node.

Mathematically, a feasible routing scheme is a connected subgraph $G' = (V', E')$ where $G' \subset G$ contains all sources ($S \subset V'$) and the sink ($t \in V'$). Depending on whether fusion is performed or not, the edge set E' can be divided into two disjoint subsets E'_f and E'_n, where $E'_f = \{e | e \in E', x_e = 1\}$ and $E'_n = \{e | e \in E', x_e = 0\}$. Our goal is to find a feasible subgraph G^* such that

$$G^* = \text{argmin}_{G'} \sum_{e \in E'_f} \Big(f(e) + t(e)\Big) + \sum_{e \in E'_n} t(e) \tag{4}$$

It is evident that a non-tree solution costs more than a routing scheme based on a tree structure [8, 1]. It has been shown that even if only transmission cost is considered, the aforementioned problem is NP-complete [2]. Heuristic algorithms have therefore been designed in literature for finding approximations to the minimum transmission cost tree [15, 2]. Since a new fusion cost is incorporated into our design, the resulting problem is NP-hard.

2.5 Discussion

If it is the policy of the network that every node performs fusion, the choice on fusion will be eliminated which results in $E'_n = \emptyset$. Thus the objective function in Equation (4) will degenerate to

$\sum_{e \in E'}(f(e) + t(e))$. Towards this end, MFST [9] provides an approximation routing algorithm that jointly optimizes over both the transmission and fusion costs. It has been shown in [9] that the resultant routing tree is bounded within $\frac{5}{4}\log(n+1)$ ratio to the optimal solution.

However, from the previous section, we have concluded that this approach is not necessarily to the best interest of the network. To incorporate the flexibility of fusion decision, which undoubtedly will affect the routing structure in turn, we design in this paper *Adaptive Fusion Steiner Tree (AFST)* algorithm that adaptively adjusts the fusion decision in the network based on data correlation, fusion cost, transmission cost, and network topology.

As AFST is based on MFST, in the next section we will give a brief overview of MFST. Following that, we will detail our new solution.

3. BRIEF OVERVIEW OF MFST

3.1 MFST Algorithm

The minimum fusion steiner tree (MFST) is based on the techniques presented in [11, 3]. It first pairs up source nodes (or source with the sink) based on defined metrics and then randomly selects a center node from the node-pair. The weight of the non-center node will be transferred to the center node, paying appropriate transmission and fusion costs on that edge. Subsequently, the non-center node will be eliminated and the center node with aggregated weight will be grouped as a new set of sources. This process will then be repeated on the new set until only the sink is left. The algorithm is detailed below for the sake of completeness.

MFST ALGORITHM:

1. Initialize the loop index $i = 0$. Define $S_0 = S \cup \{t\}$, and $E^* = \emptyset$. Let $w_0(v)$ for any $v \in S$ denote its original weight, and let $w_0(t) = 0$.

2. For every pair of nodes $(u, v) \in S_i$, find the minimum cost path (u, v) in G according to the metric
$$M(e) = q(w_i(u) + w_i(v)) + \alpha(w_i(u), w_i(v))c(e) \quad (5)$$
where $\alpha(w_i(u), w_i(v)) = \frac{w_i(u)w_i(v)(w_i(u)+w_i(v))}{w_i^2(u)+w_i^2(v)}$ for non-sink pair (u, v), and $\alpha(w_i(u), w_i(v)) = w_i(u)$ if v is t. Define $K_i(u, v)$ to be the distance under metric $M(e)$ of this path.

3. Find minimum-cost perfect matching[2] between nodes in S_i. Let $(u_{i,j}, v_{i,j})$ denote the jth matched pair in S_i, where $1 \leq j \leq |S_i|/2$. If there is only one non-sink node left after matching, match it to itself without any cost, and consider it as the last "single-node pair" in S_i.

4. For each matched pair (u, v), add those edges that are on the path defining $K_i(u, v)$ to set E^*.

5. For each pair of non-sink matched nodes (u, v), choose u to be the center with probability $\frac{w_i^2(u)}{w_i^2(u)+w_i^2(v)}$. Otherwise, node v will be the center. For pair (u, t), choose t to be the center.

6. Transport weight of non-center node to its corresponding center node. The weight of the center satisfies $w_{i+1}(center) = (w_i(u) + w_i(v))(1 - \sigma_{uv})$.

7. Remove all non-center nodes from S_i, then the remaining center nodes induce S_{i+1}.

8. If S_{i+1} contains only the sink, we return $G^* = (V^*, E^*)$, where E^* is the set of edges we constructed and V^* includes the source nodes and the sink. Otherwise increment i and return to step 2.

[2]Minimum-cost perfect matching is a matching of edges in a graph that guarantees the total cost (distance) for all pairs under $M(e)$ is minimized. For polynomial-time algorithms for this problem, see [12].

3.2 Discussion

MFST jointly considers both fusion and transmission costs. It has been shown that it yields $\frac{5}{4}log(n+1)$ approximation ratio to the optimal solution. Although extensive experiments [9] have shown that MFST can outperform other routing algorithms including SLT, SPT, and MST, one optimizing dimension is still missing, namely the aforementioned fusion decisions at sensor nodes. As MFST requires fusion to be performed along a routing path whenever possible, unnecessary energy may be wasted due to the inefficiency of fusion, for example, little information reduction due to weak correlation and high fusion cost.

Specially, this phenomena can be magnified in the proximity of the sink itself. As aggregated information streams are approaching the sink, their correlation decreases which will introduce small data reduction owing to fusion. At the same time, directly relaying the data will not incur high communication cost as fewer hops are needed for relaying. Naturally, in this scenario, there is a high probability for direct relaying to outperform aggregation.

To solve this problem, we design AFST in the next section where adaptive fusion decisions will be incorporated into the routing construction process.

4. DESIGN AND ANALYSIS OF AFST

In this section, we first introduce fusion decisions into the routing structure generated by the MFST scheme. To solve this optimization problem, by exploiting certain network properties, we propose a heuristic solution termed *Binary Fusion Steiner Tree* (BFST), which is analytically shown to have better performance than MFST. However, BFST is still constrained to the tree structure from MFST. By employing SPT rather than the structure obtained via MFST, where proper, we improve BFST to an enhanced algorithm called *Adaptive Fusion Steiner Tree* (AFST). In turn, AFST is analytically shown to be capable of achieving better performance than BFST.

4.1 3-D Binary Tree Structure

In order to facilitate our development from MFST to AFST, we first employ a 3-D binary tree structure to describe the process of the hierarchical matching technique used in MFST. Fig. 2 illustrates a mapping example of original aggregation tree and its transformation. From bottom to top, the edges between two layers represent the result of node matching and center selection in each iteration of MFST.

Assuming there are n sources in the aggregation tree G^* obtained via MFST, to perform the transformation, we first clone $N = \lceil \log(n+1) \rceil$ copies of G^*, denoted by $G^1, G^2, \ldots, G^N$. For convenience, we label the original G^* as G^0 and arrange them into vertical layers as shown in Fig. 2(b). For simplification, we will refer node v's clone in layer k as v_k. Subsequently, we map the original aggregation tree G^* as shown in Fig. 2(a) to a new graph $\hat{G}^*$ embedded into these clones. $\hat{G}^*$ is the targeted 3-D binary tree. The process is to map the result of each matching stage k in MFST to the edges between G^k and G^{k+1}. If we match u and v in stage k and v is selected as the center node in MFST, we first add an edge e linking its clones in G^k and G^{k+1} with zero unit transmission cost $c(e) = 0$. This edge is termed virtual edge (dotted line). Then we

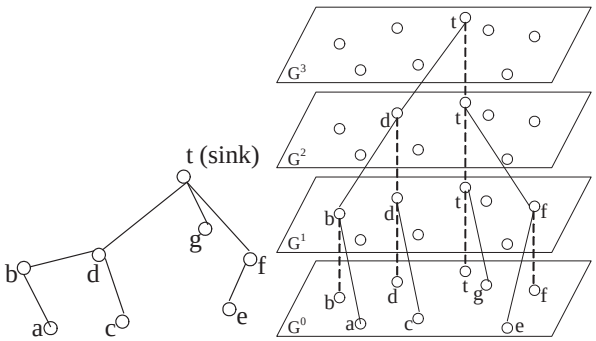
connect u_k to v_{k+1} with the same unit communication cost $c(e)$ as in G^*.

The result of this transformation is a binary tree which is rooted at the sink in G^N and has all leaves in S residing in G^0. For each source $v \in S$, there is a path going through all clones of G^*, from v in G^0 to the sink in G^N, via exactly N hops.

In order to guarantee that the resulting 3-D tree is binary, the number of source nodes is assumed a power of 2 minus 1. If this is not the case, dummy nodes with weight 0 can be created as needed to complement the binary tree, corresponding to the single node pair matching cases. These dummy nodes have aggregation ratio 0 with any other nodes and incur zero fusion cost. As virtual edges and dummy nodes/edges incur zero cost for communication and fusion, the 3-D binary tree is equivalent to the original aggregation tree.

For each node $v_k \in \hat{G}^*$ (the 3-D binary tree) in layer k, let w_{v_k} denote the weight of node v after combining the incoming data from lower layers (with or without data fusion). Each edge $e_k = (u_k, v_{k+1}) \in \hat{G}^*$, between the k-th layer and the $(k+1)$-th layer, is characterized by four parameters: edge weight $w_{e_k} = w_{u_k}$, unit transmission cost c_{e_k}, data aggregation ratio σ_{e_k}, and unit fusion cost q_{e_k}. Note that virtual edges have $c_{e_k} = 0$ and $\sigma_{e_k} = 1$ (full aggregation), and dummy edges have $w_{e_k} = 0$ and $\sigma_{e_k} = 0$ (no aggregation).

To incorporate fusion decision, we use $x_{e_k} \in \{0, 1\}$ to represent whether or not information on edge e_k is fused at the end node of this edge. Therefore, the optimal routing structure is an optimization problem over x_{e_k} as well as the tree structure. Towards this end, we present *Binary Fusion Steiner Tree* (BFST), an approximate solution to this routing problem.



(a) Original aggregation tree (b) Equivalent 3D aggregation tree

Figure 2: Expression of data aggregation tree in a 3-D binary tree structure. Solid lines in (b) correspond to edges in (a) and dotted lines represent virtual edges.

4.2 Binary Fusion Steiner Tree (BFST)

In BFST, we first obtain a routing tree using MFST algorithm, where fusion is performed by any intermediate node. Subsequently, we evaluate whether fusion on individual nodes will reduce the energy consumption of the network. If not, the fusion process on the node will be cancelled and instead data will be directly relayed.

BFST ALGORITHM:

1. Run MFST algorithm to obtain routing tree with fusion at every node possible. Convert the resulting aggregation tree to the 3-D binary tree as described above. For all edges in the tree, set $x_{e_k} = 1$.

2. From bottom to top, calculate the fusion benefit for each edge in the aggregation tree (excluding virtual edges), which can represent the energy saving by data fusion on that edge. Let $\Delta_{u_k,v_{k+1}}$ denote the fusion benefit of edge $e_k = (u_k, v_{k+1})$. It is defined as

$$\begin{aligned}\Delta_{u_k,v_{k+1}} &= (w_{u_k} + w_{v_k})C(v_{k+1}, t_N) \\ &- \Big((w_{u_k} + w_{v_k})(1 - \sigma_{e_k})C(v_{k+1}, t_N) \\ &+ q_{u_k,v_{k+1}}(w_{u_k} + w_{v_k})\Big) \qquad (6)\end{aligned}$$

   where $C(v_{k+1}, t_N)$ denotes the summation of unit transmission costs from v_{k+1} to t_N in MFST. if $\Delta_{u_k,v_{k+1}} > 0$, set $x_{e_k} = 1$; otherwise, set $x_{e_k} = 0$.

3. For all edges with $x_{u_k,v_{k+1}} = 0$, set $x_{v_k,v_{k+1}} = 0$ to their corresponding virtual edges.

In our analysis, we will employ the 3-D binary tree described in the previous subsection. To simplify the analysis, we assume that in BFST, the data reduction ratio σ is non-increasing on each path from the source to the sink while the unit fusion cost q is non-decreasing, excluding virtual edges. This assumption can be naturally justified. First, strong correlation and thus high aggregation ratio usually are due to spatial correlation resulting from short distances between nodes. In turn, these short distances will lead to small unit transmission cost. Based on the metric $M(e)$ defined in MFST (which is a combination of fusion cost and transmission cost), it will match strongly correlated nodes before matching weakly correlated nodes. Therefore, for edges on a source-sink path, the aggregation ratio for edges near the sink will not be larger than those further away. Reflected on the 3-D binary tree, this will lead to non-increasing σ on a particular source-sink path. The reason for skipping virtual edges is that their data aggregation ratio is set to 1 and does not affect the actual energy consumption of the network. Second, the unit fusion cost q is determined mainly by the complexity of the fusion algorithm and the input data set. As the information is being routed toward the sink, the data size and complexity will naturally increase due to aggregation on the route. Therefore, performing fusion thereon will incur more computation and hence more energy consumption per unit data.

Based on this assumption, we first introduce Lemma 1.

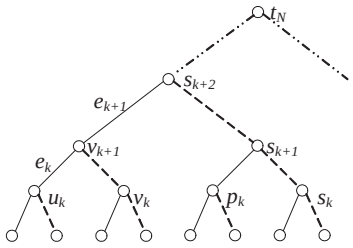


Figure 3: A fraction of the binary fusion tree for BFST: solid lines represent actual edges in the aggregation tree; dotted lines denote virtual edges added for analysis, and dash-dotted lines are paths to the sink.

LEMMA 1. *x_e is non-increasing on each path from a source to the sink in BFST.*

PROOF. Let Fig. 3 represent a branch of the binary fusion tree produced by BFST. On any path from a source node to the sink, assume e_k is the first edge with $x_{e_k} = 0$. We will enumerate different cases.

Case 1: If both e_k and its succeeding edge e_{k+1} are not virtual edges, as exemplified by $e_k = (u_k, v_{k+1})$ and $e_{k+1} = (v_{k+1}, s_{k+2})$

shown in Fig. 3, we have $\Delta_{u_k,v_{k+1}} \leq 0$. From Equation (6), we have $\sigma_{u_k,v_{k+1}} C(v_{k+1}, t_N) \leq q_{u_k,v_{k+1}}$. As $\sigma_{u_k,v_{k+1}} \geq \sigma_{v_{k+1},s_{k+2}}$ and $q_{u_k,v_{k+1}} \leq q_{v_{k+1},s_{k+2}}$ based on our assumption, and the total unit transmission cost from v_{k+1} to t_N is more than that from s_{k+2} to t_N, i.e., $C(v_{k+1}, t_N) > C(s_{k+2}, t_N)$, we can infer that $\sigma_{v_{k+1},s_{k+2}} C(s_{k+2}, t_N) < \sigma_{u_k,v_{k+1}} C(v_{k+1}, t_N) \leq q_{v_{k+1},s_{k+2}}$. This will lead to $\Delta_{v_{k+1},s_{k+2}} < 0$. Then we have $x_{v_{k+1},s_{k+2}} = 0$.

If e_k is not a virtual edge but its succeeding edge e_{k+1} is, as exemplified by $e_k = (p_k, s_{k+1})$ and $e_{k+1} = (s_{k+1}, s_{k+2})$, the same conclusion can be obtained similarly.

Case 2: If e_k is a virtual edge, as exemplified by $e_k = (v_k, v_{k+1})$, according to BFST algorithm, its matching pair edge $e'_k = (u_k, v_{k+1})$ must have $x_{u_k,v_{k+1}} = 0$. From the result of Case 1, we also have $x_{v_{k+1},s_{k+2}} = 0$.

Inductively, we can conclude that all succeeding edges of an edge that does not perform fusion will not perform fusion either. Since the fusion decision $x_{e_k} \in \{0, 1\}$, it is evident that x_e is non-increasing on the path from source to sink. □

THEOREM 1. *The total cost of BFST is no more than MFST.*

PROOF. Since BFST retains the same tree structure as MFST, for all edges with $x_{e_k} = 1$, the BFST and MFST schemes will consume the same amount of energy. For any edge with $x_{e_k} = 0$, owing to Lemma 1, any edge e_i on the path from this edge to the sink satisfy $x_{e_i} = 0$. This means that all edges on the path after e_k have negative effect on energy conservation. In other words, performing fusion will introduce additional cost. Therefore, BFST is a better algorithm than MFST by avoiding fusion when direct relaying is a better choice. □

Intuitively, Lemma 1 depicts that the routing tree generated by BFST can be divided into two parts: the lower part where data aggregation is always performed and the upper part where direct relaying is employed. As no data aggregation is performed in the upper part of the tree, instead of sticking to MFST, we can further improve the routing structure to reduce energy consumption. Inspired thereby, we develop AFST.

4.3 Adaptive Fusion Steiner Tree (AFST)

AFST further improves BFST by introducing SPT into the routing tree. Similar to BFST, it performs a matching process as in MFST in order to jointly optimize over both transmission and fusion costs. During the matching process, it also dynamically evaluates if fusion shall be performed or not. If it is determined at a particular point that fusion is not beneficial to the network, as shown by the analysis of BFST, we can conclude that any succeeding nodes on the routing path shall not perform fusion either. Consequently, we can employ shortest path as the strategy for the remainder of the route as shortest path is optimal for routing information without aggregation. Our analysis shows that AFST achieves better performance than BFST and thereon MFST.

Below, we summarize AFST algorithm. The process is similar to MFST but with fusion decision incorporated.

AFST ALGORITHM:

1. Initialize the loop index $i = 0$. Define $S_0 = S \cup \{t\}$, and $E^* = \emptyset$. Let w_{v_0} for any $v \in S$ equal to its original weight, and let $w_{t_0} = 0$.

2. For every pair of nodes $(u, v) \in S_i$, find the minimum cost path (u, v) in G according to the metric
$$M(e) = q_{u_i,v_{i+1}}(w_{u_i} + w_{v_i}) + \alpha(w_{u_i}, w_{v_i})c(e)$$
and define $K_i(u, v)$ to be the distance under metric $M(e)$ of this path.

3. Find minimum-cost perfect matching between nodes in S_i. If there is only one non-sink node left after matching, match it to itself without any cost, and consider it as the last "single-node pair" in S_i.

4. For each matched pair (u, v), calculate the fusion benefit for node u and v respectively according to this new definition:
$$\begin{aligned}\Delta_{u_i,v_{i+1}} &= (w_{u_i} + w_{v_i})SP(v_i, t) \\ &\quad -\Big((w_{u_i} + w_{v_i})(1 - \sigma_{e_k})SP(v_i, t) \\ &\quad +q_{u_i,v_{i+1}}(w_{u_i} + w_{v_i})\Big), \end{aligned} \tag{7}$$
where $SP(v_i, t)$ denotes the summation of unit transmission cost from v_i to the sink t using shortest path.
We call (u, v) a non-fusion pair if there is no fusion benefit regardless which node is selected as the center. It means that the two following inequations are satisfied
$$\Delta_{u_i,v_{i+1}} < 0 \quad \text{and} \quad \Delta_{v_i,u_{i+1}} < 0. \tag{8}$$
Otherwise, we call them a fusion pair.

5. For each non-fusion pair (u, v), add those edges that are on the shortest paths of (u, t) and (v, t) to set E^*_n, remove both nodes u and v from S_i.

6. For each fusion pair (u, v),
   - Add those edges that are on the path defining $K_i(u, v)$ to set E^*_f .
   - choose u to be the center with probability $\frac{w^2_{u_i}}{w^2_{u_i} + w^2_{v_i}}$. Otherwise v will be the center. For pair (u, t), choose t to be the center.
   - Transport weight of non-center node to its corresponding center node. According to Equation (3), the weight of the center satisfies $w_{i+1}(center) = (w_{u_i} + w_{v_i})(1 - \sigma_{u_i,v_i})$.
   - Remove all non-center nodes from S_i, then the remaining center nodes induce S_{i+1}.

7. If S_{i+1} is empty or contains only the sink, return $G^* = (V^*, E^*_f + E^*_n)$, where E^*_f and E^*_n are the sets of fusion edges and non-fusion edges, respectively, and V^* includes source nodes and the sink. Otherwise, increment i and return to step 2.

The size of set S_i is reduced at least half after one run of the algorithm. However, the process may terminate sooner than MFST and BFST if fusion is deemed unworthy in the early iterations.

THEOREM 2. *The total cost of AFST is no more than BFST.*

PROOF. The tree resulting from AFST also contains a lower part where aggregation is always performed and an upper part where no aggregation occurs. The lower part of AFST is the same as that of BFST due to their MFST based matching procedure and thus incurs the same cost as well. The task left is then to show that for any non-fusion pair (u, v) satisfying inequality (8), their transmission costs based on SPT in AFST is no more than the corresponding routing costs, including fusion and transmission costs, incurred in BFST. The proof is given below.

From (8), we have $\sigma_{u_k,v_{k+1}} SP(v_k, t) < q_{u_k,v_{k+1}}$ and $\sigma_{v_k,u_{k+1}} SP(u_k, t) < q_{v_k,u_{k+1}}$ where $SP(v_k, t)$ denotes the summation of unit transmission cost from v_k to the sink t using shortest path. Without loss of generality, we assume that v is selected as the center in BFST, and our goal is to prove $w_{v_k} SP(v_k, t) +$

$w_{u_k}SP(u_k,t) < w_{u_k}c(u_k,v_{k+1})+q_{u_k,v_{k+1}}(w_{u_k}+w_{v_k})+(w_{u_k}+w_{v_k})(1-\sigma_{u_k,v_{k+1}})SP(v_{k+1},t)$. For that, we have

$$\begin{aligned} & w_{v_k}SP(v_k,t)+w_{u_k}SP(u_k,t) \\ \leq\ & w_{v_k}SP(v_{k+1},t)+w_{u_k}(c(u_k,v_{k+1})+SP(v_{k+1},t)) \\ =\ & (w_{u_k}+w_{v_k})SP(v_{k+1},t)+w_{u_k}c(u_k,v_{k+1}) \\ =\ & (w_{u_k}+w_{v_k})(1-\sigma_{u_k,v_{k+1}}+\sigma_{u_k,v_{k+1}})SP(v_{k+1},t) \\ & +w_{u_k}c(u_k,v_{k+1}) \\ \leq\ & (w_{u_k}+w_{v_k})(1-\sigma_{u_k,v_{k+1}})SP(v_{k+1},t) \\ & +q_{u_k,v_{k+1}}(w_{u_k}+w_{v_k})+w_{u_k}c(u_k,v_{k+1}) \end{aligned}$$

□

When it is determined that the fusion benefit is positive for every node, the tree structure obtained from AFST degenerates to the tree from MFST. However, when fusion is not always beneficial for all nodes, AFST will significantly outperform MFST, which will be demonstrated in our experimental study.

4.4 Application in Clustering

Clustering in sensor networks is often used to group closely correlated sensor together in order to perform local decision making, detection, or classification. To facilitate these operations, fusion and transmission cost shall be considered due to severe resource constraints. While the concept of clustering has been widely applied, clustering itself is often based on static techniques mainly based on geographic proximity, fixed cluster number, or certain cluster sizes.

The two-layer structure of AFST naturally provides a new clustering technique for sensor networks. Separated by the routing nodes not performing fusion, the lower part of AFST routing tree is composed of discrete branches within which data of every member will be fused together. These branches can be considered as clusters decided by energy consumption due to fusion and communication. Therefore, AFST provides a clustering algorithm as a byproduct that is energy efficient for gathering correlated data. Sensors in the same cluster will employ MFST based routing structure while cluster heads (roots of the branches) will directly send the aggregated data to the sink via shortest path without fusion. Since AFST is a randomized algorithm, we can rerun the process to generate a different structure and therefore re-assign the role of cluster heads to different nodes. Load balancing in terms of fusion is thus naturally implemented.

5. EXPERIMENTAL STUDY

In this section, we compare the performance of AFST with other routing algorithms. In particular, we select MFST, SPT, MST, and SLT to represent the class of routing schemes where fusion occurs on all routing nodes if possible. For routing schemes that does not perform aggregation, we employ SPT as it is the optimal routing strategy in this class. To distinguish it from the SPT scheme with data aggregation opportunistically occurs where information streams intersect, we denote the SPT without performing aggregation by SPT-nf, short name for SPT-no-fusion.

We study the impact of network connectivity, correlation coefficient, and unit fusion cost on different algorithms. Concurring with our design goal and analysis of the AFST algorithm, our key finding of the experiments is that AFST can adapt itself to a wide range of data correlations and fusion cost. While other algorithms may achieve better performance in some extreme cases, they suffer from varying situations and hence perform poorly in general scenarios. Furthermore, AFST can adaptively adjust the number of fusion nodes according to the varying network situations, which further improves its performance especially when compared to MFST.

5.1 Simulation Environment

In our setup, 100 sensor nodes are uniformly distributed in a region of a $50m \times 50m$ square. We assume that each node sends one 400-byte packet as original sensed data to the sink at a corner of this square in each round. All sensors act as both sources and routers. We also performed sets of experiments of different number of sensors and different sizes of field, the results are similar and omitted here.

We assume the maximal communication radius of a sensor is r_c, i.e, if and only if two sensor nodes are within r_c, there exists a communication link between them. By varying r_c, we can control the network connectivity and hence equivalently the network density. Based on the model presented in [5], we instantiate unit transmission cost on each edge, $c(e) = \beta d^{\gamma} + \varepsilon$, when node distance $d < r_c$. We set $\gamma = 2$ and $\beta = 100pJ/bit/m^2$ to calculate the energy consumption on the transmit amplifier. ε denotes energy consumption per bit on the transmitter circuit and receiver circuit. Typical values of ε range from 20 to $200nJ/bit$ according to [18]. We set it to be $100nJ/bit$ in our simulation.

The correlation model employed here is an approximated spatial model where the correlation coefficient (denoted by ρ) decreases with the distance between two nodes provided that they are within the correlation range, r_s. If two nodes are more than r_s distance apart, simply the correlation coefficient $\rho = 0$. Otherwise, it is given by $\rho = 1 - d/r_s$, where d denotes the distance between the nodes. If node v is responsible for fusing node u's data (denoted by $w(u)$) with its own, we assume that the weight of node v after fusion is given by

$$w(v) = \max(w(u), \widetilde{w}(v)) + \min(w(u), \widetilde{w}(v))(1 - \rho_{uv})$$

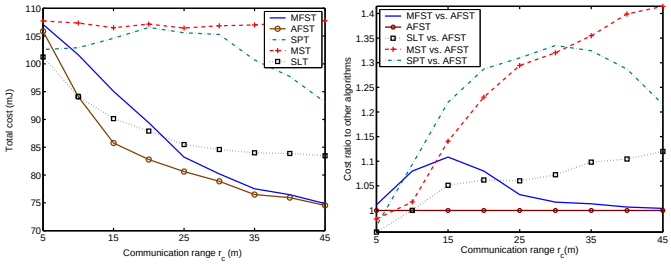
where $\widetilde{w}(v)$ and $w(v)$ respectively denote the data amount of node v *before* and *after* fusion.

Recall that we use σ to denote the data reduction ratio due to aggregation. From Equation (3), $w(v) = (w(u) + \widetilde{w}(v))(1 - \sigma_{uv})$, we can get that the data reduction ratio σ_{uv} is proportional to the correlation coefficient ρ. By varying the correlation range r_s, we can control the average correlation coefficient of the network, and further control the average data reduction after data fusion. For example, a very small r_s essentially eliminates the correlation among sensors ($\rho \to 0$), so that the amount of output data is equal to the summation of all input data. While an extremely large r_s makes the sensed data completely redundant ($\rho \to 1$), as a result, the fused data size equals to the bigger data size between the two input data. In our simulation, we use ρ instead of σ to describe the impact of data structure for ease of understanding. For the fusion cost, we assume that q is constant and use ω to denote the average fusion cost per bit at each node.

5.2 Impact of Network Connectivity

By varying r_c, the communication range of a node, we can control the connectivity of the network. The larger r_c is, the more strongly the network is connected. Here we set the average unit fusion cost to be $80nJ/bit$,which is a typical value for image fusion [10], and set the correlation range to be $50m$ to simulate a network with moderate data reduction. Fig. 4 summarizes the performance of all algorithms studied.

As we can see, MST wastes precious energy on data fusion at numerous relaying nodes and hence incurs high cost when the data reduction is not high. On the contrary, SPT wastes energy owing to redundant data transmissions and also induces high cost since it tends to use fewer hops to reach the sink. As SLT is a hybrid tree structure balancing MST and SPT, it achieves better performance than both of them. However, inherently constrained by its algorithm construction, energy consumption of SLT is still relatively high.



(a) Total cost (b) Cost ratio

Figure 4: Impact of network connectivity to energy consumption.

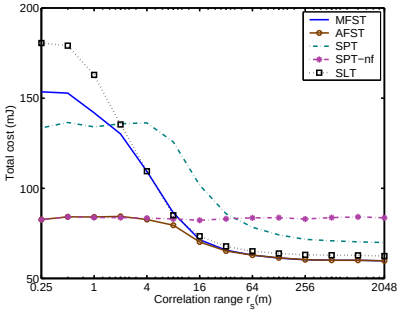


Figure 5: Impact of average correlation coefficient to energy consumption.

As MFST explicitly considers fusion cost, it can effectively trade off between multi-hop relay benefitting from high data reduction ratio and single-hop transmission benefitting from less fusion cost. In a dense network with strong network connectivity, the benefit due to this flexibility to the network energy cost is more apparent. When r_c is larger than 20m, MFST performs better than all other algorithms except AFST.

As expected, AFST performs well in all communication ranges, almost steadily outperforming all other algorithms. This can be explained as follows. In a network with moderate data correlation, data from nodes that are far apart has little correlation and hence leads to small reduction ratio if aggregated. Therefore, performing data fusion over them cannot significantly benefit the network but still incur the fusion cost. AFST can effectively avoid such situations by evaluating fusion benefit. If the data reduction ratio due to correlation together with the fusion cost could not justify the fusion process, it employs the shortest paths toward the sink by using direct relay. Therefore, the cost of AFST continuously decreases along with the increase of communication range. Notably, it has significantly less energy cost for weakly connected network resulting from short communication range as well, which can be reasoned in the same way. As shown in Fig. 4(b), the proposed AFST can save over 40% energy compared to MST, up to 35% energy to SPT, and about 10% to SLT when connectivity degree is high. Compared with MFST, AFST can save around 10% of energy in weakly connected environment while maintaining the same or better performance when network is strongly connected.

5.3 Impact of Correlation Coefficient

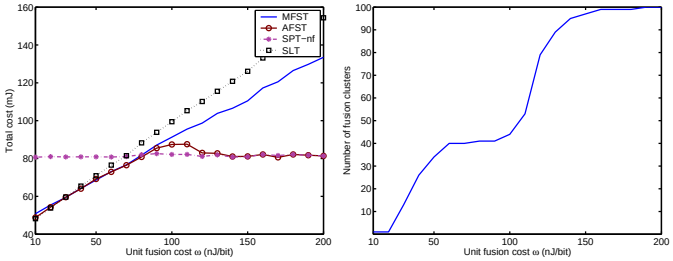
In this simulation, we fix the transmission range to $30m$ and unit fusion cost to $50nJ/bit$, and study the impact of correlation coefficient to the performance of AFST. We increase the correlation range, r_s, from 0.2 to $2000m$ which corresponds to varying ρ from 0 to 1.

Fig. 5 depicts the total energy costs for AFST and other algorithms like MFST, SPT, SLT, and SPT-nf. Naturally, costs of all algorithms with data fusion decrease as ρ increases. This exemplifies that data aggregation in sensor networks can greatly benefit the routing performance by reducing redundancy among correlated data. As SPT-nf totally ignores data aggregation, its cost remains constant.

When the data correlation in the network is very weak ($\rho \rightarrow 0$), AFST follows SPT-nf, the optimal solution, by avoiding any data aggregation. When the data correlation in the network increases, AFST dynamically adjusts its decisions accordingly by performing data fusion partially in order to benefit from data aggregation and resultant data reduction. When the data in network is highly correlated ($\rho \rightarrow 1$), AFST follows MFST to pursue the most energy saving by performing data fusion at each possible node. Due to this dynamic adjustment, the cost of AFST is extremely smooth in the whole range of the correlation coefficient and steadily outperforms others.

5.4 Impact of Unit Fusion Cost

In this set of experiments, we study the impact of varying unit fusion cost to the algorithms. Fig. 6 illustrates the results when ω, the unit fusion cost, increases from $10nJ/bit$ to $200nJ/bit$.



(a) Total cost (b) Number of fusion clusters

Figure 6: Impact of unit fusion cost to energy consumption($r_c = 30m$, $r_s = 20m$).

Fig. 6(a) shows the total cost of AFST as compared with other algorithms as the unit fusion cost increases. As we can see, the total costs of MFST and SLT increase unboundedly along with the increase of ω, even though MFST has a lower slope. On the contrary, AFST follows the performance curve of MFST first and then leans towards SPT-nf, the optimal solution when fusion cost is high. The figure can be best explained when we jointly examine it with Fig. 6(b), which depicts the number of clusters, the branches of the routing tree that always perform aggregation on their nodes. All algorithms with full fusion are unable to stop fusion even when fusion cost is extremely high. However, for AFST, as shown in Fig. 6(b), when ω is very small, there are only two fusion clusters. This denotes that data fusion is performed almost on all nodes, which takes advantage of the low fusion cost. When ω increases, AFST reduces the number of fusion clusters due to reduced fusion benefit in order to balance the fusion cost and transmission cost. And when ω is too large, AFST can achieve the same constant cost as SPT-nf by completely stopping data fusion.

As described in Section 2, fusion cost may vary widely from network to network, from application to application. Our experiments show that among all algorithms, AFST can adapt best to a wide range of fusion costs and hence be applicable to a variety of applications.

5.5 The Number of Fusion Clusters in AFST

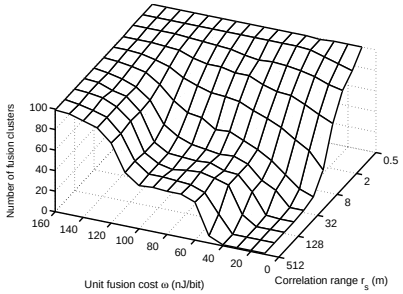


Figure 7: The number of fusion clusters in AFST.

Fig. 7 illustrates the number of fusion clusters with varying correlation coefficient and unit fusion cost. When $\rho \rightarrow 0$, the cluster number equals 100, denoting that no data fusion occurs. When the unit fusion cost is small, the number of fusion clusters decreases rapidly with the increase of ρ. When ρ approaches 1, the cluster number becomes 1, meaning that all nodes will perform fusion and the network becomes a full fusion tree. At the same time, when unit fusion cost increases, AFST will tradeoff the fusion cost with transmission cost. As a result, the decreasing of the cluster number slows down. For example, for $\omega = 80nJ/bit$, a unit fusion cost comparable to the average unit transmission cost, there are 33 fusion clusters when $\rho \rightarrow 1$. This means on average, there are only three nodes in a cluster performing data fusion. The fused data are then forwarded to the sink via shortest paths directly with simple relaying. If ω keeps increasing, fewer nodes perform fusion to avoid the high fusion cost.

6. CONCLUSION

In this paper, we propose AFST, a routing algorithm for gathering correlated data in sensor networks. AFST not only optimizes over both the transmission and fusion costs, but also adaptively adjusts fusion decisions for sensor nodes as to whether fusion shall be performed. Generalized from MFST, an algorithm guarantees an approximation ratio of $\frac{5}{4}\log(n+1)$ to the optimal solution, AFST is analytically shown to be a better algorithm than its ancestor. Extensive experiments show that AFST achieves near optimal solutions under various networking conditions, including broad scopes of fusion/transmisison costs and network/data structures.

Furthermore, our analytical result indicates that AFST partitions the routing tree into two distinct parts based on whether aggregation is performed or not. Naturally, AFST also provides a clustering scheme with near optimal routing performance, where fusion can be confined within the clusters only.

As an ongoing effort, we are quantifying theoretically the performance improvement of AFST over MFST. In our future work, we plan to develop an online algorithm based on AFST that can be executed in a distributed manner by sensor nodes.

7. REFERENCES

[1] M. Andrews and L. Zhang. The access network design problem. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, Palo Alto, CA, Nov. 1998.

[2] R. Cristescu, B. Beferull-Lozano, and M. Vetterli. On network correlated data gathering. In *Proceedings of IEEE Infocom*, Hongkong, China, Mar. 2004.

[3] A. Goel and D. Estrin. Simultaneous optimization for concave costs: Single sink aggregation or single source buy-at-bulk. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, Baltimore, MD, Jan. 2003.

[4] A. Goel and K. Munagala. Balancing steiner trees and shortest path trees online. In *Proceedings of the 11th ACM-SIAM symposium on Discrete Algorithms*, San Francisco, CA, Jan. 2000.

[5] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, Maui, HI, Jan. 2000.

[6] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann. Impact of network density on data aggregation in wireless sensor networks. In *Proceedings of ICDCS'02*, Vienna, Austria, July 2002.

[7] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions on Networking*, 11(1), Feb. 2003.

[8] B. Krishnamachari, D. Estrin, and S. Wicker. Impact of data aggregation in wireless sensor networks. In *Proceedings of the 22nd International Conference on Distributed Computing Systems*, Vienna, Austria, July 2002.

[9] H. Luo, J. Luo, S. K. Das, and Y. Liu. Routing correlated data with fusion cost in wireless sensor networks. *Submitted to IEEE Transactions on Mobile Computing*.

[10] J. Luo, H. Luo, and Y. Liu. Fusion cost for image data fusion in sensor networks. In *Technical Report*, CSE, UTA, Mar. 2005.

[11] A. Meyerson, K. Munagala, and S. Plotkin. Cost-distance: Two metric network design. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, Redondo Beach, CA, Nov. 2000.

[12] C. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications Inc, 1998.

[13] S. Pattem, B. Krishnamachari, and R. Govindan. The impact of spatial correlation on routing with compression in wireless sensor networks. In *Proceedings of IPSN'04*, Berkeley, CA, Apr. 2004.

[14] S. S. Pradhan and K. Ramchandran. Distributed source coding using syndromes (DISCUS): Design and construction. *IEEE Transactions on Information Theory*, 49(3), Mar. 2003.

[15] P. Rickenbach and R. Wattenhofer. Gathering correlated data in sensor networks. In *Proceedings of ACM DIALM-POMC'04*, Philadelphia, PA, Oct. 2004.

[16] B. R. S. Khuller and N. Young. Balancing minimum spanning and shortest path trees. In *Proceedings of the 4th ACM-SIAM symposium on Discrete Algorithms*, Austin, TX, Jan. 1993.

[17] A. Scaglione and S. D. Servetto. On the interdependence of routing and data compression in multi-hop sensor networks. In *Proceedings of ACM MobiCom*, Altanta, GA, Sept. 2002.

[18] A. Wang, W. B. Heinzelman, A. Sinha, and A. P. Chandrakasan. Energy-scalable protocols for battery-operated microsensor networks. *Journal of VLSI Signal Processing*, 29(3), Nov. 2001.

[19] Y. Yu, B. Krishnamachari, and V. Prasanna. Energy-latency tradeoff for data gathering in wireless sensor networks. In *Proceedings of IEEE Infocom*, Hongkong, China, Mar. 2004.