JOINT ROUTING AND 2D TRANSFORM OPTIMIZATION FOR IRREGULAR SENSOR NETWORK GRIDS USING WAVELET LIFTING

Godwin Shen and Antonio Ortega
Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, California, USA
godwinsh@usc.edu, ortega@sipi.usc.edu

Abstract

We address the joint optimization of routing and compression for wireless sensor networks using a lifting-based 2D transform that can be computed along arbitrary routing trees. The proposed 2D transform allows for unidirectional computation, thereby eliminating costly backward transmissions often required by existing 2D transforms. We also propose a framework for optimizing the transform by selecting among a different set of coding schemes (i.e., different levels in the wavelet decomposition). Since our transform can operate on arbitrary routing trees, we focus on the problem of jointly optimizing routing trees based on inter-node data correlation and inter-node distance. The two extreme solutions would be i) to route data along paths that maximize inter-node data correlation (at the risk of increasing transport costs), corresponding to a minimum spanning tree (MST), or ii) to follow shortest path tree (SPT) routing (where inter-node data correlation may not be as high). We propose an optimization technique that exhaustively searches for the optimal tree over a set of combinations of MST and SPT. We also propose a heuristic approximation algorithm that is amenable for use on larger networks and with which we observe total cost reductions close to 10% for some of the data.

1 INTRODUCTION

In recent years, low-cost, densely deployed Wireless Sensor Networks (WSN) have been studied for applications such as instrumentation and enviromental monitoring, among many others [2]. Because these are typically battery powered devices, it is important to find power-efficient techniques for data gathering and transmission. These techniques entail both energy-efficient routing and low-cost in-network compression.

A typical in-network compression technique collects data at individual nodes, then compresses it as it "flows" towards a fusion center or sink, along a series of pre-established paths. These approaches exploit spatial correlations in the data in order to reduce the overall number of bits needed to represent a snapshot of information sensed in the network. We focus here on *transform-based* in-network compression techniques, which include the distributed KLT [8] and wavelet based methods [1, 3–5, 16, 19, 20]. Obviously, the ultimate goal is to achieve *overall* power savings. In the case of transform techniques, a more efficient signal representation is computed in a distributed manner (by de-correlating data across neighboring nodes) resulting in fewer bits to transmit to the sink in exchange for some "local" communication overhead.

While existing transform-based methods are capable of reducing the number of bits to be transferred to the sink, almost all of them separate transform design and routing, i.e., they define transforms first then map those transforms onto efficient routing trees. In some cases, this requires nodes to transmit uncompressed data directly to a cluster head as in [8] or to a certain number of neighbors [19, 20] before transform coefficients can even be computed. If the neighbors (or cluster head) of a node are further away from the sink than the node itself, additional backward transmissions of uncompressed data will be required that increase the total cost. The method proposed in [5] eliminates backwards transmissions by computing coefficients in a unidirectional manner (as data flows towards the sink) along paths of a given routing tree. However, this results in an oversampled transform (the number of wavelet coefficients generated is greater than the number of samples captured) when multiple paths merge, resulting in additional energy consumption for merging paths. Instead in our recent work [17] we propose transforms that i) can be computed on *arbitrary routing trees*, and in particular suffer no penalty as multiple paths merge in the tree, and ii) do not require additional backward transmissions (the transform is computed in a unidirectional manner as data flows towards the sink). Our proposed transform is shown to be more energy-efficient than the bi-directional transform in [19, 20] and the path-wise unidirectional transform in [5]. However, like the other methods discussed, the method presented in [17] considers transform and routing separately, i.e., a shortest path routing tree is chosen first and then a transform is performed over that tree. Instead, the technique we propose here attempts to exploit the inherent interaction between different trees and the transform in [17]. This leads us to a practical approach for jointly optimizing compression and routing, i.e., we can aim at designing a tree with good transport cost *and* data correlation properties, knowing that no matter what tree is chosen the transform can be implemented.

A shortest path routing tree (SPT), guarantees that the path from a given node to the sink is most efficient for routing, but obviously does not guarantee that consecutive nodes in a path contain highly correlated data. For example, if data correlation is inversely proportional to distance between nodes, one would always have to route through the nearest neighbor in order to achieve maximal inter-node data correlation. Clearly SPT routing does not guarantee this, since this design aims to minimize distance to sink, not inter-node distance. The results in [14] corroborate this, where a network with high data correlation benefits most from routing and compression along shorter hops with longer overall paths. As an alternative, we could consider trees that link together nodes with high inter-node data correlation. Such trees can provide greater compression efficiency than an SPT. However, aggregating along these types of trees may force nodes to transmit data away from the sink, so that gains provided by the increase in de-correlation are offset by increased transmission cost. Since aggregation will occur along routing trees, there is a trade-off between trees that result in energy-efficient routing and ones that allow a transform to de-correlate data effectively.

In order to achieve jointly optimized routing and transform we search exhausitvely for the lowest cost tree among a set of possible trees, for a fixed distortion. In general, for a given tree T, we assume the cost to tranport one bit from a node n to the sink is a function of the edge weights in T along the path from n to the sink, which we denote $f(n)$. For example, in [17] we use as edge weights the squared inter-node distances and $f(n)$ is the sum of squared distances along each hop from n to the sink. Then the cost to transport b_n bits from node n to the sink is $b_n f(n)$. We also associate with each n a small local communication cost $l(n)$ incurred for unidirectional transform computation (as will be discussed in Section 2.2). Naturally, the local cost term $l(n)$ will vary depending on the structure of the tree (e.g., if n has multiple children) and on the number of levels of decomposition that n uses. The total cost for transforming and routing data for N nodes along T is then given by $C_T = \sum_{n=1}^{N} (b_n f(n) + l(n))$. This is the cost we seek to minimize, for fixed distortion. Since our chosen transform is computed along arbitrary trees, a natural optimization problem is to find a tree T that minimizes the total cost C_T for a fixed distortion.

While one could consider the full set of possible trees for a given communication graph, this set can be extremely large. The well-known matrix-tree theorem [11] (which provides the number of spanning trees for a given graph) implies that a complete graph with n nodes has n^{n-2} possible trees. Even if the graph is not complete, the matrix-tree theorem may still imply a very large solution space. Thus, it is not computationally feasible to consider a full solution set. To make the optimization problem tractable, we choose only to explore trees that can be obtained by combining links from an SPT computed with edges defined by physical inter-node distances (to minimize distance to the sink) with links from a minimum spanning tree (MST) computed with edge weights defined by inter-node data correlation (to maximize pair-wise inter-node correlation). More specifically, we design an MST using edge weights $w(m, n) = 1 - r_{m,n}$ with $r_{m,n}$ the correlation coefficient between nodes m and n so that an MST corresponding to these edge weights will have a link between each node n and the neighbor of n that has maximum inter-node data correlation with n. Clearly, such an MST is “best” in the sense of maximizing pair-wise data correlation along the tree, which should help achieve improved compression efficiency for our transform. Since the SPT will minimize the cost to route any amount of data from a node to the sink, we can use combinations of such an MST with an SPT to provide a direct trade-off between high compression performance and low routing cost.

To illustrate this point, consider the real network in Figure 1 taken from [13], where a combination of an SPT and MST is used for joint routing and compression. The SPT provides the shortest route to the sink from any node, but fails to link some nodes to their closest neighbors. This can reduce compression efficiency. The MST links those nodes to their closest neighbors, but also has some lengthy paths that push data away from the sink. Clearly, neither alone is sufficient to achieve the best joint routing and compression performance. Instead, the trees obtained by our proposed optimization methods tend to link nodes to their closest neighbor, but in a way that preserves short paths to the sink, resulting in improved overall performance, as will be shown in Section 4.3.

In summary, in this paper we address the joint optimization of routing and compression by using a distributed 2D wavelet transform. We first describe how our proposed transform can be applied along arbitrary routing trees. Then, we propose a technique for selecting routing and

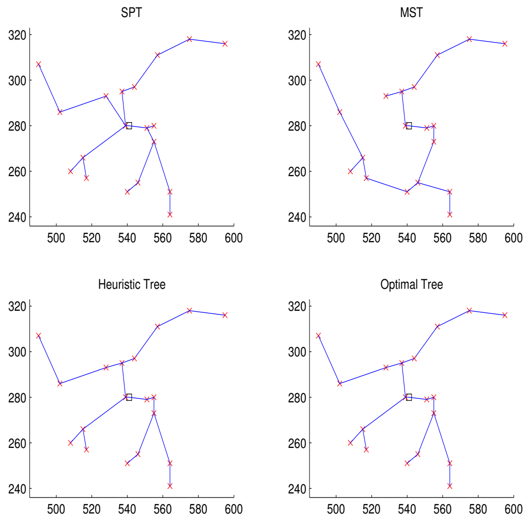


Figure 1. SPT, MST, and Combined Tree

transform jointly by accounting for both data correlation and routing costs.

1.1 Related Work

In the context of wavelet-based approaches [1, 3–5, 16, 19, 20], we consider the design of a unidirectional 2D transform (as in [17]) along routing trees in order to facilitate joint selection of routing and transform. An early proposal using wavelet transforms in WSN by Servetto [16] was based on the assumption that sensors are located in a regular grid, where wavelet transform techniques for 2D images provide a natural extension. However, these regular grid assumptions will not hold in general as illustrated in [7].

This has motivated researchers to investigage transforms that operate with irregular 2D node placements. Wagner *et al* [20] propose to apply a lifting transform along a tesselation of the nodes. This transform is applicable to arbitrary 2D node deployments, but it also requires backward data transmissions that flow away from the sink. Thus, the compression performance is good in terms of overall rate, but the transmission costs can be high. As an alternative, there have been proposals to use 1D wavelet transforms along the routing paths in the sensor network, in a way that explicitly computes the transform in a unidirectional manner [1, 5], i.e., the transform is computed as data flows towards the sink, thereby eliminating the overhead introduced in [19, 20]. In addition (unlike [19, 20]), [1, 5] provide techniques for optimizing the number of levels of decomposition. The main drawback of the approaches in [1, 5] is that they are essentially 1D transforms and, consequently, cannot exploit 2D spatial correlation as well as a fully 2D transform. Furthermore, in [5], simple ad hoc techniques are used to combine the information obtained from two or more merging paths along a route to the sink (on each of which a 1D transform has been computed), resulting in a non-critically sampled signal representation.

The transform we recently proposed in [17] addresses the disadavantages of existing transforms, while allowing the transform to be constructed on arbitrary trees. It is critically sampled, unlike that in [5]. It is also computed in a unidirectional manner, eliminating the backward transmissions required in [19, 20]. An algorithm is also provided that finds an optimal number of levels of decomposition for each node in order to exploit the tradeoff in [1, 5]. Note that the transform developed in [17] does not jointly optimize the routing and transform, but instead only optimizes the transform for a particular routing tree. The joint optimization problem is the main focus of this paper.

Mere transform design and optimization aside, only a few existing methods jointly optimize routing and compression. Optimal routing trees were found in [10], assuming concave aggregation functions. The aggregation functions, however, depend only on the number of nodes participating in aggregation which is not very realistic. A joint optimization technique using "foreign coding" was presented in [15], where data is encoded along a directed MST, and is forwarded along an SPT. On the other hand, we attempt to find a tree T which combines an MST and SPT such that performing both routing and transform along T minimizes the cost C_T (over the set of possible combinations). The edge weights used in [15] are also slightly different from ours since they start with an initial set of edge weights, which are modified by using the inter-node data correlation. Then an MST is constructed based on these modified edge weights, whereas we propose to use an MST constructed from edge weights that are directly a function of inter-node data correlation. Futhermore, this method assumes nodes will use side information from exactly one other node to compress their own data. Thus, nodes only encode their data using information from one other node, whereas in our method nodes will compress their own data using data from multiple neighbors (depending on the tree and number of levels of decomposition).

The work in [14] also studied the effects of data correlation on joint routing and compression decisions and an optimal clustering strategy was provided to maximize the benefits of this joint decision given a particular correlation level and network structure. However, no method is provided that finds a routing structure that is jointly optimal for routing and compression, which is exactly what we set out to do. Instead, nodes are grouped into an optimal set of equally spaced tiles and shortest path routing trees (defined for each tile) are used to perform joint routing and compression, and as pointed out before, these SPTs may not be best for exploiting compression. Results in [14] do, however, provide useful insights that we have drawn upon. In addition, both methods in [14, 15] utilize losseless data compression techniques which can be difficult to employ in practice. Instead, our method uses a practical, but lossy, compression method via the wavelet transform.

Therefore, we aim to achieve joint routing and compression optimization by combining the benefits of the various trees employed in existing work. In particular, we borrow insight from [14] by designing trees that exhibit properties beneficial to both routing and compression. We find such trees by combining trees best for compression (e.g., an MST as suggested in [15]) with trees best for routing. To achieve additional savings, we also use the idea proposed in [15] of overlaying an SPT (for efficient routing) over the transform tree so that fully computed coefficients are routed directly to the sink via the SPT. Naturally, our optimization accounts for this by computing the cost for such an overlay of trees (rather than the cost along just one tree). Our main goal is then met by selecting, among our set of possible trees, the tree that provides minimum cost transform and routing. To the best of our knowledge, no technique has been developed that jointly optimizes routing and compression using a unidirectional 2D wavelet transform for an irregular 2D node deployment.

The paper is organized as follows. The design and unidirectional computation of our proposed transform is presented in Section 2, where a general overview of the transform optimization framework is also provided. Our joint transform and routing optimization algorithm is then presented in Section 3. Section 4 provides experimental results and Section 5 concludes the paper.

2 TRANSFORM DESIGN USING LIFTING

In this section, we briefly review the design of the distributed 2D transform we proposed in [17]. Consider a sensor field with N nodes, where data x_n is captured by node $n \in \mathcal{I} = \{1, 2, \ldots, N\}$. Let us model the network as a graph $G = (V, E)$, where $V = \mathcal{I}$ and for any $m, n \in V$, (m, n) represents an edge in E. Let $T = (V_T, E_T)$ be a tree representing data routing through the network, with the root of the tree corresponding to the sink (indexed by $N + 1$) , and each leaf node representing the first sensor in a given path towards the sink and where $V_T \subset V$ and $E_T \subset E$. Let $\mathcal{C}_n$ and ρ_n denote the set of children and the parent of sensor n in T, respectively. Let depth(n) be the depth of node n in T, with depth$(N + 1) = 0$.

2.1 Lifting Transform Design

A lifting transform [18] can be performed once we define disjoint sets of prediction and update nodes at each level of decomposition j, denoted $\mathcal{P}_j$ and $\mathcal{U}_j$, respectively. Then given $\mathbf{p}_{n,j}$ and $\mathbf{u}_{m,j}$ as the prediction and update operators at nodes $n \in \mathcal{P}_j$ and $m \in \mathcal{U}_j$, respectively, we can compute the smooth coefficients given by $s_{m,j}$ and detail coefficients $d_{n,j}$ in a standard recursive manner [17].

We split the nodes into prediction and update sets according to their depth with respect to the root of the tree (the sink) which has depth zero, i.e., nodes of even and odd depth will be update and prediction nodes, respectively. To facilitate unidirectional transform computation along T as data flows towards the sink, we choose localized operators so that non-zero weights are assigned only to one-hop neighbors of a given node, i.e., $\mathbf{p}_{n,1}(k) = 0$ and $\mathbf{u}_{n,1}(k) = 0$ for $k \notin \mathcal{C}_n \cup \{n, \rho_n\}$. Note that nodes that are close to each other in T will also be close in physical distance. This will keep local costs low and will tend to produce data that is more correlated (thus, also providing a reasonable way to exploit signal correlation).

Figure 2 gives an example of the splitting tree for 2-levels. By extending to j-levels, T_j will consist of nodes of even depth in T_{j-1} with an edge between two nodes in T_j only if they are 2-hops apart in T_{j-1}. Then we can apply the same split method for each T_j. Clearly, $T = T_1$ under this construction. We also denote the children and parent nodes of $n \in T_j$ as $\mathcal{C}_{n,j}$ and $\rho_{n,j}$.

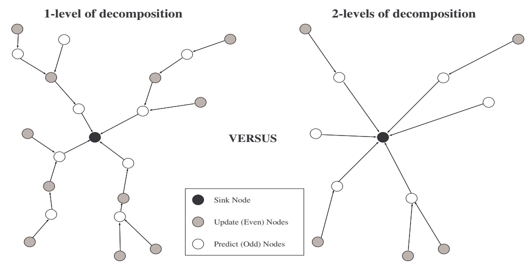


Figure 2. Trees used for splitting

2.2 Unidirectional Computation

For the sake of design simplicity, we propose the following filters for j-levels. For a node $n \in \mathcal{P}_j$, we generate a prediction by averaging data in neighboring nodes, i.e., $\mathbf{p}_{n,j}(m) = -\frac{1}{|\mathcal{C}_{n,j}|+1}$ for each $m \in \mathcal{C}_{n,j} \cup \{\rho_{n,j}\}$. and for all nodes $m \in \mathcal{U}_j$, we similarly perform smoothing by using the detail coefficients of its neighbors, i.e., $\mathbf{u}_{m,j}(k) = \frac{1}{2(|\mathcal{C}_{m,j}|+1)}$ for each $k \in \mathcal{C}_{m,j} \cup \{\rho_{m,j}\}$. Clearly we should have $\mathbf{p}_{n,j}(n) = \mathbf{u}_{m,j}(m) = 1$.

For j-levels of decomposition, the lifting transform in Section 2.1 can be computed as follows. For every $m \in \mathcal{P}_j$:

$$d_{m,j} = s_{m,j-1} + \sum_{k \in \mathcal{C}_{m,j}} \mathbf{p}_{m,j}(k) s_{k,j-1} + \mathbf{p}_{m,j}(\rho_{m,j}) s_{\rho_{m,j},j-1} \quad (1)$$

then, given each $d_{m,j}$, for all $n \in \mathcal{U}_j$ we have:

$$s_{n,j} = s_{n,j-1} + \sum_{m \in \mathcal{C}_{n,j}} \mathbf{u}_{n,j}(m) d_{m,j} + \mathbf{u}_{n,j}(\rho_{n,j}) d_{\rho_{n,j},j}. \quad (2)$$

This transform is invertible by construction and is also critically sampled. Defining the transform along an arbitrary tree also facilitates unidirectional computation since the terms corrsponding to the children and parent nodes in (1) and (2) are explicitly separated. Thus (as in [3]), a node can partially compute its own wavelet coefficient using the coefficients and/or data from its children, then can forward this partial coefficient to its parent where it will be updated or finalized. The details of this partial coefficient algorithm can be found in [17].

The partial coefficient algorithm in [17] can be a source of local communication overhead since a few additional bits are allocated to each partial to mitigate the effects of quantization as in [3]. However, this increase in cost is minimal since these added bits are only carried over a few hops. The $l(n)$ term in the cost equation C_T presented in Section 1 reflects these minor increases in cost.

2.3 Transform Optimization

As discussed earlier, a mixture of coding schemes throughout the network may be more energy-efficient than just one scheme, resulting in a tradeoff between more local transmission cost for lower final transport cost [1, 5]. However, the proposed optimization schemes in [1, 5] do not directly extend to overlapping 1D paths, so the problem must be reformulated for our proposed 2D transform. Following the notation in [17], the optimization is done by selecting (for each node $n \in \mathcal{I}$) a number of levels of decomposition $j_n \in \{1, 2, \ldots, J\}$ that minimizes the total energy consumption. This is done via a forward dynamic programming algorithm that sequentially computes, from nodes of greatest depth to nodes of depth one, the optimal cost to arrive at scheme j for every node n. For each node n of depth one in T, it then chooses the minimum cost scheme j_n^* and assigns to the descendants of n the optimal schemes that correspond to j_n^*. This results in an optimal network assignment.

3 JOINT ROUTING AND TRANSFORM OPTIMIZATION

Our proposed optimization method is inspired by the "foreign coding" technique developed in [15], where an MST is constructed with edge weights that are a function of data correlation and where data is encoded along this MST and forwarded along an SPT. In order to formulate our routing optimization, consider a graph with vertices corresponding to the node indices $m, n \in \mathcal{I}$ and edges (m, n) with weights $w(m, n)$ corresponding to some inverse measure of correlation between m and n. For instance, the edges weights could be $w(m, n) = 1 - r_{m,n}$, with $r_{m,n}$ the inter-node correlation coefficient between nodes m and n. We can then construct an MST from those edge weights that implicitly maximizes pair-wise data correlation, yielding a tree that provides good compression performance for our transform. To see why this is so, consider Prim's algorithm [6] for constructing an MST. Given a set of edge weights, Prim's algorithm can construct an MST by starting from any arbitrary node n. So by starting from n, the second node added to the MST will be a node m such that $w(n, m) \leq w(n, k)$, for all $k \in V \backslash \{n\}$. Since $w(n, m) = 1 - r_{n,m}$ is minimal over all edges containing n as a vertex, $r_{n,m}$ is maximal. Therefore, data at n is maximally correlated with data at m. Since this can be done for arbitrary n, our claim follows. It is worth noting here that an MST defined in this way only considers pair-wise correlation, and so is not necessarily optimal from a coding standpoint when using our proposed transforms (where data is filtered over multiple hops).

We could utilize Rickenbach's approach [15] for joint routing and compression (e.g. encode along MST and forward along SPT). However, this would require each node to transmit its data along its next hop in the MST which may force data to flow away from the sink. This can produce some inefficiency from a routing standpoint if the gains in coding efficiency do not offset the resulting increase in routing cost. Instead, our proposed approach can avoid such situations since it allows nodes to forward their own data for aggregation through *either* the SPT or the MST as their parent in the tree. The transforms we consider are also better since a node can compress its data using data from more than one neighbor. Thus, we can exploit our stated tradeoff more effectively by searching for a minimum cost tree among a set of trees that combine a distance-based SPT and a correlation-based MST.

An MST does have some drawbacks, though. For one, it may not have as many merge points as an SPT. Since our transform only exploits cross-path correlation at and around merge nodes, having fewer merges may actually reduce the efficiency of our transform when performed along an MST. However, an appropriate combination with an SPT should maintain these merges whenever beneficial. In fact, not all of the neighbors of a node in the MST will have high data correlation with it so some merges may actually hurt coding performance. As mentioned above, our MSTs only consider correlation over a single hop and may result in some inefficiency since our proposed transform actually filters data over multiple hops. Furthermore (as will be discussed in Section 4.2), if a predict has more neighbors it will tend to have less residual energy and so should require fewer bits. Similarly, having more neighbors at an update node can produce a smoother approximation of the original data and as such should also require fewer bits. So as an alternative to MSTs, we could develop trees that (1) preserve beneficial merges and (2) keep the number of merges at predict nodes to a minimum. These issues will be explored experimentally in Section 4.2 and are an active area of research for us.

In Section 3.1, we propose an algorithm that finds the minimum cost combination of an SPT and MST by computing, for every possible combination, the cost of transform and routing along each tree (overlayed on an SPT as discussed in Section 1.1) and then selecting the lowest cost combination for a fixed distortion $\mathcal{D}$. The algorithm is general enough to accomodate an arbitrary definition of edge weights used to construct the MST, i.e., $w(m,n) = 1 - r_{m,n}$, or $w(m,n)$ can be physical inter-node distance, or anything else that allows us to quantify the degree of inter-node data correlation. Existing methods [12] can find a tree that simultaneously maintains a low sum of edge weights (via the MST) and low path-wise distance to the sink (via the SPT). Instead, our proposed method directly balances gains in coding efficiency with low cost routing by choosing the combination for which the cost of transform and routing is minimum. Since the number of combinations grows rapidly with the number of nodes, we also propose a heuristic approximation algorithm that is amenable to larger networks in Section 3.2.

3.1 Optimization Algorithm

Note that the edge weights $w(m,n)$ used to construct the MST can be anything we choose and so the algorithm described here can be applied using an arbitrary measure of inter-node data correlation. For a set of N nodes, let T_S denote the SPT and T_M denote an oriented version of the MST. Let T represent the tree which is our desired combination of T_S and T_M, with the children and parent sets for T defined as in Section 2. An oriented version of the MST (T_M) is necessary to define a transform as described in Section 2. Basically, T_M fixes the sink node $N+1$ as the root and directs all edges in the MST towards the sink. We can represent each tree by defining parent functions ρ_n^M and ρ_n^S for T_M and T_S respectively. Under this construction, data at node n is routed to the sink through ρ_n^M in T_M, ρ_n^S in T_S, and ρ_n in T. Thus, we define the edges in each tree by the ordered pairs (n, ρ_n^M) and (n, ρ_n^S) for T_M and T_S.

We construct a minimum cost tree by searching among all feasible combinations of such edges in T_M with such edges in T_S. We first explain how to find the smallest possible set of feasible combinations (i.e., combinations that result in a connected acyclic graph) in Section 3.1.1. We then provide an algorithm that searches over this feasible set to find a minimum cost solution in Section 3.1.2.

3.1.1 Feasible Set Construction

The total number of combinations could be as many as 2^N, but many such edges in T_S and T_M will be the same so we may eliminate those from consideration. Furthermore, not all combinations of such edges will produce a valid tree (i.e., some may result in cycles or may disconnect certain groups of nodes) so the number of combinations can be reduced even further by eliminating invalid trees. We consider an edge (n,m) to be the same in both trees if $m = \rho_n^M = \rho_n^S$ (i.e., the parent of node n is the same in both trees). Thus, we define $V' = \{n | \rho_n^M \neq \rho_n^S\}$ and N' as the number of nodes in V'. We also enumerate this set as $V' = \{n_1, n_2, \ldots, n_{N'}\}$. For each node $n_i \in V'$, let $E'(n_i) = \{(n_i, \rho_{n_i}^M), (n_i, \rho_{n_i}^S)\}$ be the set of edges from n_i to the parent of n_i in either T_S or T_M. Then the full set of combinations of edges we consider in T_M and T_S is given by:

$$\mathcal{E} = E'(n_1) \times E'(n_2) \times \ldots \times E'(n_{N'}).$$

We reduce the search space further by eliminating combinations of edges in $\mathcal{E}$ that do not produce a valid tree (i.e., graphs that are disconnected or have cycles or both).

We check for tree validity as follows. Let $\tilde{\mathcal{E}}_j \in \mathcal{E}$, where j indexes the j-th combination of edges in $\mathcal{E}$. Naturally, $\tilde{\mathcal{E}}_j = \{(n_1, m_{1,j}), \ldots, (n_1, m_{N',j})\}$ for the j-th combination. Let $\tilde{E}$ be the set of edges in T_S that are the same in T_M. Then a combination $\mathcal{E}_j$ will be feasible only if the graph $\tilde{T} = (V, \tilde{E} \cup \tilde{\mathcal{E}}_j)$ is connected and acyclic. This is done by checking that each leaf node has a non-cyclic path to the sink (which is sufficient since this process traverses every node in the network). Otherwise, the graph $\tilde{T}$ does not form a valid tree. We represent the set of feasible trees by the $N_f \times N$ matrix $\mathbf{T}_f$, where N_f is the number of feasible trees and $\mathbf{T}_f(m,n)$ is the parent of node n in the m-th feasible tree.

3.1.2 Feasible Set Search

Since the full set of feasible trees is given by $\mathbf{T}_f$, we could then find the tree that optimizes routing and transform by i) fixing a target distortion level $\mathcal{D}$ (in our case, distortion is Mean Squared Error (MSE)) , ii) computing the cost C_j for performing routing and transform along every possible tree given in $\mathbf{T}_f$ with distortion level $\mathcal{D}$, and iii) choosing the tree with minimum cost. This is an exhaustive search over our set of feasible combinations of MST and SPT, and should therefore provide the minimum cost combination.

Specifically, this is done as follows. Let C^* be the cost for the best tree found up to row j and initialize it as $C^* = \infty$. Also let i^* index the row in $\mathbf{T}_f$ corresponding to C^* and initialize it as $i^* = 0$. Then for each row j of $\mathbf{T}_f$, with $j = 1, 2, \ldots, N_f$, do the following. Define the parent function $\rho_j(1:N) = \mathbf{T}_f(j, 1:N)$ and compute $C_j =$

ComputeCost$(\rho_j, \mathcal{D}, T_S)$ (a function that computes the cost of doing transform and routing along the tree corresponding to ρ_j with SPT T_S overlayed on top). If $C_j < C^*$, then $C^* = C_j$ and $i^* = j$. Once all feasible trees are exhausted, we can construct the tree T (which minimizes the cost for routing and transform over all feasible combinations of MST and SPT) using the parent function defined by $\mathbf{T}_f(i^*)$. In the context of this paper, the cost to route coefficient n to the sink equals the cost to forward coefficient n two hops along the tree resulting from ρ_j (i.e., cost to route from n to $\rho_j(\rho_j(n))$) plus the cost to route it along the shortest path from $\rho_j(\rho_j(n))$ to the sink. The function ComputeCost$(\rho_j, \mathcal{D}, T_S)$ simply returns the sum of those costs for each node.

3.2 Heuristic Approximation Algorithm

For large N, the feasible set found in Section 3.1.1 can still be very large. This makes the problem intractable for large N, which motivates the need for a good heuristic algorithm that approximates the minimum cost algorithm in Section 3.1.

The main goal of a good heuristic should be to choose links that provide a direct gain in coding efficiency only if the resulting increase in routing cost does not offset the gains achieved, so that a desirable balance of low cost routing and higher compression efficiency can be obtained. This can be done reasonably well by starting from an initial tree and searching one node at a time from nodes of greatest depth (since these nodes will be further from the sink and will benefit more from efficient coding) and decrementing depth at each stage until all nodes are covered. In our case we choose SPT as the initial tree in order to preserve low routing costs, then for each node we simply determine if the cost (as defined in Section 1, C_T) to use the next hop in the MST is lower than the cost to continue along the next hop in the current tree (e.g. SPT). If so, then the next hop of such a node will be the next hop along the MST (rather than the next hop along the SPT). This ensures that, for each node, any direct gains in coding efficiency will not be offset by the resulting increase in routing cost. This is clearly a greedy algorithm, and so can not guarantee that the optimal combination of an MST and SPT will be found. But at the very least, it will guarantee that the resulting tree provides lower cost transform and routing than a transform performed along the SPT.

This algorithm is described formally in Algorithm 1. The final tree we seek is T and initially $T = T_S$. This allows us to greedily choose an edge in T_M over an edge in T_S only if the direct gain in coding efficiency offsets the increase in routing cost. Naturally, the validity of the tree that results from switching to an edge in T_M is checked before further steps are taken. We also say that a parent function ρ_j yields a feasible tree if the tree defined by ρ_j is a connected, acyclic graph. The algorithm simply searches each resulting tree and returns the lowest cost tree it finds as T.

Algorithm 1 Find Heuristic Tree

```
1:  $T = T_S$ and $\rho_j = \rho_j^S, \forall j \in \mathcal{I}$
2:  $k = \max(\text{depth})$ and $C = \infty$
3:  while $k \geq 1$ do
4:    $\mathcal{I}_k = \{m \in \mathcal{I} : \text{depth}(m) = k\}$
5:    for each $n \in \mathcal{I}_k$ do
6:      $\rho_n^t = \rho_n^M$ and $\rho_j^t = \rho_j, \forall j \in \mathcal{I} \backslash \{n\}$
7:      if $\rho_j^t$ yields a feasible tree then
8:        $C_t = \text{ComputeCost}(\rho_t, \mathcal{D}, T_S)$
9:        if $C_t < C$ then
10:         Update $T$ and $\rho_j$ using $\rho_j^t$
11:         $C = C_t$
12:       end if
13:     end if
14:   end for
15:   $k = k - 1$
16: end while
17: return $T$
```

4 EXPERIMENTAL RESULTS

For our experiments, we used simulated data generated from a second order AR model and empirical data from a real wireless sensor network deployment. The simulation data consists of two 600×600 2D processes generated by a second order AR model with low and high (spatial) data correlation (ergo, nodes that are a certain distance away have higher inter-node correlation for the high correlation data than for the low correlation data). The nodes were placed in a 600×600 grid, with node measurements corresponding to the data value from the associated position in the grid. The set of empirical data is from a subset of 19 sensors from a habitat monitoring deployment [13] on the Great Duck Island. The dataset used is for 200 temperature readings taken at each sensor location on August 1, 2003 at roughly 5 minute intervals. We also generate smooth pseudo-stationary data by applying a fixed set of horizontal and vertical lines of discontinuity to the same simulated 2D data as above. The data along one side of a line of discontuity is made significantly different than the corresponding data along the other side to ensure that data correlation is not fully isotropic throughout the network. Such data will be locally stationary (within the boundaries of lines of discontinuity), but will be highly non-stationary near lines of discontinuity.

The specific parameters used in our experiments are as follows. We use the same edge costs for transform optimization found in [5] and our transform optimization only allows for up to 2-levels of decomposition. For our joint routing and transform optimization methods, the SPT used is based on edges defined by squared inter-node distances and the MST is based on edges defined by inter-node data correlation (i.e. edge weights given by $w(n,m) = 1 - r_{n,m}$). The performance metrics considered are the total cost (detailed in Section 3.1.2) and the Mean Squared Error (MSE) between the original and reconstructed data (expressed in terms of Signal to Noise Ratio (SNR)). These are only empirical Cost-Distortion curves where bits are allocated to sensors using standard techniques in [9], where allocations are based on the variances of wavelet coefficients. Basically, we use a set of training data to compute wavelet coefficients, compute the variances of these coefficients, and then apply these variances to the methods in [9].

Section 4.1 compares our proposed transform against the transforms in [3, 20] for two different network configurations, assuming SPTs only. This shows that our proposed transform outperforms existing methods in terms of a cost-distortion trade-off and also that the optimal transform of Section 2.3 performs the best. Section 4.2 compares the compression performance (in a Rate-Distortion (RD) sense) of an MST against the optimal tree (also in RD sense) found by searching over all possible spanning trees. This shows that the MST is near, but not necessarily, optimal for exploiting correlation. This motivates the need for more sophisticated tree construction methods to use as a starting point for our algorithms. Section 4.3 evaluates the performance of our proposed joint routing and transform optimization algorithms, where the cost-distortion curves shown compare 1-level transforms run on an SPT, the minimum cost trees (with SPT overlay), and the trees generated by our heuristic (also with SPT overlay). These results show that gains in cost-distortion performance can be achieved by using appropriate combinations of an MST and SPT. Our results also show that our proposed heuristic trees are comparable to our minimum cost trees for randomly deployed 40-node networks using simulation data and also for the real 19 node network.

4.1 Evaluation of Proposed Transform

The experiments in this section compare the performance of our proposed transform against the optimal path-wise transform in [3] and 1-level of decomposition of the transform in [20]. Our transform is performed along a shortest path routing tree, as is the transform in [3]. The structure of the transform in [20] invokes a particular localized routing structure where many nodes must transmit data in a bi-directional manner to fully compute the transform, a cost we include with the routing cost. Once transform coefficients for [20] are fully computed, they are forwarded along the shortest path routing tree to the sink.

Figure 7 shows the performance comparisons for two sample 100 node deployments for the uniform and clustered network on the left and right respectively. For both networks, our proposed transform outperforms those in [5, 20]. As expected, our optimal transform is also superior to using either our 1-level or 2-level transform throughout the network. In any case, our transform outperforms those in [5, 20] since the unidirectional computation eliminates costly backward transmissions without sacrificing critical sampling and 2D data de-correlation.

The transform in [20] actually performs worse than raw data transmission for higher rates (costs). This is consistent with experimental results in [19] where a similar version of the transform in [20] actually consumes more energy than a raw data dump for networks of 200 or fewer nodes. However, the authors do provide a number of nodes beyond which the transform outperforms raw data dump. As explained in [19], for these "smaller" networks, the amount of energy savings gained in de-correlating data is not sufficiently high to offset the local communication cost incurred for distributed transform computation.

4.2 Evaluation of MST Performance

As discussed in Section 3, the MST is not necessarily the best tree to minimize the distortion (i.e. maximize SNR) for a given rate. This is mainly due to its lack of merges (as compared to an SPT). For the sake of computational feasibility, we consider a small 20 node network as shown in Figures 3 and 4 where nodes are indexed by the number next to them. We perform an exhaustive search of all possible spanning trees for the best RD tree and the performance curves are shown in Figure 5. For each fixed rate we compute the distortion for each tree and choose the tree with the minimum distortion.

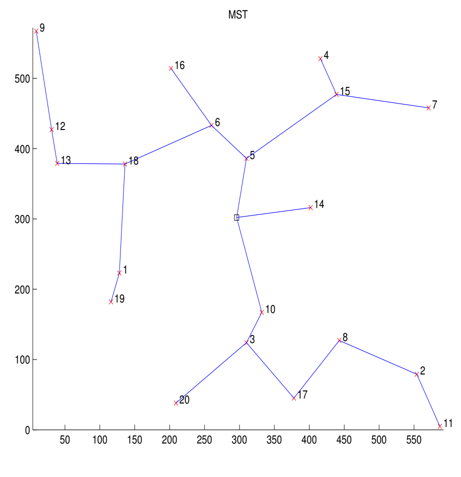


Figure 3. Minimum Spanning Tree

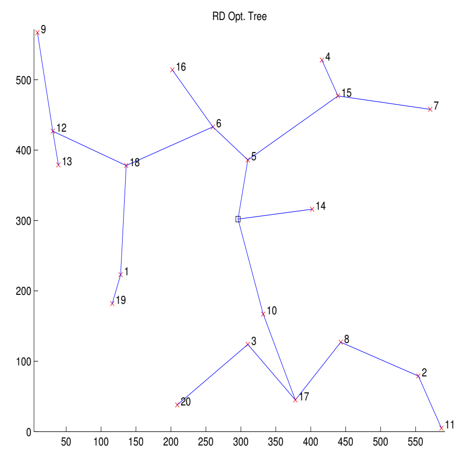


Figure 4. RD Optimal Tree

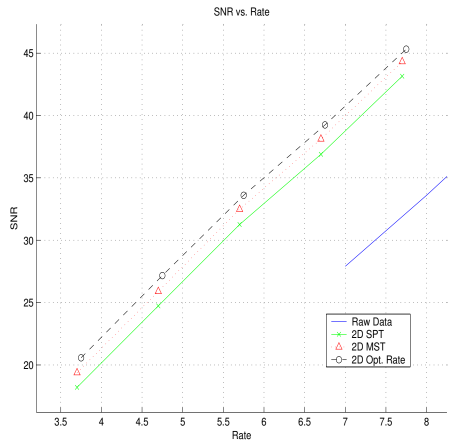


Figure 5. Performance Comparison of MST and RD Optimal Tree

The optimal RD tree shown in Figure 4 has one more merge (node 12) than the MST shown in Figure 3 and also has a different merge with different neighbors (node 17), resulting in reconstruction quality shown in Figure 5. The increase in quality is not so significant in this case, but it may be more significant as the network density grows. The fact that adding a merge improves performance is consistent with our previous discussion.

These results suggest that having more merges at a given node will generally provide better performance. This is reasonable if the data considered is spatially stationary and is highly correlated across space. As discussed before, adding more neighbors to a predict node will tend to produce smaller residual energy. Conversely, an update coefficient (low pass) will provide a smooth approximation to the original data and so adding more residues (i.e. predicts) can increase smoothness which would also reduce the number of bits. All things considered, an MST can provide a good approximation to the optimal RD tree but it is clearly not optimal, mainly because it does not have many merges and because the merges it does have may not occur in the right places in the tree. We are currently investigating the design of trees that can eliminate the shortcomings of MSTs.

4.3 Evaluation of Optimization Algorithms

The experiments in this section compare the performance of our joint optimization algorithm against our heuristic method and a transform performed along an SPT. For the sake of computational tractability, we only run our joint optimization algorithm for networks of 40 nodes and for the real 19 node network. We also only compare the performance of 1-level transforms.

The performance comparisons for our optimization methods and the SPT is shown in Figure 8 for a sample uniform and clustered network. In both cases the heuristic and minimum cost tree both outperform the SPT. This is because both trees utilize shorter edges in the MST for nodes further away from the sink, which is reasonable since it is worthwhile for distant nodes to use edges that provide more de-correlation in exchange for longer paths. Furthermore, the heuristic algorithm finds trees that are very close to the minimum cost trees in both cases. Thus, it provides a reasonable approximation to our minimum cost algorithm for these networks. The heuristic algorithm actually finds the minimum cost tree for the real data network as shown in Figure 9.

To show that this heuristic algorithm still provides performance improvements for different numbers of randomly deployed nodes and varying levels of data correlation, we compare the average costs of uniformly deployed networks for high and low data correlation. Figure 6 plots the ratio of the cost for using a transform (along both an SPT and our heuristic tree) and the cost of transmitting raw measurements directly to the sink (along an SPT) averaged over multiple random networks. The solid lines denote this relative cost for the SPT and the dashed lines denote the relative costs for our heuristic trees. Even on the average, it is clear that the transform along our heuristic tree still outperforms our transform along the SPT.

It is also worth noting that the reduction in relative cost that our heuristic tree provides over an SPT is smaller for the low correlation data (only about 1-2%) than for the high correlation data (about 3-8%). This is reasonable since linking nodes closer together will not improve coding efficiency much if data is not very well correlated. This is also consistent with the findings in [14]. In the case of our pseudo-stationary data (which is only locally stationary), the reduction in cost over an SPT is a bit higher (about 4-10%) for all numbers of nodes. However, there is a point beyond which the reduction for the pseudo-stationary data approaches that for the high correlation stationary data (around 200 nodes). This makes sense since there are many discontinuities in our pseudo-stationary data, so that the number of nodes around these discontinuities will increase as the network becomes denser, thereby mitigating the gains in coding efficiency that can be attained near discontinuities. The reduction in cost (with respect to an SPT) is also highest for the non-stationary data, mainly because there can be edges (n, m) in the MST that do not cross lines of discontinuity but in the SPT the corresponding edge $(n, \tilde{m})$ may do so. It would obviously be inefficient to filter and encode data across lines of discontinuity and using alternate edges in the MST allows us to avoid such situations (whereas if only a single tree is used we can not avoid this), hence the increase in cost reduction. Clearly these reductions in cost are diminishing for larger networks (more than 200 nodes) for all data considered, mainly because nodes will already be spaced close together and so data correlation is already inherently higher across any possible routing tree. The inter-node distances are also smaller on average, which mitigates the effect of local communication cost on total cost.

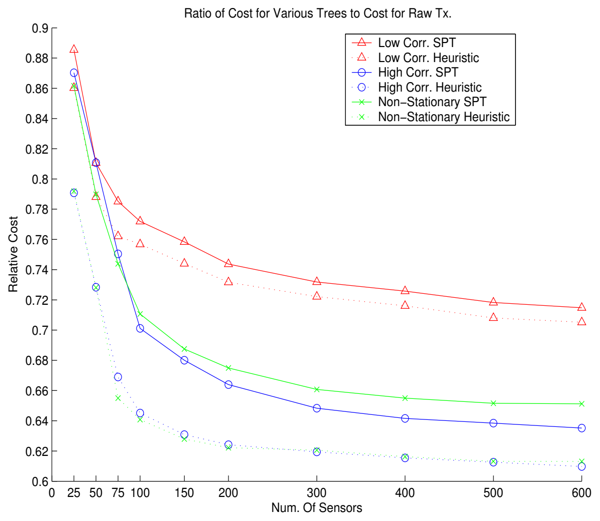


Figure 6. Relative Cost Comparisons

5 CONCLUSIONS

We have presented a framework for finding a routing topology and transform that jointly minimizes energy consumption in an irregular 2D sensor network. A heuristic routing optimization algorithm was also developed that performs comparably to our proposed optimization algorithm. Simulation results have verified that our jointly optimized routing topology and transform always performs best across various network sizes and different types of data. Our results are also consistent with our intuition that a shortest path tree will not always provide the best trade-off between distortion and energy consumption, and is therefore not always best from a joint routing and compression standpoint.

A number of issues can still be addressed. An optimal combination of a shortest path routing tree and a minimum spanning tree may not always provide the lowest cost routing tree. Instead, it may be better to design trees that directly maximize data correlation between adjacent nodes while keeping routing costs to a minimum. Additionally, our proposed lifting filters only exploit correlation effectively for very smooth fields. The de-correlation capability of our transform could be increased by designing filters that adapt to piece-wise planar data. More realistic cost functions could also be incorporated into our system design to reflect the performance of a real system more accurately.

References

[1] J. Acimovic, B. Beferull-Lozano, and R. Cristescu. Adaptive distributed algorithms for power-efficient data gathering in sensor networks. *Intl. Conf. on Wireless Networks, Comm. and Mobile Computing*, 2:946–951, June 2005.

[2] C. Chong and S. P. Kumar. Sensor networks: Evolution, opportunities, and challenges. *Proceedings of the IEEE*, 91(8):1247–1256, August 2003.

[3] A. Ciancio. *Distributed Wavelet Compression Algorithms for Wireless Sensor Networks*. PhD thesis, University of Southern California, 2006.

[4] A. Ciancio and A. Ortega. A dynamic programming approach to distortion-energy optimization for distributed wavelet compression with applications to data gathering inwireless sensor networks. In *Proc. of 2006 IEEE Intl. Conf. on Acoustics, Speech and Signal Processing, ICASSP'06*, 2006.

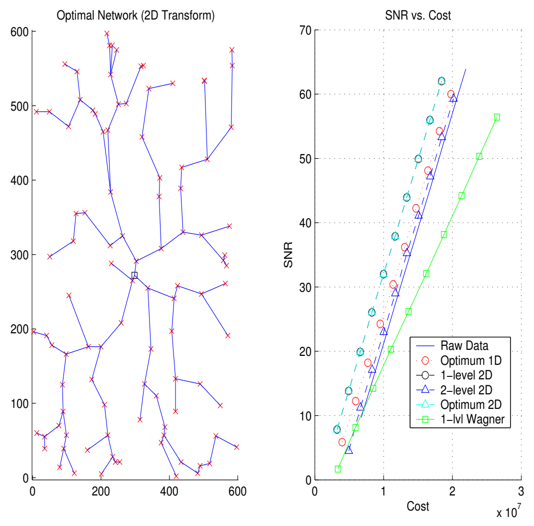
[5] A. Ciancio, S. Pattem, A. Ortega, and B. Krishnamachari. Energy-efficient data representation and routing for wireless sensor networks based on a distributed wavelet compression algorithm. In *IPSN '06: Proceedings of the Fifth International Conference on Information Processing in Sensor Networks*, pages 309–316, New York, NY, USA, 2006. ACM Press.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2nd edition, 2001.

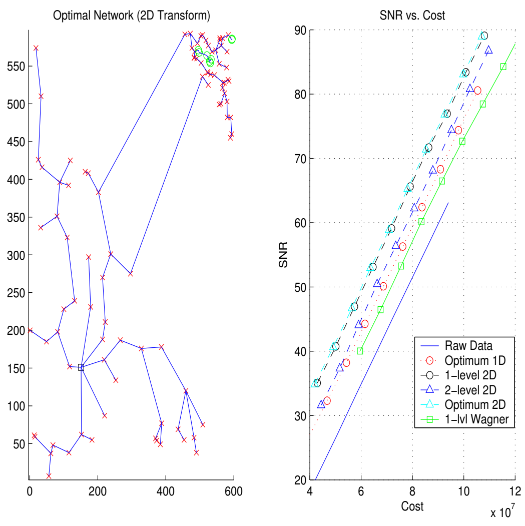
[7] D. Estrin, D. Ganesan, S. Ratnasamy, and H. Wang. Coping with irregular spatio-temporal sampling in sensor networks. *ACM SIGCOMM Comput. Commun. Rev.*, 34(1):125–130, January 2004.

[8] M. Gastpar, P. Dragotti, and M. Vetterli. The distributed Karhunen-Loève transform. In *Proceedings of the 2002 International Workshop on Multimedia Signal Processing*, St. Thomas, US Virgin Islands, December 2002.

[9] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, 1992.



(a) Uniform Network



(b) Clustered Network

Figure 7. Performance Comparisons for Our Transform

[10] A. Goel and D. Estrin. Simultaneous optimization for concave costs: Single sink aggregation or single source buy-at-bulk. In *SODA*, pages 499–505, 2003.

[11] D. Jungnickel. *Graphs, Networks and Algorithms*. Springer-Verlag Press, 2nd edition, 2004.

[12] S. Khuller, B. Raghavachari, and N. Young. Balancing minimum spanning trees and shortest-path trees. *Algorithmica*, 14(4):305–321, October 1995.

[13] H. M. on Great Duck Island. Online data-set located at http://www.greatduckisland.net.

[14] S. Pattem, B. Krishnamachari, and R. Govindan. The impact of spatial correlation on routing with compression in wireless sensor networks. In *Proceedings of the Third International Symposium on Information Processing in Sensor Networks*, April 2004.

[15] P. Rickenbach and R. Wattenhofer. Gathering correlated data in sensor networks. *Proceedings of the 2004 Joint Workshop on Foundations of Mobile Computing*, pages 60–66, October 2004. Philadelphia, PA, USA.

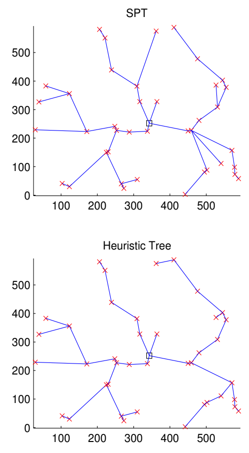
[16] S. D. Servetto. Distributed signal processing algorithms for the sensor broadcast problem. *Conf. on Information Sciences and Systems, The Johns Hopkins University*, March 2003.

[17] G. Shen and A. Ortega. Optimized distributed 2d transforms for irregularly sampled sensor network grids using wavelet lifting. In *IEEE Intl. Conf. on Acoustics, Speech and Signal Processing, ICASSP'08*, Las Vegas, Nevada, USA, April 2008.

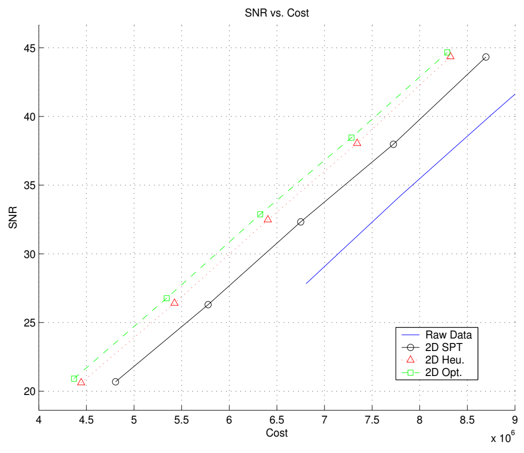
[18] W. Sweldens. The lifting scheme: A construction of second generation wavelets. Technical report 1995:6, Industrial Mathematics Initiative, Department of Mathematics, University of South Carolina, (ftp://ftp.math.sc.edu/pub/imi_95/imi95_6.ps), 1995.

[19] R. Wagner, R. Baraniuk, S. Du, D. Johnson, and A. Cohen. An architecture for distributed wavelet analysis and processing in sensor networks. In *IPSN '06: Proc. of the Fifth International Conference on Information Processing in Sensor Networks*, pages 243–250, New York, NY, USA, 2006. ACM Press.

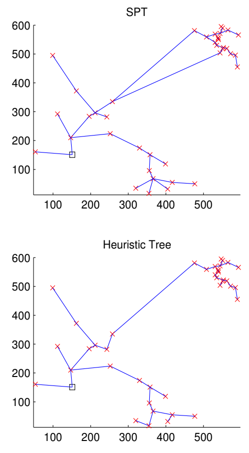
[20] R. Wagner, H. Choi, R. Baraniuk, and V. Delouille. Distributed wavelet transform for irregular sensor network grids. In *IEEE Stat. Sig. Proc. Workshop (SSP)*, July 2005.



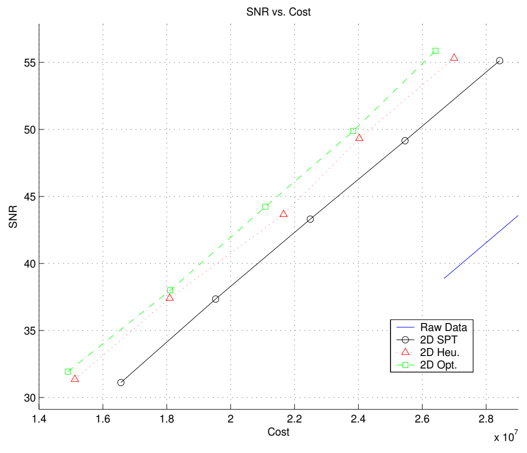
(a) Uniform Network



(b) Performance Comparisons (Uniform)

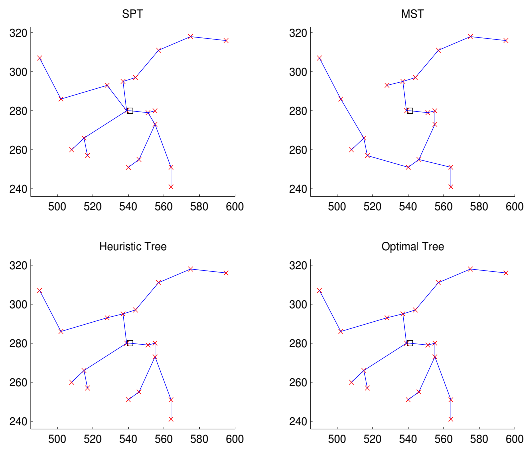


(c) Clustered Network

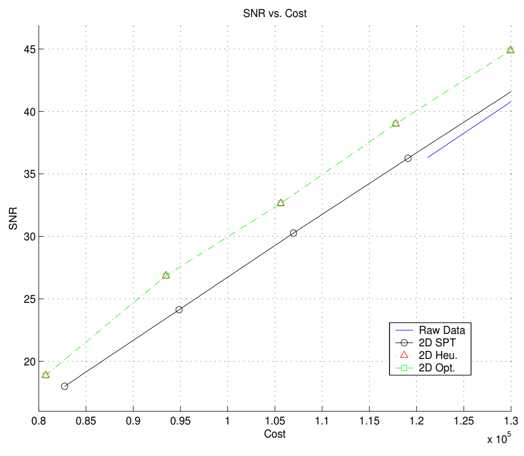


(d) Performanc Comparisons (Clustered)

Figure 8. Performance Comparisons of SPT, Heuristic, and Minimum Cost Tree for Simulated Data



(a) Real Networks



(b) Performance Comparisons

Figure 9. Performance Comparisons of SPT, Heuristic, and Minimum Cost Tree for Real Data