UNIDIRECTIONAL GRAPH-BASED WAVELET TRANSFORMS FOR EFFICIENT DATA GATHERING IN SENSOR NETWORKS

Sunil K Narang, Godwin Shen and Antonio Ortega

Ming Hsieh Department of Electrical Engineering
University of Southern California
kumarsun@usc.edu, godwinsh@usc.edu, ortega@sipi.usc.edu

ABSTRACT

We design lifting-based wavelet transforms for any arbitrary communication graph in a wireless sensor network (WSN). Since transmitting raw data bits along the routing trees in WSN usually requires more bits than transmitting encoded data, we seek to minimize raw data transmissions in the network. We especially focus on unidirectional transforms which are computed as data is forwarded towards the sink on a routing tree. We formalize the problem of minimizing the number of raw data transmitting nodes as a weighted set cover problem and provide greedy approximations. We compare our method with existing distributed wavelet transforms on communication graphs. The results validate that our proposed transforms reduce the total energy consumption in the network with respect to existing designs.

Index Terms— Data Compression, Wavelet Transforms, Wireless Sensor Networks

1. INTRODUCTION

Since nodes in a wireless sensor network (WSN) are severely energy-constrained devices, it is essential to perform in-network compression for energy-efficient data gathering. In the context of compression schemes which use distributed spatial transforms, the main objective is to compress the data into transform coefficients that require as few bits as possible. Recent examples of distributed spatial transforms are the tree-based KLT [1], tree-based DPCM [1], and various flavors of wavelet transforms [2, 3, 4, 5]. All of these schemes use spatial transforms to de-correlate data across neighboring nodes, leading to representations which require a smaller number of bits than that needed for representing raw data. This ultimately reduces communication costs and power consumption. For example, the tree-based KLT assumes that nodes are organized onto a routing a tree, then applies a KLT to the data on the subtree of each node to achieve data de-correlation. Alternatively, the various wavelet transforms are computed using lifting [6]. This is done by first dividing nodes into disjoint sets of even and odd nodes, by predicting data at odd nodes with data from even neighbors (yielding prediction errors, or detail coefficients), then by updating data at even nodes with details from odd nodes (yielding approximations, or smooth coefficients).

Note that for these sorts of techniques, some nodes in the network must transmit raw data to their neighbors before any transform computations can take place. We refer to nodes which must transmit raw data as raw data nodes. Nodes which receive this raw data can then use it to decorrelate their own data (resulting in transform coefficients), encode these transform coefficients and then send them towards a central collection (or sink) node. We refer to nodes which receive and process data as aggregating nodes. In the case of the wavelet transforms, even nodes serve as raw data nodes since they must transmit raw data to their odd neighbors. Odd nodes then use this even node data to compute detail coefficients, hence, odd nodes serve as the aggregating nodes. If the sensed data is spatially correlated and a “sufficient” amount of data is received at aggregating nodes, the decorrelated data (i.e., the transform coefficients) at aggregating nodes will generally require significantly fewer bits than those needed to represent raw data. Therefore, the cost of transmitting encoded coefficients can be many orders of magnitude smaller than the cost of transmitting raw data. This reduces the overall energy consumption in the network.

Because of the higher number of bits they require, it would be good to minimize the number of raw data nodes. Note that reducing the number of raw data nodes will also reduce the number of neighbors whose data each aggregating node can use. This may reduce the degree to which aggregating nodes can de-correlate data, hence, the number of bits needed to represent transform coefficients may increase as the number of raw data nodes decreases. However, in this work we observe that the cost for raw data transmission constitutes a large percentage of the total cost. In particular, the cost reduction from having fewer raw data nodes will often offset the higher number of bits needed for transform coefficients. So in order to design efficient distributed transforms, we propose to design transforms which minimize the number of raw data nodes.

In this work, we choose to focus on lifting based distributed wavelet transforms. These transforms are useful since they are invertible as long as, given an arbitrary even and odd splitting, data from odd nodes is only processed using data from even nodes and vice versa. Moreover, even nodes play the role of raw data nodes and odd nodes play the role of aggregating nodes. Therefore, our proposed design seeks to find invertible, energy-efficient lifting transforms by finding even and odd splittings which minimize the number of even (i.e., raw data) nodes. Such designs will implicitly minimize the number of nodes which must transmit raw data, therefore leading to transforms which are most energy-efficient overall.

To the best of our knowledge, distributed transform designs which minimize the number of raw data nodes have not been previously explored in the literature, nor has there been any work on minimizing the number of even nodes in the splitting used for lifting transforms. In fact, the work in [3, 4] seeks to do the exact opposite, i.e., they seek to find even and odd splittings that maximize the number of even neighbors per odd node. This provides a high degree of data de-correlation for odd nodes, but as discussed in that work, this...
leads to splits with roughly 25% of the nodes as odd and 75% of the nodes as even. Therefore, the amount of raw data forwarding is very high. This is the main reason why their scheme only outperforms raw data gathering for very dense networks [5, 4]. Alternatively, the split design proposed in our previous work [5] is based solely on the depth of each node in the routing tree, i.e., nodes which have even (resp. odd) depth are even (resp. odd). This leads to roughly 50% even and 50% odd nodes. Our proposed approach provides a much more efficient transform since the number of raw data nodes is significantly reduced.

In our work we apply concepts from graph theory, to construct an even/odd split of nodes which 1) minimizes the number of even nodes while ensuring that at least one even node is in the vicinity of each odd node and 2) minimizes the cost of raw data transmissions per even node. We also discuss transmission scheduling for sensor nodes which makes transform computations unidirectional. Using our proposed algorithm we achieve about 10% more reduction in communication costs as compared to existing tree based lifting transforms. This paper is organized as follows. Section 2 describes how to construct a lifting transform given an arbitrary even/odd partition on sensor nodes. Section 3 describes good even-odd partition problem as a set cover and weighted set cover problem on graphs and discusses greedy approximation solutions. Section 4 evaluates the performance of the proposed algorithms. Section 5 concludes the work.

2. UNIDIRECTIONAL LIFTING TRANSFORMS

In this section, we discuss general construction of lifting wavelet transforms on network graphs, and how to make these transforms unidirectional. The following notation is used for the rest of the paper. Let $G = (V, E)$ be a directed communication graph of a WSN with $N$ nodes indexed by $n \in \mathcal{I} = \{1, 2, \ldots, N\}$, with the sink node having index $N + 1$ and where each edge $(m, n) \in E$ denotes a communication link from node $m$ to node $n$. Let $T = (V, E_T)$ be a routing tree in $G$ along which data, denoted by $x(n)$, flows towards the sink. Let depth$(n)$ be the number of hops from $n$ to the sink on $T$ and let $\rho_n$ denote the parent of $n$, $\mathcal{C}_n$, the set of children of $n$ and $\mathcal{D}_n$, the descendents of $n$ in $T$. Also let $\mathcal{A}_n$ denote the set of nodes that $n$ routes data through to the sink excluding the sink, i.e., ancestors of $n$. Finally, let $\mathcal{E}$ and $\mathcal{O}$ denote some arbitrary set of even and odd nodes respectively.

Lifting based wavelet transforms have been proposed in [3, 4] for graphs in Euclidean Space and recently for general graphs in [7]. Shen and Ortega [8] proposed a set of conditions for a distributed lifting transform to be unidirectional, i.e., the transform can be computed along $T$ as data is routed towards the sink. Given a transmission schedule which assigns a time slot to each node to transmit its own data along routing tree $T$, a transform has unidirectional operation if each node can compute its coefficients using only data received from the nodes which transmit before node $n$ (data from descendents $\mathcal{D}_n$ and broadcast neighbors $\mathcal{B}_n$), and $n$ does not forward data from its broadcast neighbors, i.e., $m \in \mathcal{B}_n \setminus \mathcal{D}_n$. In case of lifting this means that the transform is unidirectional as long as each even node transmits its data before all of the odd nodes which are connected to it.

We use a simple strategy in which all even nodes transmit their data first according to their depth in the routing tree, followed by odd nodes according to their depth in the tree. This way odd nodes gather all the data they need to compute their predictions before transmitting their own data. This can increase the network delays as the nodes may have to transmit their own data and data from their ancestors in different time slots. However cost-wise it would make no difference since each node is transmitting its own data only once. Linear prediction operators $p_n$ and update operators $u_m$ at nodes $n \in \mathcal{O}$ and $m \in \mathcal{E}$, respectively can be designed in a variety of ways. However we use the NLMS filters implemented in [1], which adapt the spatial prediction filters over time. NLMS filters converge to optimal filters over time by gradually minimizing prediction errors. Given the even/odd split in the graph, let $\mathcal{N}_e$ denote the set of transform neighbors for all nodes $n$, where $\mathcal{N}_e \subseteq \mathcal{O}$ for $n \in \mathcal{E}$ (i.e., even nodes only have odd neighbors) and $\mathcal{N}_e \subseteq \mathcal{E}$ for $n \in \mathcal{O}$ (i.e., odd nodes only have even neighbors). Then for each $m \in \mathcal{O}$ we compute detail coefficient $d(m)$ as:

$$d(m) = x(m) - \sum_{k \in \mathcal{N}_e} p_m(k) x(k)$$

Note that if the prediction $\sum_{k \in \mathcal{N}_e} p_m(k) x(k)$ is close to $x(m)$, then $d(m)$ will have small magnitude and so can be encoded using fewer bits than that those that would be needed for raw data $x(m)$. This will ultimately lead to cost reduction for odd nodes since they transmit fewer bits per coefficient. An update step can also be computed for data from each even node to produce smooth coefficients, but the number of bits needed to encode smooth coefficients is typically the same as the number of bits needed for raw data. Therefore, in this work even nodes do not compute any update coefficients.

As an example, consider a simpler version of the transform presented in [5, 8] where the splitting is based on depth (i.e., odd depth nodes are odd, even depth nodes are even). In this case, all even nodes must forward raw data one hop to their odd parents, odd parents compute predictions, then odd parents update data from their even children. Since splitting is based on depth, roughly half of the nodes will be even, thus, roughly half of the nodes must transmit raw data. This large number (50%) or raw data nodes is due to the splitting on the tree, which has roughly half even depth nodes. We address this inefficiency in the next section by finding better split designs.

3. EVEN ODD SPLITTING DESIGN

In this section we compute node partitions (even/odd) which minimize cost of transmissions. In case of the uni-directional transform described above this problem boils down to minimizing the number of even nodes in the network. In addition, order to reduce the energy of prediction residues (which leads to lower bit-rate) we want each odd node in the network to have at least one even node in its neighborhood to compute its detail coefficients. Given the radio-ranges of the nodes, this becomes a set covering problem on directed communication graph. The radio-ranges affect the number of neighboring nodes, and thus size of the set-cover. As a first stage we fix the radio-range of each node to the minimum value that guarantees that a node can transmit data to its parent in the routing tree.

Below we formulate the set covering and weighted set covering problems for directed graphs, as they apply to the scenarios we consider here.

**Set Covering Problem**: For Graph $G = (V, E)$ denote closed neighborhood $n_{[v]} = n_{[v]} = \{v\} \cup \{u \in V : vu \in E\}$ for all nodes $v \in V$. Given a collection $\mathcal{N}$ of all sets $\{n_{[v]}\}$, a set-cover $\mathcal{C} \subseteq \mathcal{N}$ is a sub collection of the sets whose union is $V$. The set-covering problem is, given $\mathcal{N}$, to find a minimum-cardinality set cover.

In our case the neighborhood $n_{[v]}$ for node $v$ is set of all nodes within the radio-range of node $v$. Once we obtain a set-cover $\mathcal{C} =$
\{\{v_{ij}\}\}_{j=1,2,...,p} we denote set \{\{v_{ij}\}\}_{j=1,2,...,p} as even nodes and remaining nodes as odd nodes. The problem in this form is also referred to as dominating set problem.

Set-covering problem for unweighted undirected graphs is NP-hard in general. However it can be solved by a natural greedy algorithm that iteratively adds a set that covers the highest number of yet uncovered elements. It provides a good approximation [9] and can be implemented in a distributed way. The algorithm is same for directed graphs with the exception that sets with highest outdegree of central node are added first to the cover. The algorithm for choosing a greedy set cover in directed graph is given in Algorithm 1.

Algorithm 1 Greedy Set Cover for Unweighted Directed Graphs.
\begin{algorithmic}
\Require \mathcal{N} = \{n_v\}_{v \in V} \hspace{1cm} W = \{w_v\}_{v \in V}
\State Initialize \mathcal{C} = \emptyset. Define \( f(\mathcal{C}) = |\bigcup_{n_v \in \mathcal{C}} n_v| \)
\Repeat
\State Choose \( v_j \in V \) maximizing the difference \( |f(\mathcal{C} \cup \{n_{v_j}\})| - f(\mathcal{C}) \)
\State Let \( \mathcal{C} \leftarrow \mathcal{C} \cup \{n_{v_j}\} \)
\Until \( f(\mathcal{C}) = f(\mathcal{N}) \)
\Return \mathcal{C}
\end{algorithmic}

However, these set covering problems do not take into consideration the fact that selected even nodes may be very far from the sink. In that case although the even nodes are small in number, the cost of transmitting raw data from even nodes to the sink is very high. To avoid this we propose minimum weighted set covering problem. In the weighted set-covering problem, for each set \( n_v \in \mathcal{N} \) a weight \( w_v \geq 0 \) is also specified, and the goal is to find a set cover \( \mathcal{C} \) of minimum total weight. In the context of our problem weight \( w_v \) for node \( v \) is the total cost of transmitting raw data from node \( v \) to the sink along the routing path.

The greedy algorithm for weighted set cover builds a cover by repeatedly choosing a a set \( n_v \in \mathcal{N} \) that minimizes the weight \( w_v \) divided by number of elements in \( n_v \) not yet covered by chosen sets. The algorithm for choosing a greedy set cover in weighted vertex directed graph is given in Algorithm 2.

Algorithm 2 Greedy Set Cover for Weighted Vertex Directed Graphs.
\begin{algorithmic}
\Require \mathcal{N} = \{n_v\}_{v \in V} \hspace{1cm} W = \{w_v\}_{v \in V}
\State Initialize \mathcal{C} = \emptyset. Define \( f(\mathcal{C}) = |\bigcup_{n_v \in \mathcal{C}} n_v| \)
\Repeat
\State Choose \( v_j \in V \) minimizing the cost per element \( w_{v_j} / |f(\mathcal{C} \cup \{n_{v_j}\})| - f(\mathcal{C}) \)
\State Let \( \mathcal{C} \leftarrow \mathcal{C} \cup \{n_{v_j}\} \)
\Until \( f(\mathcal{C}) = f(\mathcal{N}) \)
\Return \mathcal{C}
\end{algorithmic}

4. EXPERIMENTAL RESULTS

In this section we compare the unidirectional transforms with graph-based splits presented here against the transform with tree-based split discussed in Section 2 (i.e., a 1-level transform) and an extension of this transform where odd nodes perform additional levels of decomposition on data received from their even children (i.e., a multi-level transform). This multi-level transform is constructed in the same manner as the multi-level transform proposed in [8]. For all transforms, we use the data adaptive prediction filter design in [1]. Figure 2 compares the number of raw data transmissions required by a Haar-like lifting scheme and our proposed scheme, for networks of different sizes. It is clear that our proposed method leads to a significant reduction in raw data transmissions. Assuming a nearly uniform deployment of sensors, the distances between nodes are roughly equal. Hence reduction in the number of raw transmissions is directly proportional to the reduction in transmissions costs as shown in Fig. 1. Further in Fig. 1 the cost of raw data transmissions for weighted set cover based split is lower than the cost of raw data transmissions for unweighted set cover based split. This is to be expected since even nodes selected by weighted set cover algorithms now have lower costs of transmitting data to the sink.

![Fig. 1. Cost Comparison of Different Lifting Schemes](image1)

![Fig. 2. Number of raw data transmissions in transform computations](image2)
that although both Figs. 3(a) and 3(b) have same underlying routing structure (solid blue lines), number of required even nodes are lesser for graph-based transform than tree-based transform. This leads to reduction in raw-data transmission costs.

Performance comparisons are shown in Fig. 4, which plots energy consumption versus reconstruction quality (in terms of Signal to Quantization Noise Ratio). Energy consumption is computed using the cost model in [10]. Each point corresponds to a different quantization level with adaptive arithmetic coding applied to blocks of 50 coefficients at each node. The transform proposed here is the best overall. This is to be expected since it seeks to minimize the number of nodes that must transmit raw data to their neighbors, therein reducing the total energy consumed in the data gathering process. The 1-level and multi-level transforms with tree-based split do outperform simple raw data gathering, and the multi-level transform does better than the 1-level transform since more de-correlation is achieved in the network. However, both of these methods have roughly 50% raw data nodes, hence, they are not as efficient as the two transforms with graph-based splits (which have roughly 25% raw data nodes).

![Transform Structure on SPT](image1)

![Transform Structure on Graph](image2)

Fig. 3. Transform definition on SPT and on graph. Circles denote even nodes and x’s denote odd nodes. The sink is shown in the center as a square. Solid lines represent forwarding links. Dashed lines denote broadcast links.

![SNR vs. Energy Consumption](image3)

![Total Energy Consumption (Joules)](image4)

Fig. 4. Performance comparisons.

To see this distinction more clearly, consider the lossless coding numbers for this same network shown in Fig. 1. The cost for raw data forwarding and the total cost are shown separately. As we can see, the overall performance is greatly affected by the raw data forwarding cost in that lower raw data forwarding leads to lower total cost. In particular, the methods proposed here have the lowest raw data forwarding cost, hence, they also have the lowest overall cost.

Our proposed transform can be easily applied to any arbitrary WSN, since it is computed as the data is routed towards the sink. The schedule of computation and the even-odd assignment of nodes can be pre-fed into sensors at initialization. This transform design can be seen as precursor to a new class of algorithms which would focus on minimizing raw data transmissions in a WSN by jointly optimizing routing tree and even/odd partition (or raw nodes/aggregating nodes partition).

5. CONCLUSIONS

Given an arbitrary communication graph and a routing tree, we have defined a wavelet lifting transform which uses very low number of raw data transmissions. Experimental results show performance improvements with respect to a lifting transform computed only along a routing tree. As future work, we can consider other problems including selecting the tree, transmission schedule and transform jointly for a given graph.

6. REFERENCES