Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Contributions to Distributed Spatial Compression in Wireless Sensor Networks

Javier Pérez Trufero

July 2010

Supervised by

Prof. Antonio Ortega

Signal and Image Processing Institute

Viterbi School of Engineering

University of Southern California

Los Angeles, California, USA

A mi madre,
por todo lo superado este año.

Abstract

This thesis presents several contributions in the field of distributed spatial compression in Wireless Sensor Networks. First, since in most of the spatial compression schemes some nodes (raw nodes) need to broadcast their raw data to allow other nodes (aggregating nodes) to perform compression, we design several distributed heuristics which, via local communications, split the nodes into raw/aggregating subsets and optimize the amount of energy consumed in the network. We also extend previous work in the use of graph-based lifting transforms for data compression in distributed data gathering applications, to networks with more than one sink, and scenarios where all data has to be available at every node. Additionally, under the scope of these contributions, we design a new energy-efficient multicast routing algorithm, which is based on the minimum Steiner tree and exploits the broadcast property of wireless communications. We prove via computer-based simulations that our methods reduce the energy consumption in the network in comparison with existing approaches.

Acknowledgements

I will allow myself to change the language during this section to make the acknowledgements arrive to the appropriate person in the adequate language.

Este trabajo está dedicado principalmente a mis padres, ya que su paciencia, esfuerzo y confianza durante todos estos años han sido factores cruciales para poder llegar hasta Los Ángeles a realizar este proyecto final de carrera.

Grácias también a todos los demás miembros de mi familia. En especial a Mireia por ser la mejor hermana del mundo y aguantarme durante tanto tiempo. Con el máximo cariño posible quiero agradecer también a mis abuelos su amor y apoyo, ya que siempre me han servido como fuente de motivación.

Grácias a todos mis amigos, por ser quienes sois y hacerme ser quien soy. Especialmente quiero agradecer a todos los que habéis estado conmigo desde que tengo uso de razón, vuestra amistad es lo más valioso que tengo. Grácias también a todos aquellos que os habéis ido sumando a mi vida durante el camino, ya sea en Menorca, Mallorca, Barcelona o Los Ángeles. Muchas grácias por todos nuestros momentos juntos, directa o indirectamente me habéis ido ayudado a seguir adelante con la carrera y con este trabajo. Quiero transmitir también un agradecimiento especial a todos aquellos que habéis venido hasta California a visitarme durante estos nueve meses, por querer compartir conmigo esta experiencia y facilitar mi adaptación a esta ciudad.

I would like to thank all the people working at the research group for all their help and support during these months. Specially, I want to thank Godwin Shen and Sunil Narang for their patience with my infinite questions, and for offering always valuable advice during our collaborations. Grácias también a Alfonso Sánchez por toda la ayuda prestada al principio de mi estancia en Los Ángeles y a Eduardo Martínez por sus continuos ánimos, consejos y momentos de diversión durante estos últimos meses.

Finally, I can not express in words how thankful I am to my supervisor in this project, Prof. Antonio Ortega. From the beginning of my stay at USC, he has been a source of advice, inspiration, and encouragement. Thank you for giving me the possibility to collaborate with your research group and for being not only a supervisor, but also a motivation force both academically and personally. I will never forget this experience.

This thesis has been supported in part by Fundación Vodafone España - ETSETB under grant Vodafone 09-10, Fundación Bancaja under grant Becas Internacionales Bancaja - UPC 2009-10, and AGAUR under grant MOBINT 09-10.

Grácias a todos / Gràcies a tots / Thank you all.

Contents

List of Algorithms

List of Figures

Chapter 1

Introduction to Spatial Compression for Distributed Data Gathering

1.1 Introduction

Sensor networking has been an active research area in the last decade, and its development promises an unprecedented ability to sense, monitor, and manipulate our environment. Wireless Sensor Networks (WSN) consist of many inexpensive and densely deployed sensors, which have the ability to sense phenomena from the physical world, and communicate with each other in order to perform some sort of coordinated task. The list of applications for WSN seems to be endless, with environmental and habitat monitoring, object detection and tracking, military applications such as battlefield surveillance, and home automation, being only some of the more obvious examples.

Since nodes in WSN are severely energy-constrained devices, usually powered by batteries or solar cells, the development of energy-efficient algorithms for routing and communications is one of the major challenges for the research community. Moreover, WSN are usually deployed in dynamic and hostile environments, where

topology changes are common, and damaged nodes are hard to replace. In order to address these limitations, another important requisite is to provide the network with the ability to perform in a distributed and self-organized manner. Ideally, once these requirements are fulfilled, WSN will be robust to link/node failures and their lifetime will be prolonged.

The aforementioned limitations become a matter of major importance in coordinated applications, where nodes must exchange data to achieve their goal. For example, in distributed estimation applications [30], nodes have to exchange data in order to estimate a global parameter. Exchange of information is also needed in aggregation problems [15], where each node has to compute an aggregation function (such as sums, averages, etc.) with the values of all the other nodes. Moreover, in applications such as distributed detection [19] and distributed data gathering, sensors share data to complete their tasks. In this work we focus on data gathering, where sensors transmit data to one or several central collection nodes.

Imagine that a number of sensors, equipped with omnidirectional antennas, are spread over an area, sensing data from their environment. The measurements of each node have to be sent to a randomly located base station (sink) via wireless communications. As a consequence of the limited radio range in every node, most of them may not be able to reach the sink directly in a single transmission. In that case, nodes need to use other sensors as relay stations to forward their data until it reaches the target node. The simplest form of data gathering would be to have all nodes transmit raw data to the sink. However, this solution does not exploit the existing correlation between spatially close sensors, which means that a lot of redundant data is being sent over the network. As an alternative, it would be reasonable to decorrelate the data using some sort of transform, and consequently represent the measurements with fewer bits. This reduction will decrease the communication cost in some nodes, and will lead to savings in the amount of energy consumed in the network.

For achieving this goal, there are a variety of approaches available in the literature, e.g., distributed source coding (DSC) techniques [6, 28], wavelet-based methods [1, 5, 42, 41, 36, 35], the distributed Karhunen-Love Transform (KLT) [9], the tree-based KLT [34], the tree-based DPCM [34] and the graph-based wavelet transforms as the one presented by Narang *et al* in [25]. These graph-based wavelet transforms are based on the lifting scheme [39], and are computed as data is forwarded towards the sink on an energy efficient routing tree.

The aim of this thesis is to present several contributions in the field of distributed spatial compression for distributed data gathering applications in WSN. We extend previous work in the use of graph-based lifting transforms for compression purposes to networks with more than one sink (multisink WSN), and gossip scenarios in which all data has to be available at each node. We also present a new multicast routing algorithm based on the minimum Steiner tree problem, which exploits the broadcast property of wireless communications. Moreover, since in most of the distributed spatial compression schemes the first requirement in order to implement compression is to perform a raw/aggregating node assignment (RANA), we introduce several distributed heuristics that split the sensors into subsets of raw and aggregating nodes by allowing nodes only to communicate with their direct neighbors. By doing so, our decentralized approaches are useful for the implementation in real settings.

We next discuss existing approaches for spatial compression (Section 1.2) and graph-based wavelet transforms (Section 1.3).

1.2 Brief overview of existing approaches

As mentioned before, there are several techniques for performing spatial compression in data gathering applications. For instance, one popular class of methods uses distributed source coding (DSC). In these approaches, nodes do not need to directly exchange data with each other for compression purposes, since encoding is done independently at each node by using techniques such as Slepian-Wolf coding [6]. In these schemes, each node can compress its own data based on some statistical correlation model, which is known a priori. However, nodes will need to exchange data for learning this model before compression can be done, and how to properly estimate the underlying correlation model, and what the training cost will be, are not always obvious. On the contrary, transform-based approaches require some sort of data exchange between nodes to remove spatial correlation. Specifically, some nodes first need to transmit raw data to their neighbors, and then these neighbors process all the available data in order to decorrelate the information and perform compression. The distributed KLT [9] is a cluster-based method, where nodes are first organized into disjoint clusters. Then, nodes within each cluster send raw data to the clusterhead, which processes all the information, performs some sort of compression, and forwards all data to the sink. Note that, although in these cluster-based methods it is not necessary to train nodes for learning the underlying correlation model, the large number of non-cluster-heads that transmit uncompressed data makes this strategy inefficient for energy-aware spatial compression. Furthermore, since it is based on the KLT transform, this method also requires a learning phase, where nodes need to discover and disseminate the correlation structure.

We are going to focus on routing-driven compression schemes. In these methods, data is compressed while it is being sent towards the sink. Specifically, we are going to work with graph-based wavelet transforms such as the one in [25], where some nodes first transmit raw data to their direct neighbors, and then their neighbors compress their information using the data they previously received. Since in graph-

based transforms the subset of neighbors a node uses in order to compress its data is bigger than in the other routing-driven methods, the level of decorrelation in the compressed data is higher, and therefore better compression is achieved. On the other hand, methods in [42, 41] follow the same intuition, but nodes only exchange data with their nearest neighbors. Other routing-driven alternatives are the tree-based KLT and the tree-based DPCM, both proposed in [34], and the wavelet-based methods in [36, 35, 5], which compute the transform along a routing tree. However, in these approaches, the broadcast property of wireless communication is not exploited since nodes compress their data only with information from their children in the routing tree. Hence, nodes are not using the information of all their neighboring nodes for performing compression. In addition, note that the tree-based KLT will also incur a learning cost to estimate and disseminate the correlation model.

1.3 Unidirectional Graph-based Wavelet transforms

Wavelet transforms have been widely used as powerful tools for signal compression. Specifically, distributed wavelet-based schemes, such as lifting [39], allow these transforms to be computed in arbitrary network configurations while ensuring invertibilty by construction. For implementing the lifting scheme, the first requirement is to have nodes split into raw data nodes and aggregating nodes subsets. Once this has been done, raw data nodes first broadcast raw data to their aggregating 1-hop neighbors, and then these aggregating nodes compress their data, yielding *detail* coefficients (or prediction errors), with the raw data they have received. As a last step, data from raw data nodes can be *updated* with detail coefficients from aggregating nodes, yielding *smooth* coefficients (or approximations). From now on, the splitting process is going to be referred to as the Raw/Aggregating Node Assignment (RANA).

How the RANA is implemented is a matter of major importance for optimizing the transform in terms of the energy efficiency. As presented in [25], since raw data requires more bits than encoded data, it is preferable to minimize the number of raw data nodes, while ensuring that all aggregating nodes are still covered by at least one raw node. In this case, the RANA is formulated as a minimum set covering problem. In Chapter 2 we will explain and compare different RANA strategies that have been proposed in the literature. However, in this section, and for illustrating the transform construction, we assume that the RANA is based on depth in the routing tree (distance in hops to the sink) as in [36, 35]. In other words, raw data nodes are nodes with even depth, and aggregating nodes are nodes with odd depth.

Once we have explained how each node processes the data, also referred as the *processing strategy*; it is important to note that if we want to reduce the energy consumed in the network (in terms of communication cost), we also need to define an energy efficient *routing strategy*. The routing strategy defines what data communications nodes need to make for gathering all data at the sink. In the single sink case, a simple and efficient routing strategy for reducing the communication cost is to follow the well-known shortest path tree (SPT). However, as will be explained in Chapter 3, in a network with more than one sink the SPT is no longer a good solution. Therefore, we will need to find some other routing strategy in order to gather the data in all the sinks.

1.3.1 General Formulation and Implementation

Assume a network with N nodes indexed by $n \in I = \{1, 2, ..., N\}$ and a sink having index $N+1$. Suppose that each node transmits data using R_n as radio-range and let $G = (V, E)$ be the directed communication graph which results from these choices of radio ranges. Note that each edge $(m, n) \in E$ denotes a communication link from node m to node n. Let N_n be all the adjacent nodes that can hear data from node n in G. Let $T = (V, E_T)$ be a given routing tree along which data, denoted by $x(n)$, flows towards the sink. Note that in our case E_T are the communication links that

form the graph which define the SPT or any other tree provided by some standard routing protocol (i.e., $E_T \in E$). The remaining edges define the communications overheard due to the broadcast property of wireless communications. Focusing on a single node, let depth(n) to be the number of hops from n to reach the sink on T. Assume that ρ_n denotes the parent of n in the tree, its set of children is referred as C_n, and let D_n to be the descendants of n in T. We denote the set of raw data nodes as $\mathcal{D}$ and the aggregating nodes as $\mathcal{A}$. Let the set of raw data nodes that an aggregating node overhears be referred as $\mathcal{H}_n$, i.e., $\mathcal{H}_n = \{m \in \mathcal{D} | n \in N_m\}$. Remember that in this section we assume that the splitting is based on depth in the tree (i.e., odd depth are aggregating nodes, even depth are raw data nodes). Thus, in this case, $\mathcal{H}_n$ includes C_n and all the even nodes that n overhears due to broadcast transmissions.

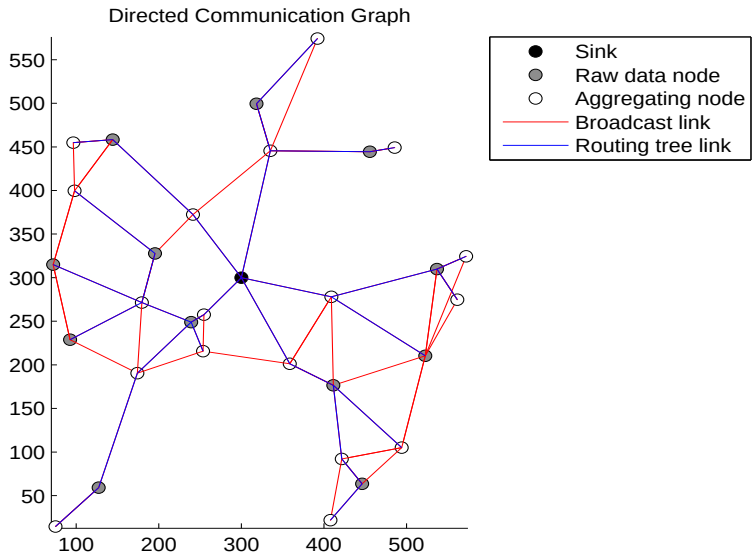


Figure 1.1: Example of a communication graph representing a WSN with routing tree and broadcast links included. The RANA is based on the depth of the routing tree.

We define a *transmission schedule* in which all raw data nodes transmit their data first, and then aggregating nodes compute the detail coefficients using data from all nodes in $\mathcal{H}_n$. After this first step, nodes start forwarding the data from their

descendants D_n until all coefficients are gathered at the sink. We also assume that transmissions between adjacent sensors are scheduled to avoid interference using some algorithm such as those presented in [37, 38].

After all these considerations, each $n \in \mathcal{A}$ computes the detail coefficient $d(n)$ as:

$$d(n) = x(n) + \sum_{k \in \mathcal{H}_n} \mathbf{p}_n(k) x(k) \tag{1.1}$$

As mentioned before, an update step can also be computed to produce smooth coefficients for data in raw nodes. However, the number of bits needed to represent these smooth coefficients is similar to the number of bits for representing raw data. Thus, for simplicity, we are not going to use an update phase in this work. Anyhow, the smooth coefficient $s(m)$ would be computed for each raw node $m \in \mathcal{D}$ using details from aggregating nodes in $\mathcal{H}_m$, i.e., $\mathcal{H}_m = \{n \in \mathcal{A} | n \in N_m\}$, as:

$$s(m) = x(m) + \sum_{l \in \mathcal{H}_m} \mathbf{u}_n(l) d(l) \tag{1.2}$$

Note that, in our case, data from raw nodes would only be updated using detail coefficients from their aggregating parents in the tree, i.e., $\mathcal{H}_m = \rho_m$.

The design of the linear prediction operator $\mathbf{p}_n$ can be done in a variety of ways. For example, $\mathbf{p}_n$ can be a simple average filter [36]:

$$\mathbf{p}_n(k) = -\frac{1}{|\mathcal{H}_n|} \tag{1.3}$$

In this thesis we are going to use the NLMS filter designed in [34], which is adapted in a distributed way without forwarding any additional information towards the sink. Source code used to generate this filter can be found online[1].

Note that we can guarantee the invertibility of the transform with any design of $\mathbf{p}_n$. By the lifting construction, we can ensure invertibility as long as aggregating nodes only predict the detail coefficient with data from raw data nodes. If we have an update step, for guaranteeing invertibility, smooth coefficients must be calculated only with detail coefficients from aggregating nodes.

[1] http://biron.usc.edu/wiki/index.php/Wavelets_on_Trees

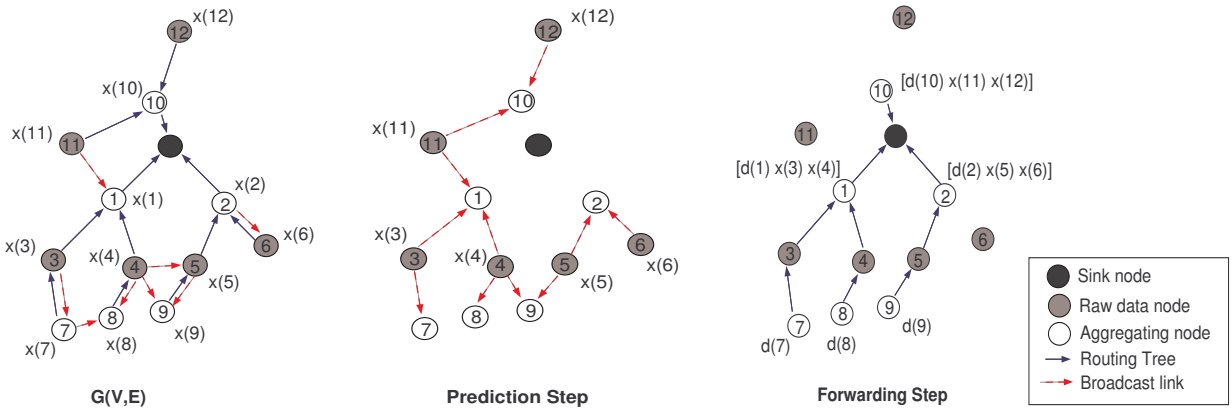


Figure 1.2: Example of the graph-based unidirectional lifting transform.

The compression and data gathering processes are illustrated in Figure 1.2. After the construction of the SPT, each node is aware of its parent and fixes its radio range. This set of selected radio ranges leads to the communication graph $G(V, E)$. Once the network is created and nodes start sensing data (i.e., $x(n)$), raw data nodes 3, 4, 5, 6, 11 and 12 first broadcast their data to aggregating neighbors. After that, aggregating nodes 1, 2, 7, 8, 9 and 10 compute their detail coefficients $d(n)$ with raw data overheard from nodes in each $\mathcal{H}_n$. In the forwarding step, all data is sent towards the sink following the pre-established SPT.

Chapter 2

Raw/Aggregating Node Assignments

For implementing the unidirectional graph-based wavelet transform presented in Section 1.3, nodes need to be split into two different subsets. Specifically, some nodes (i.e., raw data nodes) need to broadcast raw data in order to allow some other nodes (i.e., aggregating nodes) to remove the spatial correlation and compress their measurements. We refer to this process as the Raw/Aggregating Node Assignment (RANA). How the RANA is implemented is critical in terms of energy efficiency, because it affects (i) the number of nodes sending uncompressed data in the network and (ii) the number of bits required to represent the data produced by aggregating nodes.

In this chapter, we summarize the existing strategies for implementing and optimizing the RANA (Section 2.1); In Section 2.2, we propose a low complexity heuristic to perform the RANA as a distributed minimum set covering problem, and in addition we present two distributed optimizations that improve the RANA by allowing the nodes to choose between different radio-ranges. Finally, we simulate the different approaches, analyze their performance and compare our methods to those already proposed in the literature (Section 2.3).

2.1 Existing RANAs

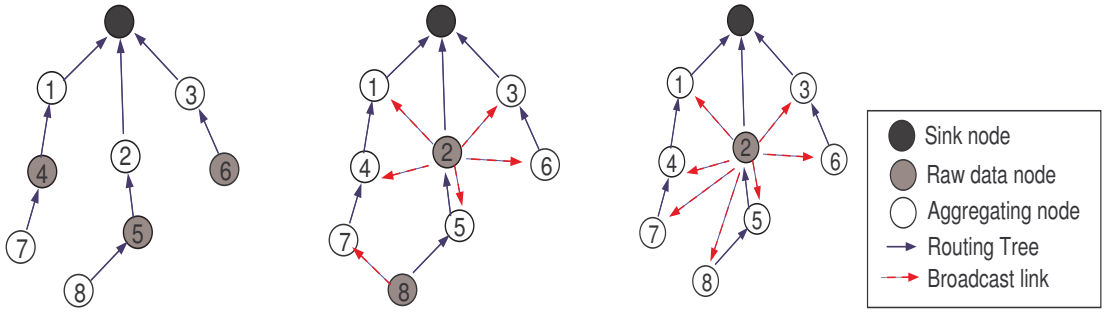
As we have seen, lifting-based techniques for spatial compression need a RANA for being implemented. Different strategies have been proposed in order to assign each node a role (i.e., raw/aggregating). The transform-based compression schemes presented by Shen *et al* in [36, 34] split nodes based on their depth in the routing tree. In their approaches, sensors with even depth (even number of hops to the sink) act as raw data nodes, and sensors with odd depth act as aggregating nodes. This approach leads to aproximately half the nodes being raw data nodes and the other half being aggregating nodes. Considering that data from raw nodes is represented with more bits than data from the aggregating ones, this solution incurs high transmission costs, and it is not optimal in terms of energy efficiency. As an alternative, the work in [42, 41] seeks to maximize the number of raw data nodes each aggregating node has within a certain distance. The intuition behind these methods is that increasing the number of raw data nodes that an aggregating node has in its neighborhood, will provide a higher degree of decorrelation. This leads to RANAs in which around 75% of the nodes forward raw data. Therefore, despite the fact that detail coefficients are represented with fewer bits, the amount of raw data forwarded is high. That explains why their approach only outperforms raw data gathering (without any compression) in very dense networks. The RANA applied in the graph-based lifting transform presented in [24] seeks to minimize the number of conflicts in the graph (the number of direct neighbors that have the same parity).

To the best of our knowledge the RANA proposed in [25] is the only one that seeks to reduce the number of raw data nodes in the network while ensuring that each aggregating node has at least one raw data node in its vicinity. This RANA is formulated as a minimum set covering (MSC) problem and is solved using a centralized greedy heuristic. Even though the resulting set cover minimizes the number of raw data transmissions, it does not take into consideration the fact that selected raw data nodes might be very far from the sink, so their raw data will need

to travel a long distance until it reaches the sink. To avoid this problem, the RANA is formulated as a minimum weighted set covering (MWSC) problem, where the cost of transmitting raw data per even node is also minimized. By solving the problem in terms of set cover minimization, an additional 10% reduction in communication cost is achieved as compared to the aforementioned methods applied to the same lifting transforms.

In recent work [26], partly developed under the scope of this thesis, the RANA is optimized by allowing nodes to change their radio range. This is formulated as a linear program for MWSC and it is solved with standard tools. This optimization reduces the number of raw data nodes in comparison with standard minimum set covering methods. Thus, it leads to total cost reductions up to a 25% as compared with the technique in [25].

Figure 2.1 shows the resulting RANA for different strategies in the same communication graph. In Fig. 2.1(a) the assignment is based on the depth in the tree. Aggregating nodes (resp. raw) are the ones which have odd (resp.even) depth in the routing tree [36, 34]. Fig. 2.1(b) illustrates the RANA using the method in [25], where the assignment is solved as a minimum set covering problem. Finally, Fig. 2.1(c) shows the intuition of the approach presented in [26]. As we can see, allowing node 2 to increase its radio range, it can also cover nodes 7 and 8. Thus, the number of raw data nodes is reduced. Note that, as will be explained in Section 2.2.3, there is a trade-off in increasing the radio range of some nodes, because (i) the communication cost in these nodes will be greater, and (ii) the mean distance between raw data nodes and aggregating nodes will increase. This increase in distance leads to a lower coding efficiency in aggregating nodes' data. Consequently, their measurements have to be represented with a higher number of bits. In other words, while the number of raw nodes decreases, the average number of bits per aggregating node increases.



(a) RANA based on depth in the tree

(b) RANA solved as a Min. Set Covering problem in a fixed graph

(c) RANA after allowing nodes in the set cover to change their radio range

Figure 2.1: Raw/Aggregating node assignments using different existing methods.

2.2 Distributed Set Cover Algorithms

In this section, we first formulate the RANA as a minimum set covering problem as proposed by Narang *et al* in [25]; then, we propose a new distributed heuristic for minimum set cover, where nodes only use information from their direct neighborhood to choose between becoming a raw/aggregating node. Finally, we design two distributed optimizations which, allowing nodes to choose among different radio ranges, reduce the set-cover and the communication costs in the network. All these distributed algorithms are also presented in [26].

2.2.1 Minimum Set Covering Problem

Assume $G(V, E)$ to be a directed communication graph, where for all nodes $v \in V$ there is a closed neighborhood $n_{[v]} = \{v\} \cup \{u \in V : (v, u) \in E\}$. The set covering problem consists in finding the minimum number of nodes such that the union of all their $n_{[v]}$ is V.

In terms of WSN, the neighborhood $n_{[v]}$ represents all nodes within the radio range of node v. Once we have the set $\left\{n_{[v_j]}\right\}_{j\in 1,2,\ldots p}$ conforming the set cover $\mathcal{C}$, we denote $\{[v_j]\}_{j\in 1,2,\ldots p}$ as raw data nodes and the remaining ones as aggregating nodes. Thus, this minimum set cover problem seeks to obtain the minimum number

of raw data nodes while ensuring that every aggregating node is within the radio range of at least one raw data node. This problem is also refered in the literature as the *minimum dominating set problem*.

This problem is NP-hard in general, but it can be solved by a centralized greedy algorithm that iteratively assigns as raw data node that with the highest number of uncovered nodes in its neighborhood. Algorithm 1 illustrates the procedure implemented in [25] in order to solve the set covering problem.

Algorithm 1 Greedy Set Cover for Unweighted Directed Graphs

Require: $\mathcal{N} = \left\{n_{[v]}\right\}_{v \in V}$

1: Initialize $\mathcal{C} = \phi$. Define $f(\mathcal{C}) = \left|\bigcup_{n_{[v]} \in \mathcal{C}} n_{[v]}\right|$

2: **repeat**

3: Choose $v_j \in V$ maximizing the difference $\left[f\left(\mathcal{C} \cup \left\{n_{[v_j]}\right)\right\} - f(\mathcal{C})\right]$

4: Let $\mathcal{C} \leftarrow \mathcal{C} \cup \left\{n_{[v_j]}\right\}$

5: **until** $f(\mathcal{C}) = f(\mathcal{N})$

6: **return** $\mathcal{C}$

2.2.2 Low complexity Distributed Heuristic for Minimum Set Cover

As we have seen in the previous section, the minimum set covering problem can be solved by using a greedy centralized heuristic. However, even though they are accurate, centralized solutions are impractical for real settings for several reasons. First of all, they incur high communication costs in order to gather all the necessary information in a central node or base station. Secondly, they can't adapt to network changes or failures. This means that if there is a change in the network structure, we will need to solve the centralized algorithm again to update the RANA. As an alternative, a more practical solution is to implement the RANA in a distributed way. Using distributed algorithms, nodes can decide their parities via limited communications in their direct neighborhood. Additionally, in case of a change or node failure in the network, nodes are able to readjust and adapt the RANA using only local communications. There are several approaches to solve the minimum set cov-

ering problem in a distributed manner. One solution would be to use a distributed implementation (such as in [17]) for the linear programming formulation presented in [26]. However, the large number of additional communications required to implement this method make it impractical. Other alternatives are based on simulated annealing [13]. Here, the algorithm starts with an arbitrary solution, and then it repeatedly tries to make improvements based on local information until a consensus is achieved. Although the solution can be close to a global optimal, it requires a large number of communications and a long time to converge.

In this section, we present a new distributed algorithm that follows a similar intuition to the one in [12], but in a simpler way. Both algorithms seek to reproduce the centralized greedy algorithm (Algorithm 1) in a distributed manner. In [12], the *span* or outdegree of each node is calculated and forwarded to all neighbors within distance 2 at each round of the algorithm. Moreover, all the uncovered nodes send, also at each round, information related to the number of nodes that are candidates to cover them. After that, the decisions about which of the candidates are added to the dominating set are made based on some probabilistic assumptions. In contrast to [12], in our algorithm nodes only use information from their 1-hop neighborhood, and the necessary communications in order to decide which of the nodes are considered in the set cover are reduced, with nodes exchanging information related to their *span* only once at the beginning of the algorithm. Because of the restrictive conditions imposed by the properties of wireless sensor networks, sometimes it is better to keep the complexity and communication cost small even at the cost of a suboptimal solution. Therefore, our aim in this section is to obtain a small set cover with minimum number of local communications.

A simple way to approach the RANA is to assign as raw data nodes those with greater number of nodes in their direct neighborhood. This procedure can be done in a distributed manner by allowing each node to exchange a few messages with its neighbors, and then letting each node decide whether to become a raw or aggregating node.

Our proposed algorithm consists of three main steps: (i) determine number of neighbors at each node, (ii) exchange outdegree information, and (iii) sequential raw-aggregating assignment.

Algorithm 2 describes our method from the point of view of a single node. In the first step, after scheduling transmissions between adjacent sensors to avoid interference [37, 38], each node broadcasts a pilot signal which includes its power level. After listening for pilots, each node broadcasts an acknowledgement packet at the maximum radio range of the pilots it has received. The outdegree of a node is defined as the number of acknowledgments it receives (i.e., the number of nodes it covers). In the second step the outdegree of each node is broadcasted among its neighbors. Finally, in the third step, nodes choose between becoming a raw or aggregating node in a sequential way. The nodes first run random timeouts during which they listen to their neighbors broadcasting their raw-aggregating node decision. If one node hears a neighbor declaring that it has become a raw node, it assigns itself as aggregating node and broadcasts its label. Once the timeout is over, the node compares its outdegree with that of its still unassigned neighbors. Nodes reach this step of the algorithm if during the timeout all the neighbors that have broadcasted their parity are aggregating nodes, and if at this moment there are still neighbors that have not declared themselves as raw or aggregating nodes. If its outdegree is the greatest within its neighborhood, then the node declares itself as a raw data node and transmits its label. Otherwise, it restarts another timeout and listens to one of its neighbors turning into a raw data node.

Note that each node only needs 4 communications to complete its task: one broadcast for its pilot signal, one broadcast of an ACK message, a third one to share its outdegree, and a final communication when the node announces its assignment. Thus, the number of communications required to find a minimum set-cover is $O(N)$, where N is the total number of nodes in the network. Although the resulting set cover is not as close to the global minimum as other more complex distributed solutions [12, 13], our proposed algorithm requires a considerably lower number of

local communications, and it is valid for any kind of graph. Its low complexity and communication costs make this distributed heuristic appropriate for real-time implementations. In the following section, we will show how to improve the set cover by allowing nodes to increase their radio range and to cover current raw nodes.

2.2.3 Distributed Set Cover Modifications

The assignment given by Algorithm 2 can be improved by allowing nodes to change their radio range. However, this increase in radio range implies a *trade-off*, because the mean distance between raw data nodes and aggregating nodes will also increase. This increase in distance leads to a decrease in the level of correlation between raw/aggregating neighbors. Thus, a higher rate will be required in order to represent data measured by aggregating nodes. Additionally, an increase in radio range also implies more communication costs in the network (nodes need more power to cover a greater distance). In this section, we propose two distributed set cover modifications which exploit this *trade-off* with decisions made only using local information, and reduce the overall energy consumption in the network in comparison with standard set covering approaches.

Assume that nodes can transmit in a radio range within the interval $[R_n^{min}, R_{max}]$, where R_n^{min} is the minimum radio range that connects each node with its parent in the routing tree. We consider a discrete number of possible radio ranges $\{R_n^1, R_n^2, R_n^3\} \in [R_n^{min}, R_{max}]$. These are the radio ranges that allow node n to cover different sets of nodes. All other values within the interval only add more communication cost to the total energy consumption in the network. Each node can know the different R_n^m values by broadcasting a pilot signal with radio range R_{max}. Then, all nodes that have received the pilot signal send an ACK message with the same R_{max}. By comparing the attenuation that each ACK has suffered, the node can learn the different discrete radio ranges in order to reach its neighbors. With this procedure, the total number of communications per node to allow all sensors

determine the discrete radio ranges is equal to $1 + \mathcal{I}_n$, where $\mathcal{I}_n$ is the indegree of node n. In other words, each node has to transmit one pilot signal and $\mathcal{I}_n$ ACK messages. The additional cost per node can be quantified as $(1 + \mathcal{I}_n)R_{max}^2$.

Let $\rho(n)$ be the parent node of node n in the given routing tree $T = (V, E_T)$). We denote $g(n)$ the cost of routing one bit from node n to the sink. Assume that raw data can be encoded using B_r bits, and c_n is a *rate-reduction ratio* that reflects the level of data compression in node n. We consider $c_n \in (0, 1]$. Thus, following this notation, the cost of routing raw data from raw data node n to the sink is $B_r g(n)$, and the cost of routing compressed data from aggregating node n to the sink is $c_n B_r g(n)$. Note that we can rewrite $g(n)$ as $B_r(R_n^2 + g(\rho(n)))$. Finally, assume $\widehat{\mathcal{D}}_n$ to be the set of new raw data nodes that are covered by node n after increasing its radio range from R_n to $\widehat{R}_n$, i.e., $\widehat{\mathcal{D}}_n = \left\{ m \in \mathcal{D} | m \notin R_n, m \in \widehat{R}_n \right\}$ where $\mathcal{D}$ is the complete set of raw data nodes.

With this notation, the cost of routing data towards the sink from raw data node n and nodes in $\widehat{\mathcal{D}}_n$, after increasing node n's radio range, will switch from C_1 to C_2.

$$C_1 = B_r(R_n^2 + g(\rho(n))) + \sum_{m \in \widehat{\mathcal{D}}_n} B_r g(m) \tag{2.1}$$

$$C_2 = B_r(\widehat{R}_n^2 + g(\widehat{\rho}(n))) + \sum_{m \in \widehat{\mathcal{D}}_n} c_m B_r g(m) \tag{2.2}$$

Moreover, it may be also worth to increase the radio range of an aggregating node n and switch its assignment to raw data node. These changes will allow it to cover some current raw data nodes in the set $\widehat{\mathcal{D}}_n$. In this case, there is a exchange of roles between node n and nodes in $\widehat{\mathcal{D}}_n$. Now, the cost of routing data towards the sink from both n and nodes in $\widehat{\mathcal{D}}_n$ will change from C_3 to C_4.

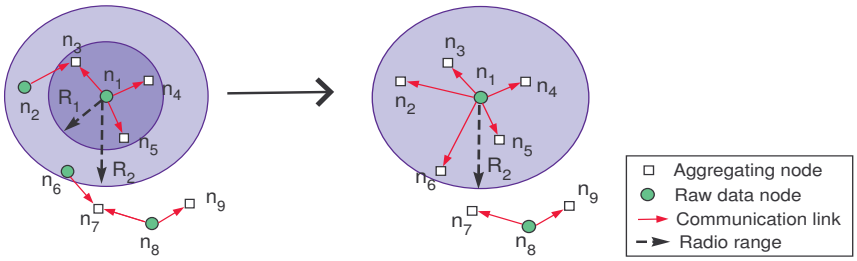
$$C_3 = c_n B_r(R_n^2 + g(\rho(n))) + \sum_{m \in \widehat{\mathcal{D}}_n} B_r g(m) \tag{2.3}$$

$$C_4 = B_r(\widehat{R}_n^2 + g(\widehat{\rho}(n))) + \sum_{m \in \widehat{\mathcal{D}}_n} c_m B_r g(m) \tag{2.4}$$

If C_1 (resp. C_3) is greater than C_2 (resp. C_4), we are successfully exploiting the *trade-off* that arises from increasing the radio range, and thus, we are reducing the energy consumption in the network. In other words, each node needs to find the radio range $\widehat{R}_n$, among its possible values, that maximizes the difference $\delta = C_1 - C_2$ (resp. $\delta = C_3 - C_4$). Following this procedure, the overall cost in the network will always go down. However, note that before changing the parity of any node, we have to ensure that a set cover will still exist. This means that, before turning some raw data nodes into aggregating nodes, we have to check that the aggregating nodes that are now covered by these raw data nodes will still be covered after the change. Note that these changes in assignment are done one node at a time, so it is only necessary to check for a set cover within two hops of each node. This can be done by a simple exchange of ACK/NACK messages between the nodes involved. First of all, the node n that wants to increase its radio range (if it is an aggregating node, it will also change its parity) asks the raw data nodes in $\widehat{\mathcal{D}}_n$, if it can cover them. Then, these adjacent nodes send a request to all their aggregating neighbors for ensuring that they will be covered by another node after the change. If nodes in $\widehat{\mathcal{D}}_n$ only receive ACKs as answers to their request, they can switch their parity and inform node n that it is allowed to cover them. By this procedure, the existence of a set cover will be guaranteed.

Figure 2.2 illustrates the basic intuition behind increasing the radio range of some raw data nodes. In Fig.2.2(a) we allow node n_1 to change its radio range from R_1 to R_2. By doing so, raw data node n_1 is now able to cover nodes n_2 and n_6, both of which in-turn become aggregating nodes (Fig.2.2(b)). Note that after the change of parity, there are no aggregating nodes uncovered. Aggregating nodes that were covered by nodes n_2 and n_6 are now covered by n_1 or some other raw data nodes.

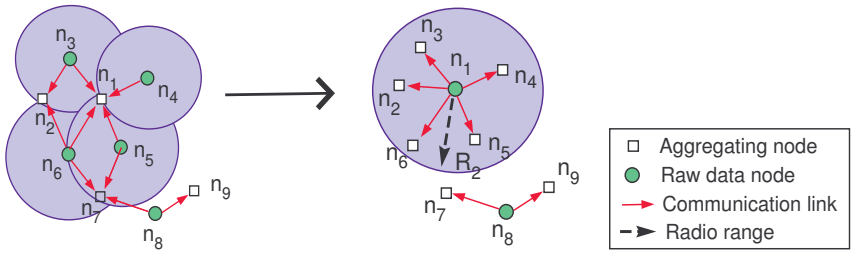
On the other hand, Figure 2.3 shows the modification process from the point of view of an aggregating node. Observe that in Fig. 2.3(a) node n_1 is covered by nodes n_3, n_4, n_5, and n_6. In Fig 2.3(b), we change node n_1 to be a raw data node and we increase its radio range to R_2. By doing so, n_1 is now covering nodes from n_2



(a) (b)

Figure 2.2: Min. Set Cover Modification for raw data nodes. (a) RANA before increasing n_1's radio range from R_1 to R_2. (b) Modified RANA. Raw data node n_1, after increasing its radio range, is now covering nodes n_2 and n_6, which can switch their parity into aggregating nodes.

to n_6, which can become aggregating nodes. Again, we observe that all aggregating nodes in the network are still covered after the modification.



(a) (b)

Figure 2.3: Min. Set Cover Modification for aggregating nodes. (a) Original RANA in which n_1 is an aggregating node. (b) Resulting RANA after changing n_1's parity to raw data and increasing its radio range. Now, n_1 covers former raw data nodes n_3, n_4, n_5, and n_6, which can be switched to aggregating nodes.

For implementing these modifications in a distributed manner, nodes need to get or estimate several parameters. First of all, they need to know the c_n values. Each node can estimate these values in a distributed way after the exchange of

some training data with all its neighbors. Thus, for each neighbor m_k, node n can compute a $c_n(m_k) = c_{n,k}$ as the worst case c_n if node n can only listen to raw data from its neighbor m_k. Another option would be to model c_n as a function inversely proportional to the correlation between neighboring nodes. For example c_n can be assumed to decrease monotonically with distance. However, this kind of modeling provides very conservative estimates of c_n. Second, nodes also need to obtain their routing cost $g(n)$ and the routing cost of all nodes in $\widehat{\mathcal{D}}_n$. Moreover, in order to exactly reproduce the previous equations, nodes need to know, for each $\widehat{R}_n$, which would be its new parent node $\widehat{\rho}(n)$ and the routing cost $g(\widehat{\rho}(n))$.

All these requirements made these modifications hard to be implemented in a distributed way, because nodes need a large number of local communications in order to make each decision. Instead of doing that, we propose some approximations to equations (2.1)-(2.4) that allow nodes to improve the original set cover resulting from Section 2.2.2 with information that is already available to them. Firstly, nodes can assume c_n to be a constant value between $(0, 1]$ for all neighbors. We have noticed that in a dense and uniformly deployed network, the difference in rate-reduction ratio between neighboring nodes is not very significant. For example, in our experiments, and for simplicity, we have considered $c_n = c = 0.5$. Second, we also assume that the difference in routing cost between the current parent node and the one after the increase in radio range is negligible, i.e., $g(\rho(n)) \approx g(\widehat{\rho}(n))$. Finally, we have observed that in dense networks, node n will often be located near the center of its neighbors in $\widehat{\mathcal{D}}_n$, which means that its routing cost is close to the average of the routing costs in its neighborhood: $g(n) \approx \sum_{m\in\widehat{\mathcal{D}}_n} g(m)/|\widehat{\mathcal{D}}_n|$. Thus, $\sum_{m\in\widehat{\mathcal{D}}_n} B_r g(m) \approx B_r|\widehat{\mathcal{D}}_n|g(n)$ and $\sum_{m\in\widehat{\mathcal{D}}_n} c_m B_r g(m) \approx cB_r|\widehat{\mathcal{D}}_n|g(n)$.

After all these approximations, we can simplify (2.1)-(2.4) as:

$$\widehat{C}_1 = B_r(R_n^2 + g(\rho(n))) + B_r \left|\widehat{\mathcal{D}}_n\right| g(n), \tag{2.5}$$

$$\widehat{C}_2 = B_r(\widehat{R}_n^2 + g(\rho(n))) + cB_r \left|\widehat{\mathcal{D}}_n\right| g(n), \tag{2.6}$$

$$\widehat{C}_3 = cB_r(R_n^2 + g(\rho(n))) + B_r \left|\widehat{\mathcal{D}}_n\right| g(n), \tag{2.7}$$

$$\widehat{C}_4 = B_r(\widehat{R}_n^2 + g(\rho(n))) + cB_r \left|\widehat{\mathcal{D}}_n\right| g(n). \tag{2.8}$$

Note that now, for implementing the distributed set cover modifications, all the necessary parameters in (2.5)-(2.8) are always available at node n. Even though the modifications are based on some conservative approximations, they allow nodes to improve the original set cover in a distributed manner, and reduce the overall energy consumption in the network.

Algorithm 3 and Algorithm 4 show our distributed modifications from the point of view of a raw data node and an aggregating node, respectively. In the worst case, nodes will need to execute algorithms 3 or 4 during $O(\mathcal{O}_{max})$ rounds, where $\mathcal{O}_{max}$ is the maximum outdegree for each node. Note that each node will check if it is worthwhile increasing its radio range to at most $\mathcal{O}_{max}$ possible radio ranges. Moreover, each raw data node will ask at most $\mathcal{I}_{max}$ times to its aggregating neighbors if they are covered by some other raw node in order to know if it can change its parity. Assume that $\mathcal{I}_{max}$ refers to the maximum indegree for each node. Then, each raw data node will also need to send at most $\mathcal{I}_{max}$ ACKs/NACKs to inform its neighbors about that possibility. On the other hand, each aggregating node will send, in the worst case, $\mathcal{I}_{max}$ ACKs to inform to raw data neighbors that they can switch their parity. Thus, the total number of communications for implementing Algorithms 3 and 4 for each raw data node, including the necessary communications to determine the discrete radio ranges, is equal to $1 + \mathcal{O}_{max} + 3 \cdot \mathcal{I}_{max}$. For each aggregating node that value is equal to $1 + \mathcal{O}_{max} + 2 \cdot \mathcal{I}_{max}$. Therefore, in the worst case, nodes need $O(\Delta_{max})$ communications to implement both distributed algorithms, where Δ_{max} is the maximum degree of each node. In the simulations described in Section 2.3 we have implemented the distributed algorithms in net-

works with 70 nodes randomly placed in a 600 x 600 grid. In our experiments, the mean value for the maximum indegree and outdegree is approximately 7. However, note that this value will change if we have different density and number of nodes in the network. Moreover, the indegree and outdegree are also dependent on how the maximum radio range is defined for each node. Figure 2.4 summarizes the maximum number of communications per node needed to implement the distributed algorithms presented in this chapter.

We have observed that for uniformly deployed networks it is better to first let the current raw data nodes check if by changing their radio ranges they can improve the present set cover, and then do the same with the aggregating nodes. On the other hand, for clusterized networks it is better to do it in the reverse order. The order in which nodes implement their corresponding distributed algorithm can be set by using different random timeouts in each sensor.

In conclusion, by using the algorithms described in this section, nodes can verify if it is worth to increase their radio ranges just by knowing how many raw data nodes $\left|\widehat{\mathcal{D}}_n\right|$ they can cover at each time. These approximations still reduce the number of raw data nodes in the network while they are suitable in a practical setting due to the low number of communications required.

Algorithm	Number of communications per node
Algorithm 2	4
Algorithm 3 + Algorithm 4	Raw data nodes $= 1 + \mathcal{O}_{max} + 3 \cdot \mathcal{I}_{max}$
	Aggregating nodes $= 1 + \mathcal{O}_{max} + 2 \cdot \mathcal{I}_{max}$

Figure 2.4: Number of communications per node using the different distributed algorithms. $\mathcal{O}_{max}$ and $\mathcal{I}_{max}$ refer to the maximum outdegree and indegree for each node.

Algorithm 2 Distributed Heuristic for Minimum Set Cover

```
1: Broadcast a pilot signal adding the power level used as information.
2: Run a timeout and listen to all neighbor's pilot signals.
3: Broadcast an ACK message at the maximum power of all received pilots.
4: Outdegree = Number of ACK's received.
5: Broadcast Outdegree.
6: Receive neighbor's outdegree.
7: while still unassigned do
8:    Run a random timeout.
9:    while timeout > 0 do
10:      Listen to decisions of neighbors.
11:      if One neighbor becomes raw data node then
12:         Stop timeout
13:         Become aggregating node
14:         Broadcast decision
15:      end if
16:   end while
17:   if There are still unassigned neighbors then
18:      if Outdegree ≥ Unassigned neighbor's outdegree then
19:         Become raw data node.
20:         Broadcast decision.
21:      else
22:         Restart timeout.
23:      end if
24:   else
25:      All neighbors are Aggregating Nodes
26:      Become raw data node.
27:      Broadcast decision.
28:   end if
29: end while
```

Algorithm 3 Approximation-based Distributed Min Set Cover Modification for raw data nodes

Possible Radio Ranges = $[R_n^{min}, R_n^1, ..., R_{max}]$
Run random timeout
while Timeout > 0 **do**
  Listen if some other neighbor wants to cover you.
  if Request for being covered received **then**
    Ask Aggregating neighbors if they are covered by another raw node.
    if All Aggregating neighbors are covered by some other raw data node **then**
      Send ACK
    else
      Send NACK
      Restart Timeout
    end if
  end if
  if CONFIRMATION received **then**
    Change parity to Aggregating Node
  end if
end while
$\delta_{max} = 0$
$\widehat{R}_n = R_n^{min}$
for $R_n = R_n^1$ to R_{max} **do**
  Send pilot signal at power level R_n^2
  if All raw nodes in $\widehat{\mathcal{D}}_n$ accept being covered **then**
    Calculate $\widehat{C}_1$ and $\widehat{C}_2$:
    $\widehat{C}_1 = B_r(R_n^2 + g(\rho(n))) + B_r \left|\widehat{\mathcal{D}}_n\right| g(n)$
    $\widehat{C}_2 = B_r(\widehat{R}_n^2 + g(\rho(n))) + cB_r \left|\widehat{\mathcal{D}}_n\right| g(n)$
    $\delta_i = \widehat{C}_1 - \widehat{C}_2$
    if $\delta_i > \delta_{max}$ **then**
      δ_{max}=δ_i
      $\widehat{R}_n = R_n$
    end if
  end if
end for
if $\widehat{R}_n \neq R_n^{min}$ **then**
  Send CONFIRMATION at power level $\widehat{R}_n^2$
  Fix Radio Range at $\widehat{R}_n$
  Update parent in the tree
end if

Algorithm 4 Approximation-based Distributed Min Set Cover Modification for aggregating nodes

Possible Radio Ranges = $[R_n^{min}, R_n^1, ..., R_{max}]$
Run random timeout
while Timeout > 0 **do**
  Listen if a new raw data node wants to cover you.
  if Pilot signal received **then**
    Update number of raw data nodes that cover you.
  end if
  Listen if any of your current raw data nodes request permission for changing its parity.
  if Request received **then**
    if Covered by more raw nodes that the ones who have sent the request **then**
      Send ACK
      Update number of raw data nodes that cover you.
    end if
  end if
end while
$\delta_{max} = 0$ and $\widehat{R}_n = R_n^{min}$
for $R_n = R_n^1$ to R_{max} **do**
  Send Request at power level R_n^2
  if All raw nodes in $\widehat{\mathcal{D}}_n$ accept being covered **then**
    Calculate $\widehat{C}_3$ and $\widehat{C}_4$:
    $\widehat{C}_3 = cB_r(R_n^2 + g(\rho(n))) + B_r \left|\widehat{\mathcal{D}}_n\right| g(n)$
    $\widehat{C}_4 = B_r(\widehat{R}_n^2 + g(\rho(n))) + cB_r \left|\widehat{\mathcal{D}}_n\right| g(n)$
    $\delta_i = \widehat{C}_3 - \widehat{C}_4$
    if $\delta_i > \delta_{max}$ **then**
      δ_{max}=δ_i and $\widehat{R}_n = R_n$
    end if
  end if
end for
if $\widehat{R}_n \neq R_n^{min}$ **then**
  Send CONFIRMATION at power level $\widehat{R}_n^2$
  Fix Radio Range at $\widehat{R}_n$
  Change parity to Raw Data Node.
  Update parent in the tree
end if

2.3 Simulation Results

We now compare the performance of our distributed algorithms against existing centralized methods. Specifically, we compare our low complexity distributed algorithm for solving the set covering problem, against the centralized greedy solution in [25], the Haar-like tree-based transform with 1 level of decomposition proposed in [35], and against raw data gathering without compression. We demonstrate the benefits of allowing some nodes to change their radio range by comparing the approaches mentioned before with the distributed modifications proposed in Section 2.2.3.

2.3.1 Experimental Setup

In our performance evaluation, we use an AR-2 model to generate simulation data with high spatial correlation. Nodes are randomly deployed in a 600 x 600 grid, and the sink is placed in the center of the network. Data is routed towards the sink following a shortest-path tree (SPT). In order to compare the energy consumption, we use the cost model proposed in [43, 10], where the energy of transmitting k bits over a distance D is $E_T(k, D) = E_{elec} \cdot k + \varepsilon_{amp} \cdot k \cdot D^2$ Joules, and the energy consumption related with the reception of k bits is defined as $E_R(k) = E_{elec} \cdot k$ Joules. In both formulas, the $E_{elec} \cdot k$ term captures the energy dissipated by the radio electronics while processing the k bits, and $\varepsilon_{amp} \cdot k \cdot D^2$ denotes the additional energy consumed in the amplification of the signal for ensuring a reasonable signal power at the receiver. We are not considering the energy dissipated when nodes perform computations, because the computation costs are typically negligible compared with transmission and reception costs.

In both the distributed and centralized MSC cases, we implement the simplified graph-based lifting transform presented in Section 1.3, where we use an adaptive prediction NLMS filter [34] for generating the transform coefficients in each aggregating node. Raw data is represented using $B_r = 12$ bits, and in each epoch nodes transmit $M = 50$ measurements taken at M different times. The transform coef-

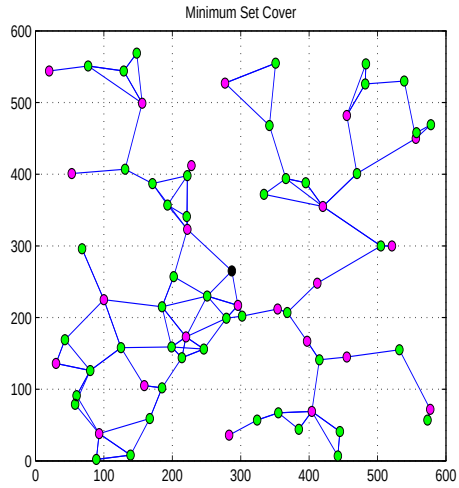
ficients are always quantized using a dead-zone uniform scalar quantizer and are encoded using an adaptive arithmetic coder [33].

We are going to measure the performance as the trade-off between the total energy consumption at each quantization level and the reconstruction quality (in terms of the signal-to-quantization-noise ratio) expressed in dB. Higher SNR for a fixed cost implies higher fidelity reconstruction of the data.

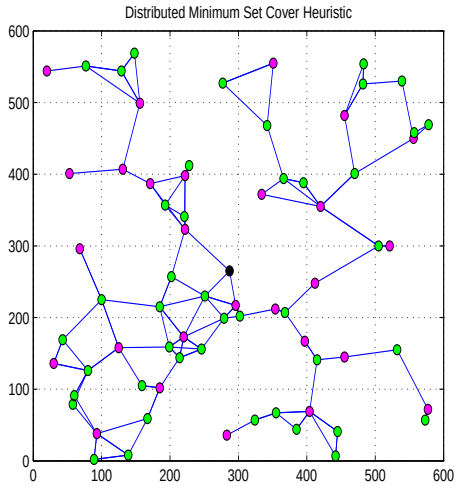
2.3.2 Performance Evaluation

We have simulated the different algorithms for computing the RANA in a 70 node network. Figures 2.5 and 2.5(c) illustrate the resulting minimum set covers (MSC) using the greedy centralized algorithm and our proposed distributed heuristics. Fig. 2.5(a) shows the centralized MSC algorithm (Algorithm 1), the distributed low-complexity heuristic (Algorithm 2) is shown in Fig. 2.5(b), and the combination of all our distributed algorithms is shown in Fig. 2.5(c). Note that our low complexity distributed heuristic has most raw nodes (pink circles). However, it can be seen that the number of raw data nodes can be reduced by using Algorithms 3 and 4, and allowing some nodes to increase their radio range.

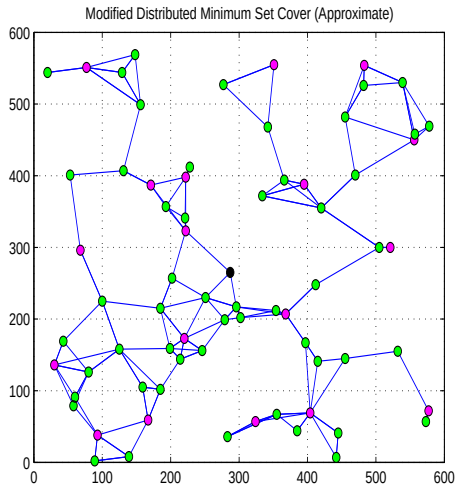
In Fig 2.6 we can see that the reduction of raw data costs is directly proportional to the reduction of the overall cost in the network. Thus, in approaches where the raw data cost is higher, the total cost in the network is also higher, with the distributed modifications providing lowest raw data cost and hence lowest overall cost.



(a) Centralized MSC



(b) Distributed MSC



(c) Modified Distributed MSC

Figure 2.5: Comparison of different RANAs with various minimum set covering algorithms. (a) Resulting RANA after aplying Algorithm 1. (b) RANA solved in a distributed way using Algorithm 2. (c) Resulting RANA after modifying the assignment in (b) with Algorithm 3 and Algorithm 4. Pink circles represent raw data nodes, and green circles are aggregating nodes.

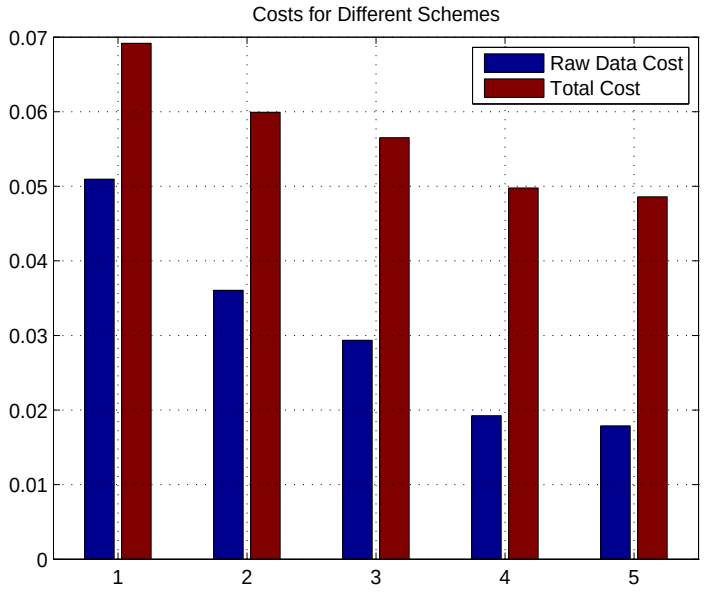


Figure 2.6: Raw-costs and total costs for different RANA approaches. 1: Tree-based Haar-like lifting transform with 1 level of decomposition, 2: Distributed MSC, 3: Centralized MSC, 4: Distributed modification with approximate values, 5: Distributed modification with exact values.

Figure 2.7 shows the cost-distortion curves. It can be seen that implementing the RANA as a minimum set covering problem reduces the overall energy consumption in the network in comparison to the tree-based assignment. Thus, we can deduce that reducing the number of raw data nodes in the network is a crucial factor in the optimization, in terms of energy efficiency, of a lifting-based transform in WSN. As expected, all approaches outperform raw data gathering.

Among all the MSC algorithms, note that the combination of our distributed algorithms (Algorithms 2, 3, and 4) does the best. Note that the exact and approximate modifications are nearly identical. Although the low complexity distributed heuristic at first does worse than the centralized greedy algorithm, after the modifications we have more than 5 dB increase in SNR for a fixed cost. It is worth mentioning that the LP-optimized solution presented in [26] would outperform our distributed modifications, as we do not consider the large number of combinations

of assignments and radio range choices that the LP-optimization method takes into account. However, our proposed algorithms are significantly simpler and can be implemented in a practical setting with low additional cost.

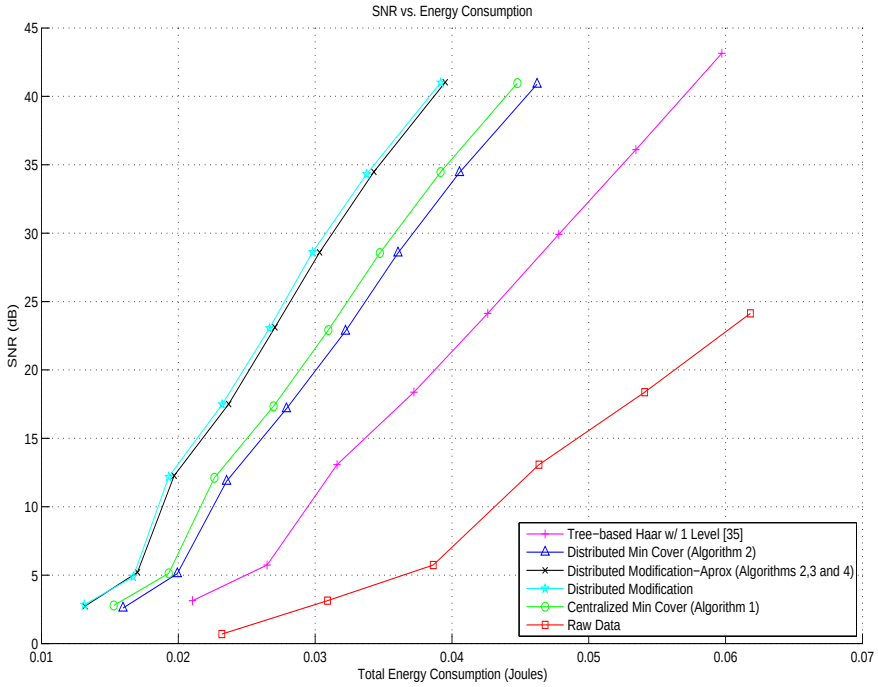


Figure 2.7: Cost-distortion curves for various set covering algorithms. All outperform tree-based assignment and raw data gathering. Distributed heuristic with set cover modification does the best.

Chapter 3

Graph-based Wavelet Transforms in Multisink WSN

3.1 Introduction

In this chapter, we focus on scenarios in Wireless Sensor Networks (WSN) where nodes have to send data to several destinations (or sinks) in order to perform a certain task. The growing interest in WSN and the recent technological advances in that area have developed a broad range of applications in where the basic single sink model considered in the previous chapters is not longer valid. For example, in wireless sensor and actor networks (WSANs) [2], the network consists of several low-cost, low power devices (sensors) which sense phenomena from the environment, and send the measurements to some other resource rich nodes (actors) equipped with better processing capabilities. These more powerful devices are then responsible for performing different actions based on decisions made with the received data. Therefore, in some applications, we may have different kinds of actors which need all the measured data to develop their task. These networks can be an integral part of systems for applications such as battlefield surveillance, home automation, microclimate control, or environmental monitoring. Another example is related with next generation networks, which will integrate WSN, GSM, and Internet. In

this case, sinks can be conceived as gateways connecting the WSN with all the other systems. Thus, in some scenarios, sensors will need to communicate simultaneously to more than one sink in order to provide data via the different networks in real time.

Although scenarios and applications with multiple sinks are increasingly being proposed, there is a lack of work on spatial compression and distributed data gathering in this kind of networks. In this chapter, we extend the graph-based lifting transform of Section 1.3 to the case where all data has to be gathered in more than one sink. We noted that the *processing strategy* used in that transform is general for any kind of *routing strategy* (i.e., unicast, multicast or broadcast). Therefore, after all raw data nodes broadcast their data, and aggregating nodes compute the detail coefficients, both uncompressed and compressed data can be forwarded towards any number of destinations following some kind of routing algorithm. Thus, in order to make the transform suitable for WSNs with more than one sink, we just need to find an optimal *routing strategy* for multicast communications. As an alternative to our proposed method, there are several approaches that use network coding for multicasting over WSN [22, 21]. However, in contrast to our work, network coding is based on operations over a finite field, and it does not exploit the correlation between spatially close sensors.

The present chapter is organized as follows. In Section 3.2 we discuss several existing methods for multicasting in wireless networks, and we present a new routing algorithm based on the Steiner tree problem, which exploits the broadcast property of wireless communications. Section 3.3 summarizes the scheme for spatial compression in multisink networks, and in Section 3.4 we simulate and compare the performance of the graph-based lifting transform using different multicast algorithms.

3.2 Multicast Routing Algorithms

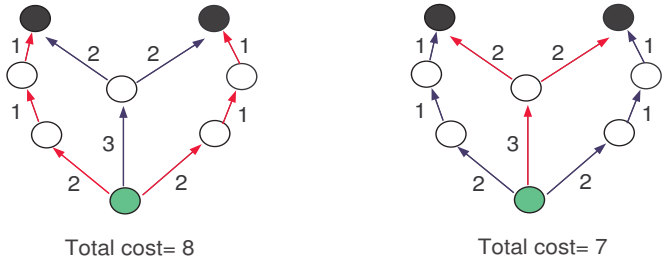
3.2.1 Existing methods

In our current scenario, we wish to send information from all the sensors to a subset of other nodes (destinations/sinks) in a communication network. Remember that in WSNs, sensors are low-cost, energy constrained devices equipped with a wireless communication system. Since sensors have a limited radio range, nodes which are not able to communicate directly with all the sinks need to send their data via multihop paths using other nodes as relay stations.

A simple strategy in order to send the data from a node to a set of destinations would be to compute the union of shortest path trees from the source to each sink. This approach would minimize the distance (or cost) from the sender to each receiver, but it ignores the fact that sending data to the different sinks by sharing some links in the tree can reduce the overall transmission cost. As illustrated in Fig. 3.1, this is not the best solution for multicast routing. The problem of finding a tree that, in a communication graph $G(V, E)$, spans a subset $S \in V$ with minimal total distance on its edges is referred as the minimum Steiner tree (MST) problem. Because finding a minimal Steiner tree for any given graph is NP-complete [14], it is necessary to solve the problem with some kind of heuristic algorithm, such as those proposed in [16, 31].

Note that the MST is only an optimal solution for wired networks, because it considers that a node n needs k transmissions to send a packet to k of its neighbors. On the contrary, in wireless communications (see Fig. 3.2), the transmission cost for multicasting a data packet from a node n to k of its neighbors is equal to the cost of transmitting a single packet to the most distant receiver. Thus, a good approach for building routing trees in wireless networks has to consider what we are going to refer in what follows as the *wireless broadcast advantage*.

Ruiz and Gomez-Skarmeta [32] propose an alternative to the MST for wireless multihop networks, which considers the wireless broadcast advantage. In their work,



(a) Union of SPTs (b) MST

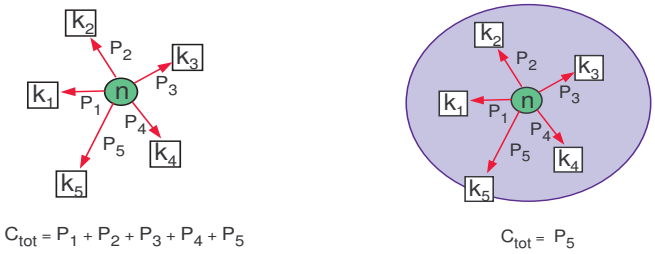
Figure 3.1: Difference in cost for several multicast trees. The numbers associated to each link are communication costs. In a) the union of SPTs minimizes the cost between the source and each sink. In b) the MST minimizes the overall edge cost between the source and both receivers.

the problem is re-formulated in terms of minimizing the number of transmissions needed to reach all the destinations. The minimum cost tree is defined as the one which connects the multicast group using the minimum number of communications, and enhanced heuristics are proposed to approximate such trees minimizing the number of forwarding nodes.

A simulation-based performance comparison of SPTs, the MST as in [16], and the method in [32] is presented in [27] for multicasting in wireless mesh networks. As proved in that work, the heuristic proposed by Ruiz and Gomez-Skarmeta only builds optimal trees in highly dense networks. Although the MST heuristic is a link-based approach originally proposed for wired networks, it is still the algorithm with minimum total edge cost and minimum number of transmissions in comparison with the other methods.

Another classical approach for building multicast trees is to *prune* a previously defined broadcast tree. For example, we can construct the well-known minimum-cost spanning tree, which spans all nodes in the network with minimal total cost, and then delete the unnecessary links until the leaf nodes in the tree are only the source and the set of receivers. This procedure is illustrated in Figure 3.3.

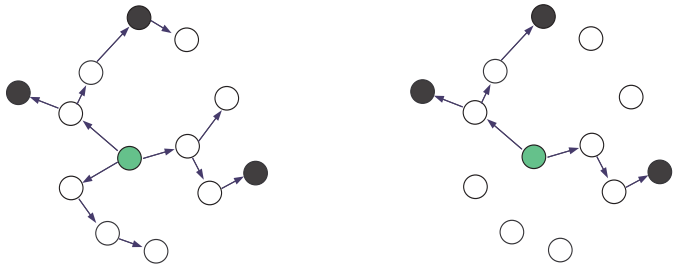
Following this intuition, Wieselthier *et al* [46], present the Broadcast Incremental Power algorithm (BIP) and a *pruned* multicast alternative (i.e., MIP). Their meth-



(a) Wired Network

(b) Wireless Network

Figure 3.2: The Wireless Broadcast Advantage. (a) In a wired network the total cost of sending a message from n to all its k_i neighbors is $C_{tot} = \sum P_i$. (b) Instead in a wireless network, it can be done in a single transmission with $C_{tot} = max\{P_i\}$.



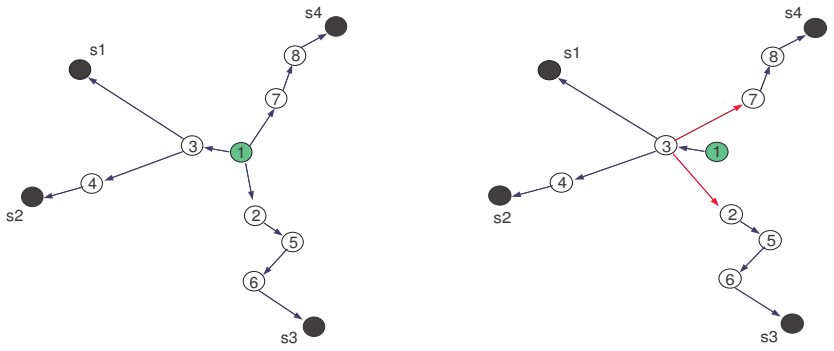
(a) Minimum Spanning Tree (Broadcast)

(b) Pruned-Minimum Spanning Tree (Multicast)

Figure 3.3: Example of the *pruning* method for multicast trees. The broadcast tree in (a) is *pruned* until all the leaf nodes are only the source and the set of receivers, as in (b).

ods are based on the same idea as most minimum-cost spanning tree algorithms (such as Prim's algorithm). They iteratively grow a minimum-power tree rooted at the source, incorporating one node at a time until all nodes are included in the tree. In contrast with all other link-based approaches, they exploit the aforementioned wireless broadcast advantage. The MIP procedure is summarized as follows. First of all, the source's nearest neighbor is added to the tree. Then, there are two alternatives: either the nearest neighbor of the last added node is incorporated to the tree, or the source increases its radio range to cover a second node. The alternative with minimum additional cost is chosen. The procedure is continued until all nodes

are included in the tree. Once the algorithm is finished, a *sweep operation* is run. The intuition behind the sweep procedure is shown in Fig.3.4. Note that node 3's radio range is sufficient to reach nodes 7 and 2. Thus, without loss of connectivity, node 1 can decrease its radio range. By doing so, the energy consumption in the network is reduced. The authors prove that MIP outperforms the P-MST (Pruned-Minimum Spanning Tree) and the union of SPTs. However, since a good broadcast tree is constructed looking to the global picture of overall energy savings, *pruning* it back does not necessarily give an efficient multicast solution. *Pruning* a broadcast tree in order to obtain a multicast routing scheme could be useful when the multicast group is large (i.e., more than the 65% of all nodes in the network) [23].



(a) Pre-Sweep Multicast Tree

(b) Post-Sweep Multicast Tree

Figure 3.4: The sweep operation. (a) Node 1's radio range can be reduced since node 3 can cover nodes 2 and 7 with its current radio range. (b) The modification reduces the energy consumption in the network

To the best of our knowledge, the algorithms Multicast Incremental Power with Potential Power Savings (MIP3S) and its simplified version MIP3S-b are shown to work better than all other known methods [23]. They are based on the same idea as the *sweep operation*. Their intuition is that, while constructing a multicast (or broadcast) tree rooted at the source, sometimes we need to increase the power level of a certain node n in order to cover some new nodes that are not yet in the tree. This increase could make the power assignment at some previously added nodes

redundant, in the sense that n is now also covering nodes which have been added to the tree before. Thus, as seen before in Fig. 3.4, we can reduce the radio range of some nodes without losing connectivity and saving energy. This idea is referred to in [23] as the *incremental power with potential power saving* (IP3S). In MIP3Sb, the shortest path P between a node which is already in the tree and an uncovered destination is added at each iteration. Then, after adding P to the multicast tree, it is required to verify whether after updating the radio range assignment in the nodes participating in P, any of them is now also covering some other nodes that were added to the tree in a previous iteration. Therefore, if node $n \in P$ is now also covering node v, which was in the tree before adding the path P, the radio range of the node that is currently covering node v is reduced.

MIP3S is a more complex version in which, instead of adding the shortest path at each iteration, a path is added that involves the least total incremental cost minus the total potential power saving (which in MIP3Sb is calculated afterwards). In other words, for each path connecting a node which is currently in the tree with an uncovered sink, a calculation is made of the total incremental cost and the amount of energy that would be saved with the new power assignment if that path is added to the tree. Then, the path with minimum *"incremental cost minus potential power saving"* is chosen. Note that this alternative requires considerable more work than the previous one, since for each node in the multicast tree we have to find all the possible paths to each uncovered sink. Moreover, for each of these paths we have to run a power reduction method (i.e., the sweep operation) in order to know the amount of energy that will be saved if that path is added to the tree.

The procedure of MIP3Sb is illustrated in Figure 3.5. First of all, in Fig. 3.5(a), the shortest path P_1 from the source (node 1) to any of the sinks, i.e., $s_i \in \{s_1, s_2\}$, is added to the tree. Then, we need to find the shortest path P_2 from any of the previously added nodes in P_1 to s_2, since it is at this moment the only sink that lacks to be covered. As we can see in Fig.3.5(b), in this case the shortest path P_2 is rooted in node 3. Finally, once we have a new destination covered, we run a power

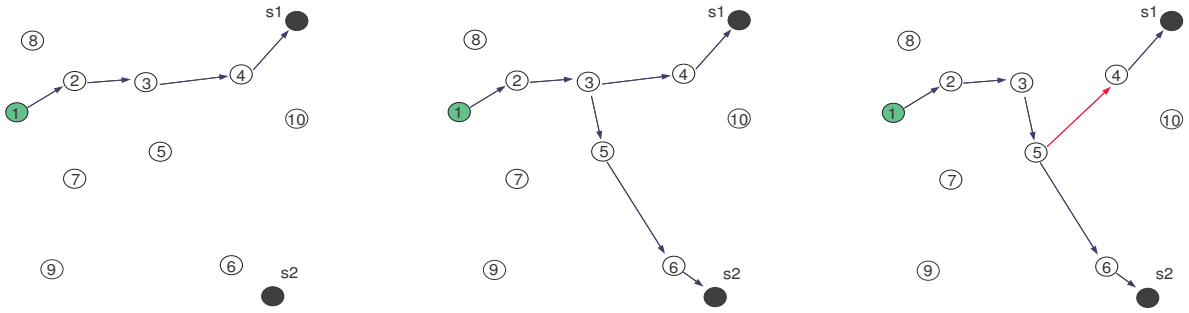


Figure 3.5: Multicast Incremental Power with Potential Power Saving (MIP3Sb). Node 1 is the source, and s_1 and s_2 are the sinks. (a) First, the shortest path from the source to one of the destinations is added to the tree. (b) Then, we add the shortest path from one node already in the tree to any of the still uncovered sinks (i.e., s2). (c) Finally, the resulting tree is modified with a power reduction method. Note that since node 5 is also covering node 4, node 3's radio range can be reduced without losing connectivity. By doing so, the overall power consumption in the network is reduced.

reduction method in a similar way as in the sweep operation. Note that now node 5's radio range is sufficient to reach node 4 (which is already in the tree). Thus, by allowing node 5 to cover node 4, we can reduce node 3's power level. However, before reducing the radio range of any node, we need to check if the tree will be connected after the modification. In other words, we need to verify that after the change there will still be a path from the source to any of the already covered destinations. In this example, node 5 can also cover node 2, but we cannot reduce node 1's radio range because if we do so the source will be disconnected from the tree.

3.2.2 The Wireless Minimum Steiner Tree

As mentioned in the previous section, in wired networks, the multicast tree can be obteined by solving the minimum Steiner tree (MST) problem, where the sender and the receivers are referred as *terminals*, and all other nodes participating in the routing tree are known as *Steiner nodes*. Therefore, the MST is the tree that

connects the source with all destinations, using other Steiner Nodes as relay stations, with minimum overall cost. Because this problem has been shown to be NP-complete [14], we need to approximate the optimal solution using heuristic algorithms [16, 31]. In WSN, where nodes are supposed to have omnidirectional antennas, and communications are done via broadcast, the Steiner Tree is no longer an optimal solution.

In this section, we introduce a new centralized algorithm for constructing minimum Steiner trees in wireless ad-hoc networks, which considers the broadcast advantage of wireless communications. We will refer to our method as the wireless-minimum Stenier tree (W-MST) algorithm. The W-MST is based on the heuristic for MST construction presented in [16], and incorporates in the procedure the MIP algorithm [46] in order to exploit the wireless broadcast advantage.

Assume that a wireless ad hoc network is modeled as a directed communication graph in the same way as in Section 1.3.1. We model the symmetric edge cost function $c(u, v)$ as the Euclidean distance between u and v raised to a fixed power α, i.e. $c(u, v) = d(u, v)^{\alpha}$, where α is an environmentally dependent real constant between 2 and 4 representing the attenuation loss of the signal. In our experiments we have considered $\alpha = 2$. Let $S \in V$ denote the multicast group (terminals), which consists of the source r and the set of destinations (or sinks), $D \in \{S - r\}$. We assume that nodes have omnidirectional antennas, and that each node u can transmit within a radio range $R_u \in [R_u^{min}, R_{max}]$. Thus, given a communication graph with the aforementioned characteristics, the problem consists in finding a multicast tree, rooted at the source r, and spanning all the nodes in D, with minimum overall cost. Due to the broadcast nature of wireless communications, we define the cost of node u as the power level needed to reach its most distant parent in the tree. Then, if node u is connected to nodes v_1 and v_2, the cost in node u is equal to $cost(u) = max\{c(u, v_1), c(u, v_2)\}$. Our goal is to minimize the sum of costs of all nodes participating in the tree, i.e., $\sum_{u \in V} cost(u)$. Note that the cost optimization

can also be referred as the minimization of the power assignment in all nodes.

What makes the minimum Steiner tree algorithm in [16] inefficient as a solution to our problem is that the multicast tree is constructed without considering that wireless transmissions are used, so that multiple nodes can overhear each communication. On the other hand, the MIP algorithm in [46] is inefficient since it considers all nodes in the network to be candidates for the multicast routing tree, and then it has to delete the unnecessary links to connect only the source with the destinations. What we want to do by merging both approaches is to build a communication graph only with the nodes that the algorithm in [16] would consider as candidates to be in the multicast tree, and then to span all these nodes taking into account the broadcast property of wireless communications by implementing the same procedure as in [46]. Therefore, the first steps of our algorithm seek to find out which of the nodes are the best candidates in order to connect the source to the sinks with minimum overall cost. Then, instead of connecting all these nodes with a minimum spanning tree as in [16], which would treat our graph as a wired network; we run the MIP algorithm which was originally proposed for wireless networks.

First of all, given the communication graph $G(V, E, c)$, and the multicast group S consisting of the source and destinations, we construct the complete undirected distance graph $G_1 = (V_1, E_1, d_1)$ where $V_1 = S$, $E_1 = \{(u_i, v_j) | u_i \in S, v_j \in S, u_i \neq v_j\}$, and $d_1(\{u_i, v_j\})$ is equal to the distance of the shortest path in G between nodes u_i and v_j. Then, we find a minimum spanning tree, T_1, of G_1, which spans all nodes in V_1 with minimum total cost. After that, we construct the subgraph G_2 with all nodes participating in T_1 and replacing each edge by the corresponding shortest path in G. At the end of this step, we know that nodes in G_2 are the best candidates to be in the multicast tree. Until this point, the procedure is the same to the Algorithm H in [16]. However, now, we are going to construct the multicast tree T_2 that connects all terminals in S using only nodes in G_2, following the same idea as in MIP [46]. Starting at the source r, we first add to the tree its closest neighbor in G_2. Then, iteratively, we consider if we should add the closest neighbor of the

last added node, or if it is better to increase the radio range of one of the other nodes already in the tree, and allow it to have more than one parent. We make that decision based on the additional cost involving each action. For example, in Figure 3.6, after the root r has increased its radio range to reach its closest neighbor (i.e., node 1), there are two options. Either we can add node 1's closest neighbor (i.e., node 2), with a cost equal to $c(1,2)$; or r can increase its radio range and reach node 3 with *incremental cost* equal to $I_r = c(r,3) - c(r,1)$.

Then, if there are leaf nodes in T_2 that are not part of the multicast group, we delete their edges. Finally, we improve T_2 by implementing the *sweep operation*, which is described in Algorithm 6. Note that we allow sinks to be also forwarding nodes. The procedure for the construction of the W-MST is summarized in Algorithm 5.

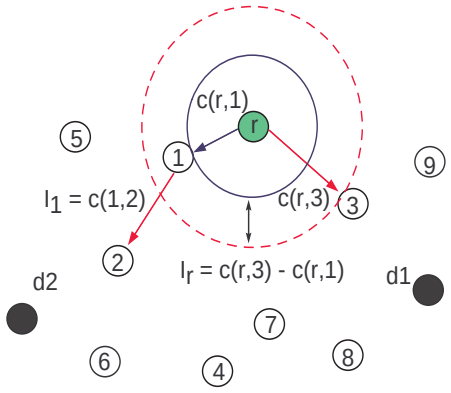
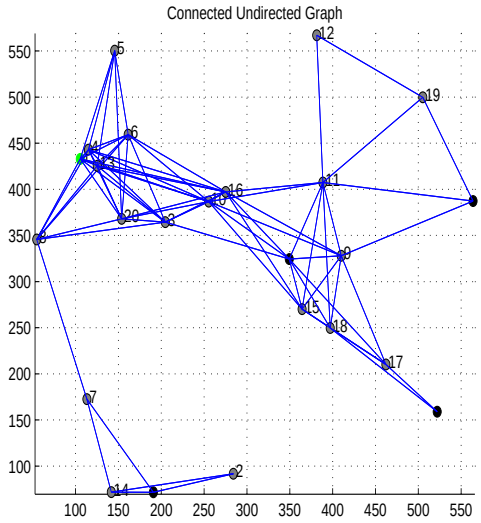


Figure 3.6: Concept of *Incremental Cost* in W-MST.

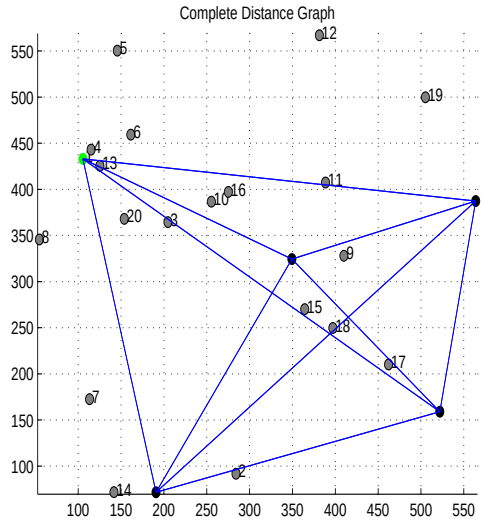
The steps for constructing the W-MST are shown in Figures 3.7 and 3.8. Figures 3.7(a) to 3.8(a) belong to the steps of the Algorithm H in [16], in which we seek to reduce the set of nodes that will be considered when constructing the multicast tree. Figures 3.8(b) and 3.8(c) belong to the MIP algorithm. The given communication graph G is represented in Fig. 3.7(a). Fig. 3.7(b) shows the complete distance graph G_1 of all nodes in the multicast group. Here, the cost of each

edge is proportional to the total distance of the shortest path between each pair of nodes in G. Then, in Fig. 3.7(c), we construct the minimum spanning tree T_1 of G_1 connecting all nodes in S. Note that each edge in T_1 represents a shortest path in G, which connects each pair of terminals using some other sensors as forwarding nodes. All nodes participating in T_1 are shown in Fig. 3.7(d). After that, we construct the communication graph G_2 between all nodes in T_1 (Fig. 3.8(a)). Note that, in contrast to the approach in [46], we have reduced the set of nodes from the network that can be in the multicast tree by constructing a graph only with the nodes in T_1. Finally, as illustrated in Fig. 3.8(b), we build the W-MST of G_2 following the MIP algorithm. As mentioned before, the resulting W-MST can be improved by applying the *sweep operation*. Note that in Fig. 3.8(c), we can reduce the power level of node 20, since node 13 has node 3 within its radio range. Thus, node 20 is not needed in the tree. In addition, node 9 can reach node 17 in a single transmission, so we also do not need nodes 15 and 18.

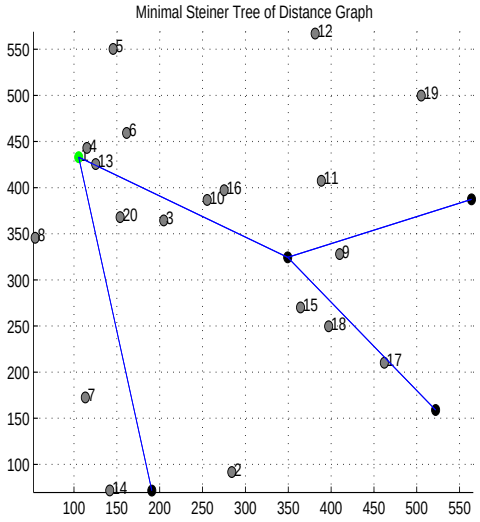
In conclusion, in this section, we have presented a new algorithm that builds a multicast routing tree, based on the minimum Steiner tree problem, and taking into consideration the broadcast property of wireless communications. This contribution can be applied not only to WSN, but it is also applicable to any kind of wireless ad hoc network (such as wireless mesh networks), where a multicast routing strategy is needed.



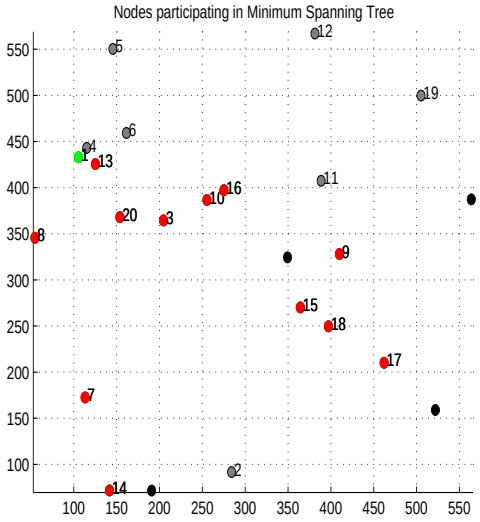
(a) Directed communication graph $G(V, E, c)$



(b) Complete distance graph $G_1(V_1, E_1, d_1)$

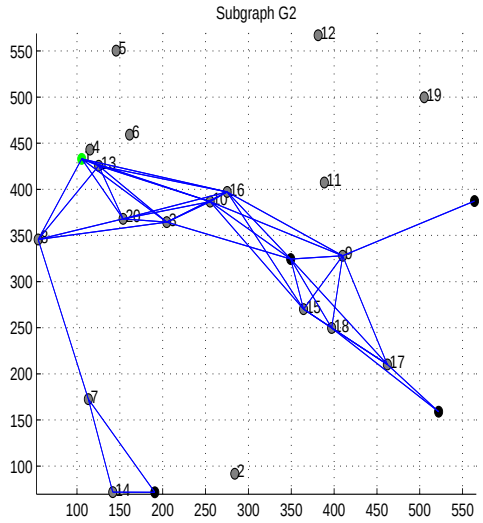


(c) Minimum Spanning Tree T_1 of G_1

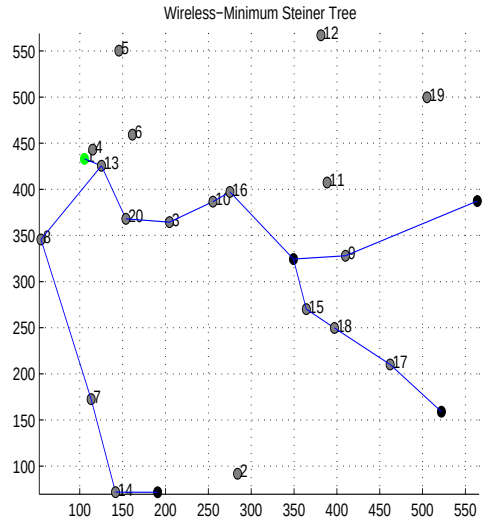


(d) Nodes participating in T_1

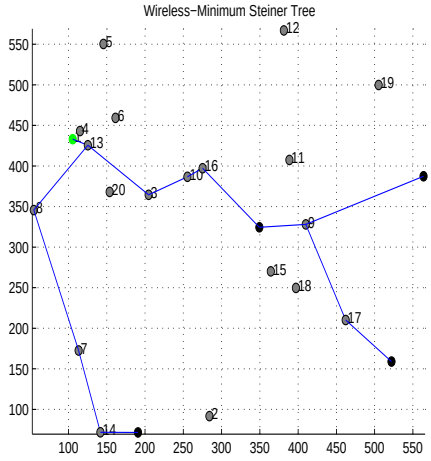
Figure 3.7: The Wireless Minimum Steiner Tree construction.



(a) Directed communication graph G_2



(b) W-MST before the *Sweep Operation*



(c) Improved W-MST after the sweep operation

Figure 3.8: The Wireless Minimum Steiner Tree construction.

Algorithm 5 The Wireless-Minimum Steiner Tree algorithm (W-MST)

Given $G(V, E, c)$, with $V = \{r, n_1, n_2, ..., n_k, d_1, d_2, ..., d_n\}$. The subset $S = \{r, d_1, d_2, ..., d_n\}$ denotes the multicast group, which consists of the source r and all the destinations $D = \{d_1, d_2, ..., d_n\}$.

1: Construct the complete distance graph $G_1 = (V_1, E_1, d_1)$ from G, where $V_1 = S$, and $d_1(u, v)$ is the total distance of the shortest path tree between u and v in G.
2: Find the minimum spanning tree T_1 of G_1.
3: Construct the subgraph G_2 with all nodes in T_1, and replacing each edge by its shortest path tree in G. Add also all edges due to broadcast communications.
4: Set $M = \{r\}$
5: Set $lastnode = \{r\}$
6: **while** $S \notin M$ **do**
7: Obtain *lastnode*'s distance to its closest neighbor from outside the tree ($u_{ln} \notin M$ and $u_{ln} \in G_2$). Compute $cost(lastnode, u_{ln}) = d(lastnode, u_{ln})^2$.
8: Calculate for all nodes $m_i \in M$, the minimum *incremental cost* for reaching another node $u_i \notin M$ (but $u_i \in G_2$).
9: Find node $u = \{u_{ln}, u_1, u_2, ..., u_{|M|}\}$, which involves less additional cost.
10: $u \rightarrow M$
11: $lastnode = \{u\}$
12: **end while**
13: Delete edges of leaf nodes that are not from the multicast group.
14: Run the *Sweep Operation*.

Algorithm 6 The Sweep Operation

Given a multicast tree $T = (V_T, E_T)$, with $V_T = \{r, n_1, n_2, ..., n_k, d_1, d_2, ..., d_n\}$. The subset $S = \{r, d_1, d_2, ..., d_n\}$ denotes the multicast group, which consists of the source r and all the destinations $D = \{d_1, d_2, ..., d_n\}$.

1: Assign to each node an index $n_i = n_1, n_2, ..., n_{|V_T|}$.
2: $T_{TEST} = T$
3: Obtain $Cost(T)$
4: **for** $node = n_1$ to $n_{|V_T|}$ **do**
5: List all nodes within *node*'s radio range.
6: Delete from the list the nodes that participate in the path from r to *node*.
7: In T_{TEST}, connect *node* to the remaining nodes in the list.
8: In T_{TEST}, disconnect from all the remaining nodes in the list those that are currently covering them in T.
9: Obtain $Cost(T_{TEST})$
10: **if** $Cost(T_{TEST}) < Cost(T)$ **then**
11: $T = T_{TEST}$
12: $Cost(T) = Cost(T_{TEST})$
13: **end if**
14: **end for**

3.3 Graph-based Lifting Transforms for Multisink WSN

In this section, we show briefly how the graph-based lifting transform described in Section 1.3 can also be applied in multisink WSN.

First of all, assume that we have a communication graph $G(V, E)$, where V consists of N sensors and M sinks, $V = \{n_1, n_2, ..., n_N, s_1, s_2, ..., s_M\}$. Let $T = (V, E_T)$ be a multicast routing tree along which data from every sensor flows towards the complete set of destinations. Assume that T is constructed as the union of all the multicast trees spanning each sensor with all the sinks. In order to build each tree, we can use any of the methods previously mentioned in this chapter. Note that the radio range of every node in G is chosen based on the distance of its most

distant parent in T.

Basically, the difference in this case with respect to the single sink scenario is how we define the routing strategy. Once we have the multicast routing tree and its corresponding communication graph defined, the graph-based lifting transform can be computed following the same method as in Section 1.3.

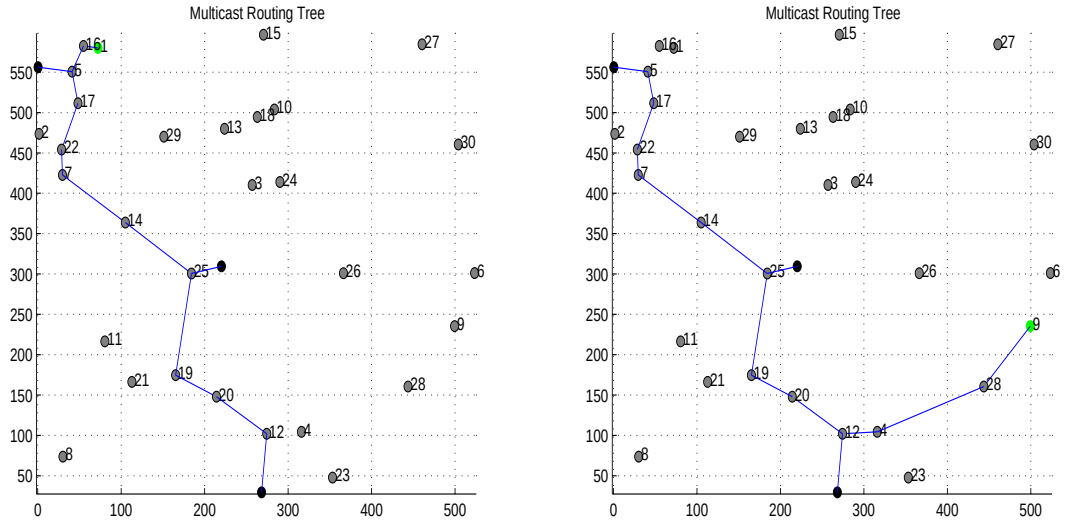
As we already know, for implementing the lifting scheme, it is necessary to perform first a Raw/Aggregating Node Assignment (RANA). Remember that how the RANA is implemented affects the energy efficiency of the complete compression scheme. In Chapter 2, we have shown that the most energy efficient RANA can be achieved by reducing the number of raw data nodes, while allowing some of these nodes to increase their radio range. Note that, if the RANA is solved as a minimum weighted set covering problem as in [25], the weight of each node now has to be proportional to the transmission cost of routing data from the node to the complete set of sinks.

After that, the processing strategy is independent of the number of sinks in the network. First, raw data nodes broadcast their data to all their aggregating neighbors. Then, the aggregating nodes compute the detail coefficients using the data that they have received. After the aggregating nodes have compressed their data, all nodes forward it following the multicast routing tree T until all data is gathered at the sinks. In this section, since nodes can have more than one parent in the tree, we assume that each node has a routing table specifying the corresponding parent in T to which data needs to be forwarded, depending on what sink data is being sent to. By doing so, we are considering that nodes have a variable radio range, and that it can be modified depending on to which of their parents are they transmitting.

For ensuring invertibility at all the sinks, as mentioned for the single sink case and as proved in [35], the aggregating nodes have to compute the detail coefficients using only data from their raw data neighbors. Moreover, in order to get the original data, either each aggregating node has to piggyback in its packet the indexes of the

raw data nodes from which it has used data to compress its measurements, or all the sinks need to have information about the topology and the RANA implemented in the network. With the fulfillment of these conditions, invertibility is ensured for any kind of network, independently of the number of sinks and RANA.

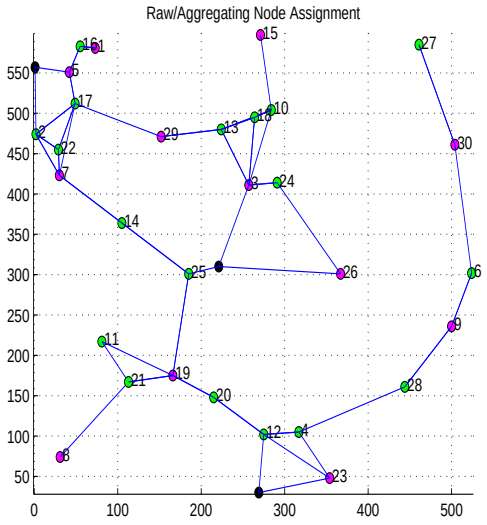
The procedure of the graph-based lifting transform in a multisink network is illustrated in Figures 3.9 and 3.10. Fig. 3.9(a) and Fig. 3.9(b) are examples of multicast trees for different nodes. As we have mentioned, the complete routing tree is computed as the union of the multicast trees for all the nodes. Once we have the routing strategy, we perform some sort of RANA in order to split the sensors into raw data or aggregating nodes. Fig. 3.10(a) shows the complete communication graph and an example of a RANA where the raw data nodes have been minimized. Fig. 3.10(b) illustrates the first step of the compression procedure. Raw data nodes broadcast their data to all their aggregating neighbors. Then, after the aggregating nodes have computed the detail coefficients, all nodes forward the data towards the destinations following the multicast tree shown in Fig. 3.10(c).



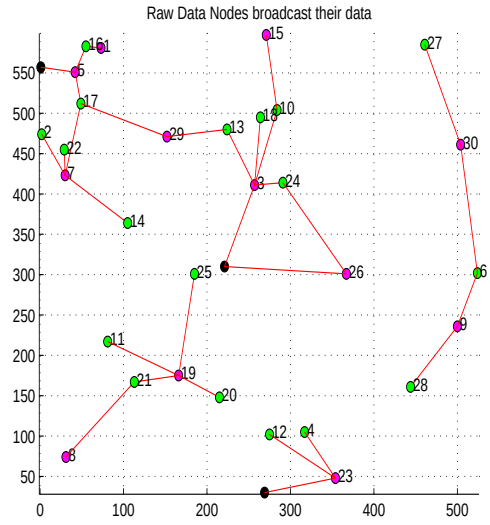
(a) Multicast tree for Node 1

(b) Multicast tree for Node 9

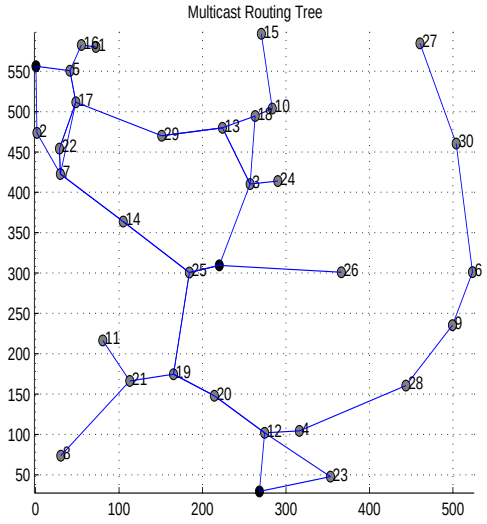
Figure 3.9: Graph-based lifting transform in multisink WSN.



(a) Communication Graph $G(V, E)$ and RANA



(b) Raw Data Nodes broadcast first their uncompressed data.



(c) Raw data and detail coefficients are forwarded towards the sinks.

Figure 3.10: Graph-based lifting transform in multisink WSN. In (a) and (b) raw data nodes are pink circles, and aggregating nodes are the green circles.

3.4 Experimental Results

In this section, we simulate and compare the performance of different algorithms for the construction of multicast trees in wireless networks. Focusing on an scenario where the data of all the nodes has to be gathered in more than one sink, we prove that our distributed transform allows nodes to compress their measurements, and as a result, to reduce considerably the energy consumption in the network. We compare the W-MST presented in this chapter to the union of SPTs, the link-based MST in [16], the pruned-based MIP in [46], and the MIP3S and MIP3Sb methods presented in [23]. Specifically, we compare these algorithms in terms of the energy consumption due to the communication costs needed to send all data to the sinks. We show that our proposed approach outperforms all the aforementioned methods.

3.4.1 Simulation Setting

We are going to simulate the performance of different routing algorithms in networks with 70 sensors and different number of sinks. In our simulations, we use the same experimental setup as in Section 2.3.1. However, in this case the data is forwarded towards the complete set of sinks following different kinds of multicast routing algorithms. We assume that the complete multicast tree is the union of all the multicast trees that connect each sensor with all the sinks. For the construction of each routing tree, we let the nodes transmit with a radio range within the interval $[R_u^{min}, R_{max}]$, where R_u^{min} is the minimum radio range that keeps node u connected to the network, and R_{max} can be an arbitrary value common for all nodes. In our experiments, we have considered R_{max} to be the maximum of all the R_u^{min} in the network. Once the tree has been built, the maximum radio range of each node is defined by the distance between that node and its most distant parent in the tree.

The performance of the different algorithms is evaluated in terms of communication costs. We perform the spatial compression using the graph-based lifting transform of Section 3.3. The RANA is formulated as a minimum weighted set

covering (MWSC) problem, and is solved using the centralized algorithm presented in [25]. The cost model, parameters of the filters, and transform coefficients are also defined in the same way as in Section 2.3.1.

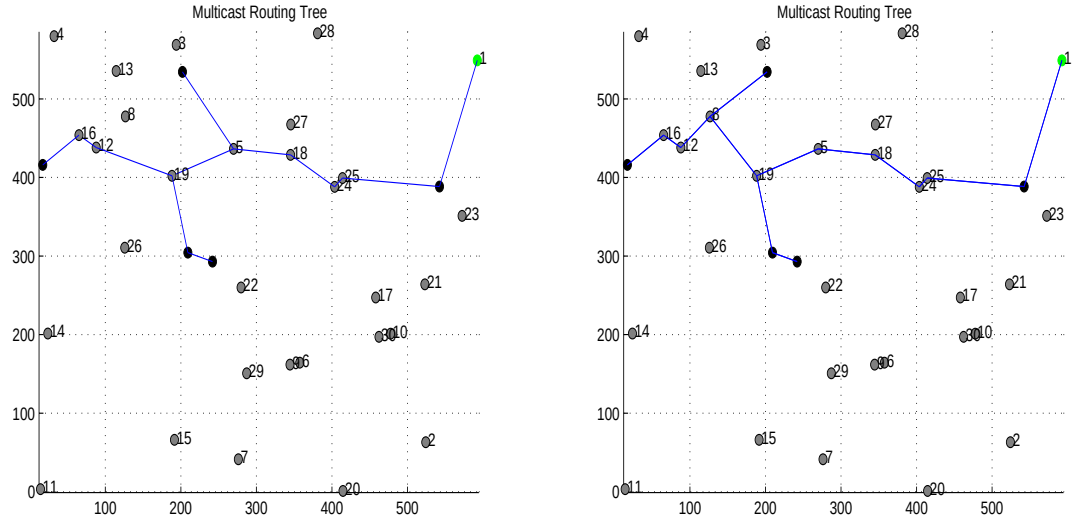
We first compare the energy consumption for gathering raw data in all the sinks using the different routing algorithms. Then, we evaluate the performance of the distributed lifting transform as the trade-off between the energy consumption and the reconstruction quality at each quantization level. The energy consumption is expressed in Joules, and the reconstruction quality in dB. Note that, fixing the cost, higher SNR implies higher fidelity in the reconstruction of the original data, and a difference of 1 dB in SNR translates to a decrease by a factor of 10 in MSE.

3.4.2 Performance Comparisons

We have evaluated the performance of six algorithms for the construction of multicast trees in wireless networks. Specifically, we compare our W-MST algorithm against two link-based methods, the Union of SPT and the MST as in [16], and also against three node-based approaches, the MIP algorithm presented in [46], and the MIP3Sb and MIP3S algorithms described in [23]. As mentioned before, to the best of our knowledge, in terms of energy efficiency, MIP3S and MIP3Sb are the best multicast routing algorithms for wireless ad hoc networks. All values presented in this section are an average of 30 iterations simulated under the same conditions.

Figures 3.11 and 3.12 show the corresponding multicast tree for the same node in a network consisting of 30 nodes and 5 sinks. As we can see in Fig. 3.12(a), MIP solution requires a longer path to span the entire multicast group, and therefore the overall routing cost will be higher in comparison to other alternatives. This is because MIP is based on pruning a broadcast tree, and a broadcast tree seeks to optimize the overall cost of spanning all nodes in the network (not only a specific subgroup). Thus, pruning it back is not always a good local solution. As expected, W-MST in Fig. 3.12(c) is similar to the link-based MST in Fig. 3.11(b). However,

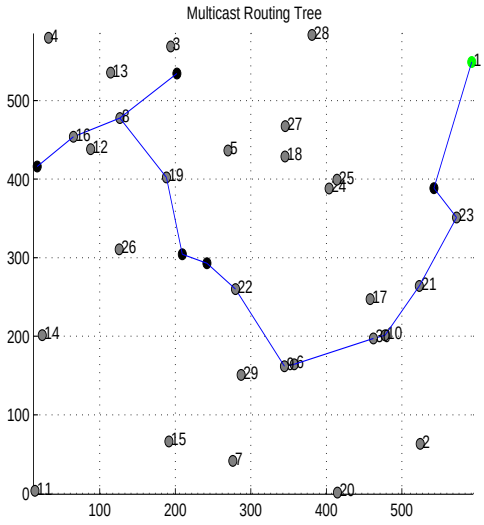
note that in our proposed method the broadcast property of wireless communications is properly exploited. As can be seen in Fig. 3.12(c), Node 5 covers 3 sinks with only one transmission, reducing in that way the amount of energy consumed in the network. Note that in this multicast tree the paths to route data from the source to all the sinks have more links in common than most of the other alternatives. Because MIP3S incorporates a "shortest path" from one node in the tree to an uncovered sink at each iteration, solutions 3.11(a) and 3.12(b) are quite similar in this example. However, with the sinks located further from each other the solution given by the union of SPT would be much worse compared to those using the other approaches, since it does not take into consideration neither the minimization of the overall routing cost of spanning the source with all the destinations nor the broadcast advantage of wireless communications.



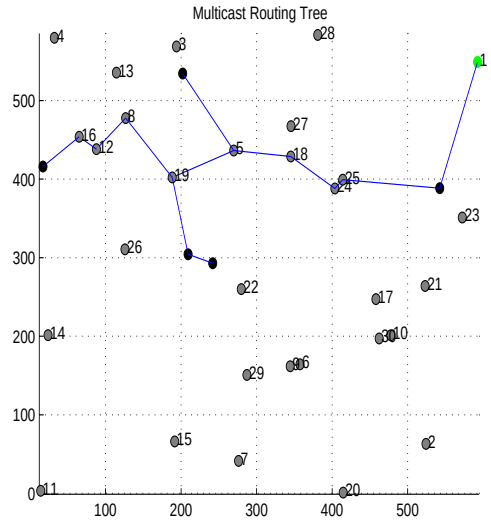
(a) Union of SPTs

(b) Minimum Steiner Tree

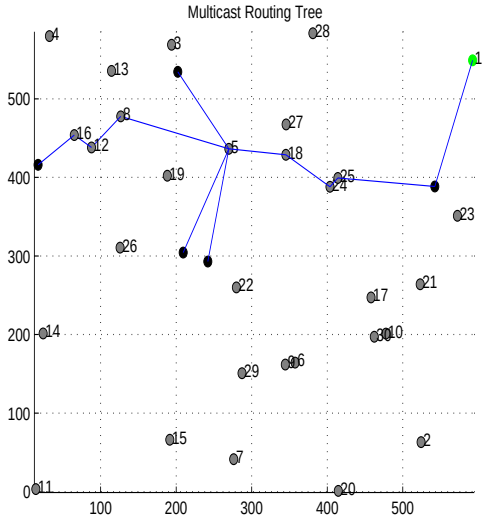
Figure 3.11: Examples of multicast trees using different algorithms.



(a) MIP



(b) MIP3S



(c) W-MST

Figure 3.12: Examples of multicast trees using different algorithms.

Figure 3.13 illustrates, for each algorithm, the communication cost for gathering raw data in a network with 70 sensors, and an increasing number of sinks. The horizontal axis represents the number of sinks, and the vertical axis is the overall energy consumed in the network due to communication costs. In our experiments,

W-MST outperforms all other algorithms for any size of the multicast group. As expected, MIP3S provides the second best performance. Note that MIP works better in networks with a large number of sinks. As mentioned earlier in this chapter, and in the related literature [46, 23], pruning a broadcast tree in order to obtain a multicast routing scheme might be a good solution for networks such that more than 65% of the nodes are part of the multicast group.

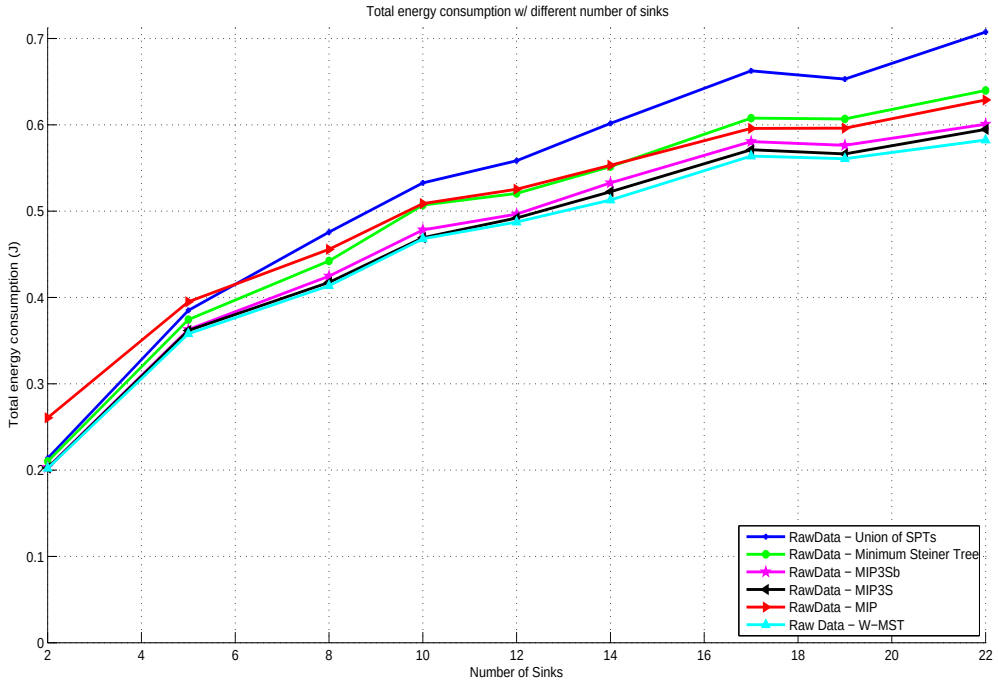


Figure 3.13: Energy consumption for raw data gathering using different routing algorithms and different number of sinks in networks with 70 sensors.

In Figure 3.14, we have added the energy consumption curves for gathering compressed data after implementing the graph-based wavelet transform presented in this chapter. Note that, since each tree entails a different communication graph, the RANA can be different in each case. This means that the difference in performance is not only related with the routing tree, but also with the number of raw data nodes

in the network. For example, it can be possible that a multicast tree with high routing costs provides a highly connected communication graph. In that case, after the splitting process, it can have lower number of raw data nodes in comparison with a better routing algorithm. Therefore, in this kind of situation, a better routing algorithm does not always imply a lower overall energy consumption in the network. However, note that the complete compression scheme using W-MST still outperforms all the other approaches for any size of the multicast group. Our experiments show that, in comparison with gathering raw data, we can reduce the energy consumption in the network up to a 40% by implementing the graph-based lifting transform. This reduction is independent of the number of sinks, or the routing algorithm used.

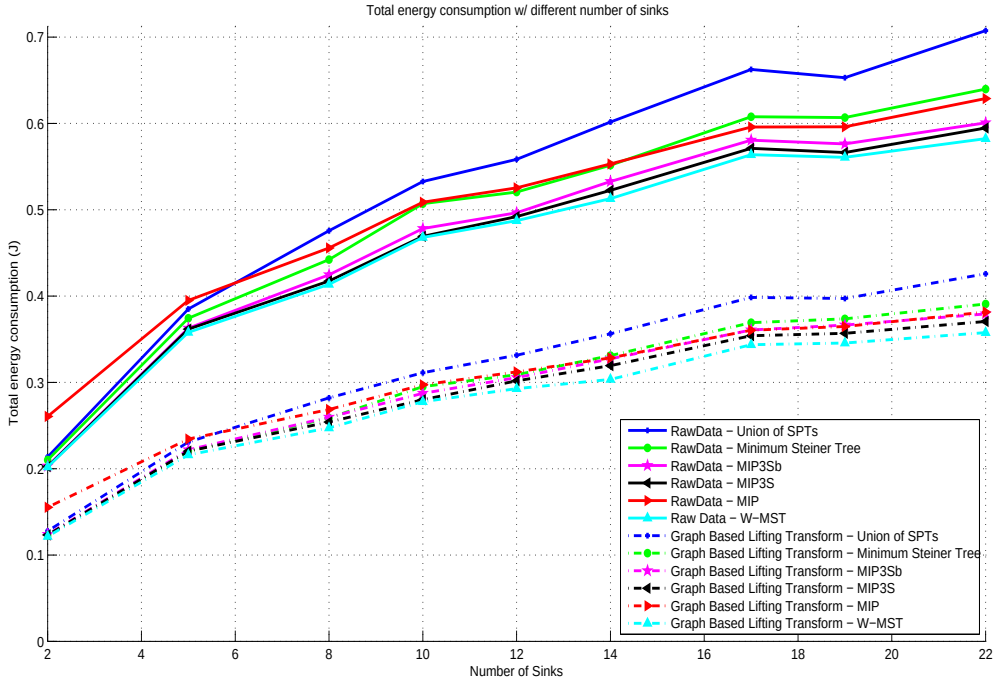


Figure 3.14: Energy consumption for raw and compressed data gathering using different routing algorithms and different number of sinks in networks with 70 sensors.

Finally, the cost versus distortion curves are shown in Figure 3.15 and Figure 3.16. As in the previous figures, W-MST outperforms all other routing algorithms for both raw and compressed data gathering, closely followed by MIP3S and MIP3Sb. MIP works better in networks with a large number of sinks, and the worst approach is the union of SPTs.

In conclusion, in this chapter, we have presented a new multicast routing algorithm that provides lower energy consumption than other proposed methods, and a distributed compression scheme for multisink data gathering applications that leads to considerable energy savings in comparison with transmitting raw data.

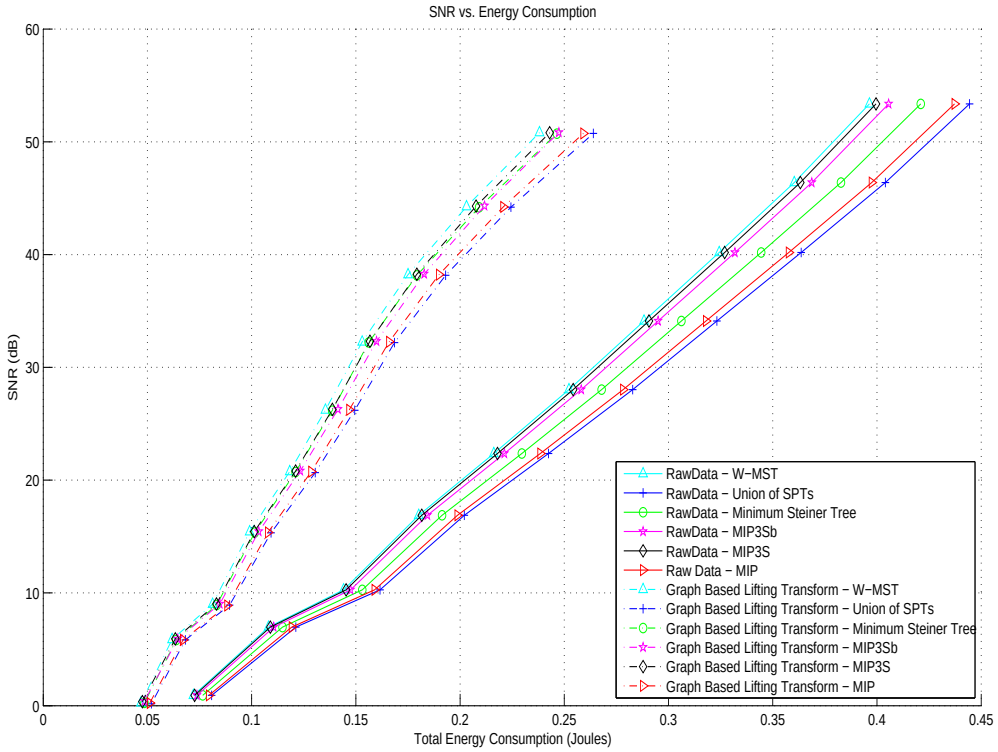


Figure 3.15: Cost-distortion curves for various routing algorithms in a network with 70 sensors and 7 sinks.

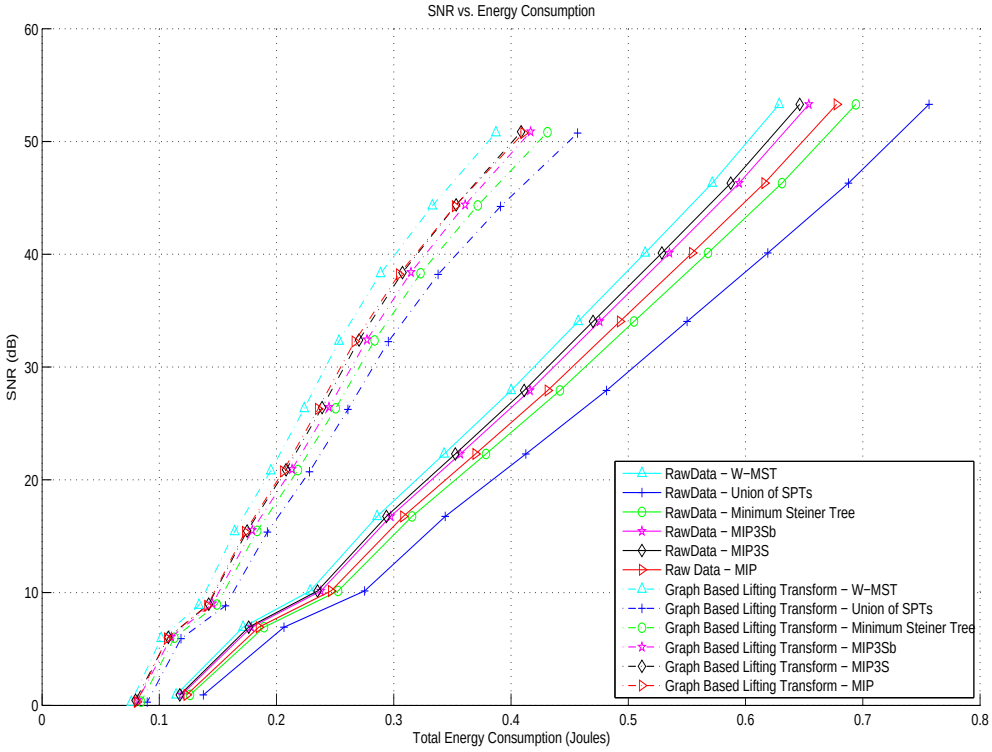


Figure 3.16: Cost-distortion curves for various routing algorithms in a network with 70 sensors and 25 sinks.

Chapter 4

Distributed Spatial Compression and Data Broadcasting in WSN

4.1 Introduction

The cooperative nature of wireless sensor networks (WSN), where nodes coordinate with each other in order to perform a certain task, makes the energy-efficient broadcast of data an important research topic for the development of many applications in which nodes need to disseminate some data all over the network. For example, broadcasting has traditionally been used as an efficient way to distribute control information for topology management purposes. In some scenarios, due to the instability of wireless communications, and considering that the sensors are working under severe conditions, the network topology becomes highly volatile and dynamic, with link and node failures becoming the norm instead of a rarity. In these cases, in order to broadcast data throughout the network, centralized routing algorithms, such as spanning trees or connected dominating sets, are impractical since they are based on the knowledge of the entire network topology at a central node or base station. Therefore, after any change or failure in the network, centralized algorithms have to be recomputed, involving additional high communication and computational costs. On the other hand, distributed algorithms do not require

any kind of global knowledge, and are robust to node failures and unreliable wireless network conditions.

Another constraint in WSN are the limited energy resources of the sensor devices, which are often supplied by weak batteries or small solar cells. Thus, energy-efficient broadcasting algorithms are also necessary to optimize the battery power, and to prolong the lifetime of the network in any kind of application or task.

Assuming an scenario in which all the nodes want to disseminate their measurements all over the network, and in order to tackle all the aforementioned requirements and limitations; our aim in this chapter is the development of a spatial compression and data dissemination framework, where the entire procedure is done in a distributed manner. In other words, we propose a decentralized scheme where sensors compress their measurements exploiting the spatial correlation existing between spatially close neighbors, and then the data is shared with all other nodes in the network using a decentralized routing algorithm.

Specifically, our contribution in this chapter is the implementation of the graph-based wavelet transform described in Section 1.3 when a deterministic routing strategy has not been defined, and data from all over the network has to be available at each node. In order to disseminate the measurements, we are going to focus on probabilistic broadcast methods known as *gossip algorithms*, which do not require any specialized routing or centralized coordination, and lead to networks working in a self-organized and autonomous manner.

The remainder of this chapter is organized as follows. Section 4.2 gives an overview of gossip algorithms and explains several existing methods. Our distributed framework for spatial compression and data broadcasting is introduced in Section 4.3. Section 4.4 presents simulation results of the proposed scheme.

4.2 Gossip Algorithms

Gossip algorithm are peer-to-peer communication protocols which are based, as their name suggests, on how a *rumor* is spread over a social network. Imagine that one node has a piece of data and wants it to be known by all the other nodes in a network. First of all, the node randomly selects a peer to communicate with and *gossips* its message. Once one node receives the message, it selects a new peer and keeps spreading the *rumor* over the network. The algorithm finishes when all nodes know the original data. Gossip algorithms are characterized by their distributed and random operation, and because they do not need a reliable network topology in order to disseminate the information.

Gossip algorithms have been widely studied recently for information processing and data dissemination in arbitrarily connected networks. Since they do not require any kind of centralized routing scheme, and are robust to node failures and topology changes, gossip algorithms are practical for WSN. However, there is a trade-off that has to be considered in the implementation of gossip algorithms, since they might require more time to converge, and more communications than centralized routing approaches.

At first, gossip algorithms were proposed as solutions for the *consensus problem* [15], in which all nodes in the network have to achieve a common opinion about the value of a certain parameter through local exchanges of data. One example of this problem is the computation of aggregated information, such as sums or averages. Imagine that we want each node to have an estimation of the average temperature in the area covered by the network. Each node n starts the algorithm fixing as the average its own value $x^n_{avg}(0) = x_n$. Then, using a simple gossip algorithm, such as the proposed in [4], nodes can start exchanging and updating their average values until the consensus is achieved. At each time t, a randomly chosen node n exchanges its value with a random neighbor m. After each iteration, all the participating nodes have the same updated average value,

$x^n_{avg}(t) = x^m_{avg}(t) = (x^n_{avg}(t-1) + x^m_{avg}(t-1))/2$. The algorithm finishes when all nodes arrive at a consensus value which is the average of the initial measurements from all over the network.

There are several different gossip algorithms proposed in the literature, and they all follow the same basic intuition. In gossip-based protocols, at each iteration, one node forwards packets to one or a few other random nodes with a certain probability. How this probability is specified classifies the different methods into static or adaptive algorithms. Pair-wise randomized gossip [4] is a common static approach in which a node chosen uniformly at random contacts a neighbor also chosen uniformly at random and exchange values with it. Broadcast gossip [3] follows the same idea, but exploiting the broadcast advantage of wireless communications. Thus, when one node transmits, all neighbors within its radio range receive the data. By doing so, the algorithm needs less number of communications to converge. Geographic gossip is proposed by Dimakis *et al.* in [8]. Assuming that each node knows its location in the network, they combine geographic routing with gossip-based broadcast in order to accelerate the diffusion of information among the nodes. The main idea is that nodes can communicate with other random nodes located anywhere in the network, instead of only exchanging data with nodes within their one hop neighborhood. They prove that the extra cost due to multi-hop routing is compensated with the reduction in number of communications needed to converge. Smart gossip [18] is an adaptive method, which focuses on information dissemination applications, the probability of each node transmitting is adapted in function of the local topological properties in its surroundings. For example, in areas with high density of sensors, forwarding packets probability will be lower. As a result, smart gossip can adapt in a distributed manner to random placements of sensors, and to topology changes or node failures. In the recent work [40], the greedy gossip with eavesdropping (GCE) algorithm is presented. In GCE, nodes keep track of their neighbors' values by exploiting the broadcast nature of wireless communications. Then, when one node is going to transmit, instead of choosing a neighbor uniformly at random, it chooses

the node which has the value most different from its own.

Gossip algorithms have been recently applied to solve several distributed problems in WSN [7]. Examples of applications performed using gossip algorithms are: distributed linear parameter estimation, compression and dissemination of information, distributed field estimation, and source localization. In the following section, we incorporate the gossip-based algorithms in our framework for distributed compression and data broadcasting in WSN. In our experiments, we propose a gossip algorithm based on the same idea as in [3], and adding *metadata* negotiation between nodes in order to reduce redundant transmissions.

4.3 Distributed Spatial Compression and Data Dissemination using Broadcast Gossiping

4.3.1 Problem statement and existing approaches

Imagine that we have a network consisting of several randomly deployed sensors, each of them having its own piece of information. Our objective in this section is to allow each node to gather all the information available in the network in a decentralized and energy-efficient manner. Considering the limited energy resources, and assuming that the measurements of spatially close sensors are correlated, we propose a decentralized approach for spatial compression and data dissemination based on the implementation of the distributed graph-based lifting transform described in Section 1.3, and distributing data using a gossip-based routing strategy.

As for the multisink case, there is a lack of proposed work in distributed compression for broadcast scenarios in WSN. One alternative to our method can be the use of distributed source coding (DSC) techniques, such as Slepian-Wolf coding [6]. The most similar work to ours is described in [29], where Rabbat *et al.* present a system for distributed compression and data dissemination via randomized gossiping. In [29], they use compressive sensing (CS) techniques to compress the sensors' measurements. CS is based on the notion that data is sparse (i.e., compressible) in

some sort of basis. However, finding a basis which leads to a sparse representation of the measurements is not always obvious. In [29], each node computes a projection of its data onto random vectors. Then, all these projections are distributed among the network using randomized gossip [4]. Assuming that the basis in which the sensor data is compressible is known; a user can query any node in the network and reconstruct the real values with small error. In contrast to the aforementioned methods, our distributed scheme does not need any kind of assumption about the sensors' data, and the measurements can be compressed and disseminated all over the network in a distributed manner.

Assume that the only information we know is that we have N randomly placed nodes, and each of them can transmit using a discrete number of radio ranges between some predefined minimum and maximum values. In order to design our decentralized framework we have to tackle three main different problems. First, nodes need to collaboratively determine their radio range in order to define the network topology. Second, the distributed graph-based wavelet transform has to be implemented to allow some nodes to compress their data exploiting the spatial correlation existing within their neighborhood. Finally, we need to define the gossip-based routing strategy that allows nodes to disseminate their measurements all over the network in a distributed manner. How to define the network topology is solved in Section 4.3.2, and Section 4.3.3 discusses how to distributively compress and disseminate the data.

4.3.2 Distributed Topology Control Algorithms

Once the nodes have been placed in the network, the first problem to tackle is how to choose their radio ranges among all the possible values in order to define an energy-aware network topology using only local information. This decision affects to the size of the neighborhood of each node, and therefore it defines the set of nodes that each sensor can reach in direct transmission. Depending on the distribution

of the nodes in the network, sometimes it is more efficient to communicate with some distant sensors using other closer neighbors as forwarding nodes instead of reaching them in a single transmission using a higher radio range, which may also incur higher overall communication costs. Thus, first of all, we need to create a communication graph $G = (V, E)$ by allowing nodes to collaboratively determine their energy-efficient radio ranges.

One straightforward solution to this problem is to let each node transmit at maximum power. However, this is not an energy-efficient strategy since it does not take into consideration the possibility of using other nodes as relay stations, and nodes would run out of battery in a very short period of time. Therefore, the main objective of a topology control algorithm is to avoid long-distance links, and instead propose a communication graph in which nodes can route their data using energy-efficient multi-hop paths. Other simple approaches are the minimum spanning tree (MST) and the Delaunay triangulation [11]. Although with MST the overall link cost in the network is minimized, spatially close neighbors can in the end be far from each other. On the other hand, the Delaunay triangulation can't be computed locally and therefore is not a practical solution for WSN. There are plenty of proposed topology control algorithms in the literature. However, many of these solutions often require unrealistic assumptions, such as considering nodes to know their exact position in the network by using GPS information [20]. Other approaches, for example, only consider unit disk graph (UDG) [11], what means that all nodes have the same radio range. In [44], it is assumed that nodes can estimate the direction from which another node is transmitting by using more than one directional antennas. For our distributed framework, we are going to use the XTC algorithm proposed in [45]. It is, to the best of our knowledge, the simplest decentralized algorithm for topology control, and it is based on realistic assumptions. In XTC, nodes only have to communicate two times with their direct neighbors, and make decisions about which links to keep in the resulting communication graph based only on information related to the link quality (such as distance, signal strength or packet rate).

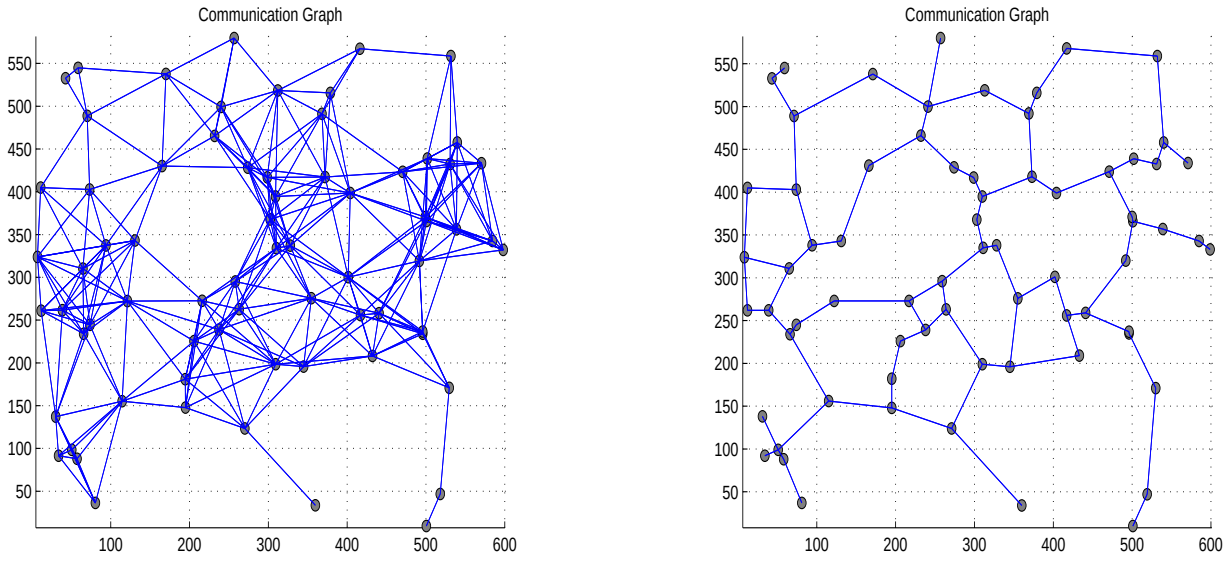
XTC consists of three main steps. First of all, let $G_{max}(V, E_{max})$ be the communication graph with all nodes transmitting at their maximum radio range. At the beginning, nodes acquire information about the link quality between them and their 1-hop neighbors, and compute a total order reflecting which candidates are the best to communicate with. It can be done in a simple way by allowing nodes to transmit/receive beacon signals and evaluate their signal strength. In our scheme, in order to make this algorithm energy-efficient, we are going to consider the link quality concept as the Euclidean distance between nodes. Each node n, at the end of the first step, has a total order $\prec_n$ with its neighbors ordered with respect to decreasing link quality. The second step consists in each node sharing its order information with all its neighbors. Then, once each node knows its own order information and those from its neighbors, it decides which nodes will be part of its neighborhood based only on the link quality information. In the third step, node n starts looking to $\prec_n$ in decreasing order (best candidates go first). Imagine that at a certain time, node n is considering wether to add a node m to its neighborhood. The criteria is that if there is not any already evaluated node q that appears before n in node m's order, then node m will be included in node n's neighborhood. The procedure of the XTC Algorithm presented in [45] is detailed in Algorithm 7 from the point of view of node n. Two points are worth mentioning about the XTC algorithm. First, note that the resulting communication graph is symmetric, since the conditions between nodes n and m are the same from the point of view of both nodes. Second, observe that the entire method is completely local, and nodes only need to communicate with their direct neighbors.

Algorithm 7 XTC Algorithm

Assume $G_{max}(V, E_{max})$ to be the communication graph when nodes transmit at their maximum radio range.

1: Transmit a pilot signal at maximum radio range.
2: Receive pilot signals from direct neighbors N_n.
3: Compute order $\prec_n$ over all the neighbors in G_{max}.
4: Broadcast $\prec_n$ at maximum radio range.
5: Recieve order information from all neighbors.
6: Set $N_n^{in} = \{\}$.
7: Set $N_n^{out} = \{\}$.
8: **while** there are unprocessed neighbors in $\prec_n$ **do**
9: m = least unprocessed neighbor in $\prec_n$.
10: **if** $(\exists q \in N_n^{in} \cup N_n^{out} : q \prec_m n)$ **then**
11: $N_n^{out} = N_n^{out} \cup m$.
12: **else**
13: $N_n^{in} = N_n^{in} \cup m$.
14: **end if**
15: **end while**

Figure 4.1 illustrates for a given $G_{max}(V, E_{max})$ (Fig. 4.1(a)) its resulting communication graph using XTC method(Fig. 4.1(b)). Note that long distance links have been replaced by energy-efficient multi-hop paths using shorter links. In the following sections of this chapter, we will refer to the communication graph after XTC as $G(V, E)$. Once nodes know which are their neighbors and their corresponding radio ranges, we can implement the distributed graph-based lifting transform, allow nodes to compress their data, and finally distribute both compressed and uncompressed information all over the network.



(a) $G_{max}(V, E_{max})$ (b) $G_{XTC}(V, E_{XTC})$

Figure 4.1: XTC Algorithm. Fig. 4.1(a) reflects the original communication graph with nodes transmitting at maximum radio range. The resulting graph after implementing the toplogy control algorithm XTC is shown in Fig.4.1(b).

4.3.3 Spatial Compression and Data Broadcasting

Once the communication graph $G(V, E)$ is correctly defined, the main problem to solve is how to implement the graph-based lifting transform detailed in Section 1.3 without having a centralized routing strategy defined. Then, we can use a gossip-based routing algorithm to allow nodes to broadcast the compressed and uncompressed data in a distributed manner.

First of all, in order to compute the aforementioned transform, we have to split the nodes into raw/aggregating sets by means of implementing an energy-efficient RANA. Since we seek the entire system to work in a decentralized manner, we are going to use our distributed heuristic algorithm presented in Chapter 2 (Algorithm 2). Then, as we have proved for the multisink case in Section 3.3, we can implement the simplified graph-based lifting transform with any kind of routing strategy. The only requirement is that we first need a coordinated step in which

raw data nodes broadcast their measurements to all their aggregating neighbors. Then, aggregating nodes can compute the detail coefficients with data received from raw data nodes. In the lifting scheme, invertibility is ensured by construction if aggregating nodes only use data from their raw data neighbors. As we have mentioned for multisink WSN, in order to obtain real data from the detail coefficients all nodes will need to know with which raw data nodes each aggregating node has compressed its measurements. This can be done with low overhead by allowing aggregating nodes to piggyback the indexes of the raw data nodes used for compression in their respective packets.

Finally, once the coordinated step has finished, we want to make all data in the network available at each node without the need of a centralized routing algorithm. In order to do that, we can use any gossip-based approach mentioned in Section 4.2. However, for the sake of simplicity, we propose a simple gossip algorithm based on the broadcast gossip method in [3], with the difference that we use *metadata* negotiation in order to reduce the number of redundant transmissions. In other words, we want to disseminate all data using the same intuition as in [3], which exploits the broadcast property of wireless communications to reduce the number of iterations that the algorithm needs to converge, with the difference that we do not allow nodes to receive the same data more than once.

There are several design factors that have to be defined before implementing a gossip-based algorithm. First, there are two time models for scheduling transmissions. In the *synchronous* time model, time is assumed to be slotted commonly across nodes. In each slot, all nodes contact a neighbor to communicate with. Note that in this time model all nodes communicate simultaneously. On the other hand, in the *asynchronous* time model each node has its own Poisson clock. Thus, when a node's clock *ticks*, it is the only one that chooses a random node to transmit its data. Since only one node transmits at each time slot, the number of packet collisions is reduced in comparison with the *synchronous* model. However, more

iterations may be needed in order to complete the data broadcasting task. We use the *asynchronous* time model since it is the one which adapts better to the decentralized nature of WSN. The second design factor is the data exchange protocol. In the push protocol, one node chooses another node to transmit its data to. The oppossite alternative is the pull protocol, in which the node requests the other node's data instead of transmitting its data to the selected neighbor. The third approach is when both nodes exchange their data during the same time slot. In our algorithm, we are going to use the push protocol as in most of the existing gossip algorithms. Finally, the gossip communication mechanism can be uniform or probabilistic. In *uniform/static* gossip, nodes choose their communication partners at random. On the contrary, in *probabilistic/adaptive* gossip, nodes can adapt the probability to communicate with each of their neighbors in function of the network topology or any other critical factor in the network. Note that, in this chapter, we just want to illustrate the possibility to extend the graph-based lifting transform to a distributed framework for *all-to-all* data transmission in WSN. Thus, we are going to implement a uniform mechanism as in [4, 3].

Assume that each node has a Poisson clock that runs independently. Moreover, in order to keep track of the received data, each node has a vector Av_{data} in which it memorizes the indexes of the nodes from which it has data already available. At the beginning of the algorithm, nodes know their complete set of neighbors and the different necessary radio ranges to reach each of them. When node n's clock *ticks*, n selects, uniformly at random, one neighbor m to communicate with. First of all, the participating nodes negotiate which data has to be sent. Node n sends its vector $Av_{data}(n)$ data to m. Then, node m compares its vector $Av_{data}(m)$ with the received one, and sends to node n the indexes of nodes in $Av_{data}(n)$ that it lacks to receive data from. Finally, node n transmits to m only data from the requested nodes. As in [3], nodes within the radio range of n can also receive the data that n is sending, reducing in that way the necessary iterations for the algorithm to complete its task.

The proposed gossip-based algorithm is detailed, from the point of view of a single node, in Algorithm 8.

Algorithm 9 summarizes the distributed scheme for spatial compression and data broadcasting in WSN. Assume that we have N randomly deployed nodes, and let each of them have minimum and maximum possible radio ranges defined by construction. First of all, we run the distributed topology control algorithm XTC (Algorithm 7), which defines the communication graph $G(V, E)$. Then, after nodes have identified their neighbors, we run the distributed heuristic for the RANA implementation (Algorithm 2). After that, sensors know if they are raw data or aggregating nodes. The implementation of the graph-based lifting transform needs a coordinated step, in which raw data nodes broadcast their data to their aggregating neighbors. Once they have done that, aggregating nodes can compress their data and compute the detail coefficients. Then, nodes can broadcast both compressed and uncompressed data using any gossip-based routing algorithm, such as Algorithm 8, until all data is available at each node.

In conclusion, in this chapter we have presented a framework for data compression and dissemination, which is computed in a distributed manner, with nodes only knowing information about their direct neighborhood. Thus, this distributed scheme is adapted to the decentralized nature of WSN, it does not incur bottlenecks, and it is robust to topology changes and failures.

Algorithm 8 Broadcast-Gossip Algorithm with Metadata Negotiation

Set of neighbors, $\mathcal{N}_n = [m_1, m_2, ..., m_M]$.

Set of radio ranges, $R_n = \left[R_n^1, R_n^2, ..., R_n^M\right]$.

$Av_{data}(n) = [n]$

$Req_{data}(n) = []$

while There are still nodes without all data available **do**
  Run Poisson clock $t(n)$.
  while $t(n) > 0$ **do**
    Listen to data transmissions in the neighborhood.
    Update Av_{data} if necessary.
    if Node q wants to gossip data **then**
      Receive $Av_{data}(q)$.
      $Req_{data}(n) = Av_{data}(q) - (Av_{data}(q) \cap Av_{data}(n))$.
      Send $Req_{data}(n)$.
      Receive data from nodes $\in Req_{data}(n)$.
      Update $Av_{data}(n) = Av_{data}(n) \cup Req_{data}(n)$.
    end if
  end while
  Choose one neighbor m at random.
  Transmit $Av_{data}(n)$ with radio range R_n^m.
  Listen to $Req_{data}(m)$.
  Send data from nodes with indexes $\in Req_{data}(m)$.
end while

Algorithm 9 Distributed Scheme for Data Compression and Broadcasting in WSN

Given N randomly placed sensors,

with possible radio ranges within the interval $R_n = [R_{min}, R_{max}]$.

1: Run XTC Algorithm for topology control purposes (Algorithm 7).

2: Obtain $G(V, E)$.

3: Run Distributed Heuristic for Minimum Set Cover (Algorithm 2).

4: Obtain the Raw Data/Aggregating nodes assignment.

5: Raw Data nodes broadcast their data.

6: Aggregating nodes compute detail coefficients.

7: Run Gossip Algorithm for data broadcasting (Algorithm 8).

4.4 Simulation Results

In this section, we prove the usefulness of our proposed distributed scheme by comparing it with raw data broadcasting using the same gossip-based algorithm. As done before in this thesis, we evaluate the performance of both approaches as the trade-off between the energy consumption and the reconstruction quality at each quantization level. The energy consumed is expressed in Joules, and the reconstruction quality in dB. The results shown in this section are an average of 100 iterations of the respective methods under the same conditions. The simulation setup used for the cost model and the transform paramaters is the same as in Section 2.3.1.

Figure 4.2 shows the cost versus distortion curves of both raw and compressed data broadcasting approaches. As can be seen, our proposed scheme reduces considerably the energy consumption in the network, with savings up to a 40% in comparison with disseminating raw data.

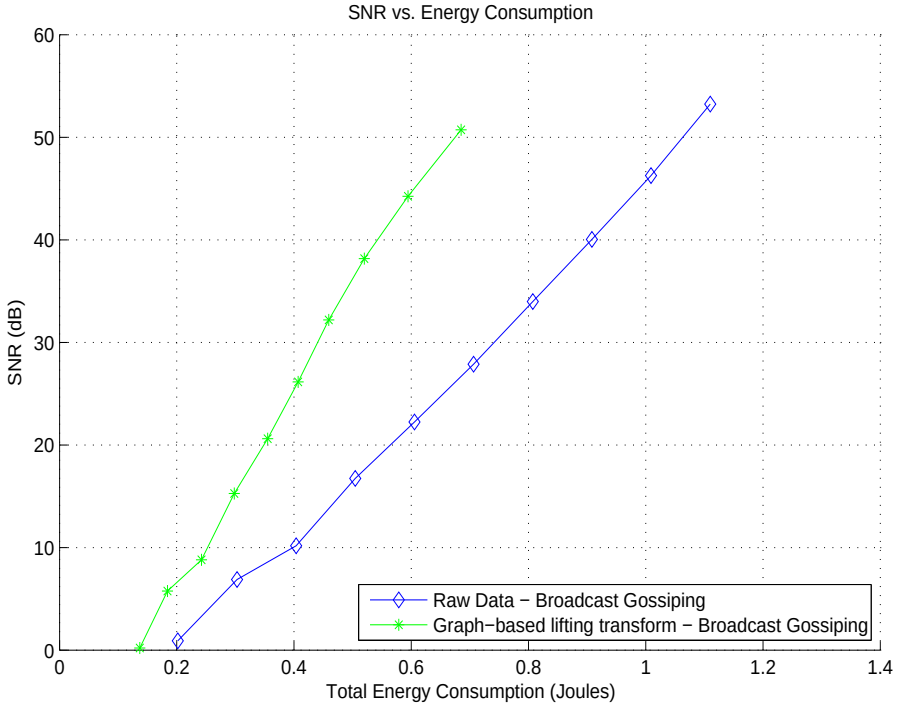


Figure 4.2: Cost-distortion curves for compressed and raw data broadcasting in WSN with 70 nodes.

In conclusion, we have presented a decentralized scheme that, starting with N random deployed nodes, exploits the spatial data correlation, and distributes all the information in the network among the complete set of nodes, using only local communications. As shown in our experiments, in contrast to the other methods, our solution reduces the energy consumed in the network without the need of any kind of assumption about the sensors' data.

Chapter 5

Conclusions and Future Work

In this thesis we have presented several contributions to the field of spatial compression in Wireless Sensor Networks (WSN) for distributed data gathering applications. Specifically, our contributions have been mainly focused on extending previous work related to the use of a graph-based lifting transform to different scenarios in WSN.

First of all, we have introduced a new distributed heuristic for solving the Raw/Aggregating Node Assignment (RANA) as a Minimum Set Covering Problem. The RANA is a mandatory requirement for the implementation of the lifting scheme in WSNs, and how it is solved has been proved to be a critical factor for the energy efficient performance of the transform. Since centralized algorithms are not well suited to this kind of networks, mainly because they are not robust to topology changes and node failures, our proposed method allow nodes to choose between being a raw data or aggregating node using only information gathered from their 1-hop neighborhood. In order to compensate the suboptimality of our low complexity heuristic, we have also developed several distributed set cover modifications, which iteratively improve the first assignment by allowing some nodes to increase their radio range and/or switch their parity. We can emphasize that our complete distributed solution outperforms other centralized methods in reducing the energy consumption in the network, and it can be used as a practical alternative in a real setting due to its simplicity and decentralized performance. The

proposed distributed algorithms in this part of the thesis have been included in [26].

Our second contribution has been the extension of the graph-based lifting transform to a distributed data gathering scenario in WSN with more than one sink (multisink). Under the scope of this contribution, we have also proposed a new multicast routing algorithm for wireless ad hoc networks, which is based on the minimum Steiner tree problem, and which considers the broadcast advantage of wireless communications. To the best of our knowledge, our method is the first routing-driven compression scheme for multisink WSN which, in contrast to all other approaches, exploits the spatial correlation among neighboring nodes while data is being forwarded to the sinks. We have proved that our proposed routing solution outperforms most of the existing multicast routing algorithms, and that our compression scheme provides energy savings up to a 40% in comparison with raw data gathering.

Finally, as a natural extension to the distributed data gathering in networks with more than one sink, we have proposed a decentralized framework for distributed compression and data broadcasting for applications in which all data has to be available at each node. In our scheme, after using the same procedure as in the multisink case in order to perform compression, nodes broadcast all data following a gossip-based routing algorithm. Moreover, based on the existing Broadcast Gossip, we have designed a new gossip algorithm which reduces redundant transmissions via metadata negotiations. To the best of our knowledge, our method is the first work that combines gossip routing with spatial compression in WSN. It is worth mentioning that the complete procedure of our scheme is performed in a distributed manner. Thus, it is a suitable practical solution for multicast and broadcast applications in this kind of networks. Our simulations show that, in comparison with raw data gathering, our solution reduces considerably the energy consumption in the network.

As future work, for the first of our contributions, we should refine our distributed heuristics to improve their energy efficiency, and provide a deeper analysis of their

performance. Moreover, following the research trend in WSNs, our algorithms should also be modified in order to reflect the effects of node mobility. In the future, the transform for the multicast and broadcast scenarios can be extended to allow the development of more levels of decomposition, improving in that way the data decorrelation. Finally, we can consider the implementation of the proposed distributed compression schemes not only for applications in WSNs, but also in areas such as social networks or image and video compression.

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] I. F. Akyildiz and I. H. Kasimoglu. Wireless sensor and actor networks: research challenges. *Ad Hoc Networks Elsevier 2*, pages 351–367, 2004.

[3] T. Aysal, M. Yildiz, and A. Scaglione. Broadcast gossip algorithms. In *Proc. IEEE Information Theory Workshop*, May 2008.

[4] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Gossip algorithms: Design, analysis and applications. In *Proc. IEEE Infocom*, 2005.

[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*.

[6] R. Cristescu, B. Beferull-Lozano, and M. Vetterli. Networked Slepian-Wolf: Theory, algorithms, and scaling laws. *IEEE Transactions on Information Theory*, 51(12):4057–4073, December 2005.

[7] A. G. Dimakis, S. Kar, J. M. F. Moura, M. G. Rabbat, and A. Scaglione. Gossip algorithms for distributed signal processing. In *Submission to the IEEE Proceedings*, March 2010.

[8] A. G. Dimakis, A. D. Sarvate, and M. J. Wainwright. Geographic gossip: Efficient aggregation for sensor networks. In *Proceedings of the Fifth International Conference on Information Processing in Sensor Networks (IPSN)*, 2006.

[9] M. Gastpar, P. Dragotti, and M. Vetterli. The distributed Karhunen-Loève transform. *IEEE Transactions on Information Theory*, 52(12):5177–5196, December 2006.

[10] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient routing protocols for wireless microsensor networks. In *Proc. of Hawaii Intl. Conf. on Sys. Sciences*, January 2000.

[11] J. Hou, N. Li, and I. Stojmenovic. *Handbook of Sensor Networks: Algorithms and Architectures*, chapter 10, pages 311 – 341.

[12] L. Jia, R. Rajaraman, and R. Suel. An efficient distributed algorithm for constructing small dominating sets. In *Proceedings of the 20th Annual ACM Symposium on Principles of Distributed Computing*, pages pp 33–42, August 2001.

[13] R. Kalapala, M. Pelikan, and A. Hartmann. Hybrid evolutionary algorithms on minimum vertex cover for random graphs. Technical Report MEDAL Report No.2007004, University of Missouri-St.Louis, 2007.

[14] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Plenum Press, 1975.

[15] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *Proc. of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS'03)*, pages 482–491, 2003.

[16] L. Kou, G. Markowsky, and L. Berman. A fast algorithm for Steiner trees. *Acta Informatica 2*, 15:141–145, 1981.

[17] F. Kuhn, T. Moscibroda, and R. Wattenhofer. The price of being near-sighted. In *In SODA 06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 980–989. ACM, 2006.

[18] P. Kyasanur, R. Choudhury, and I. Gupta. Smart gossip: An adaptative gossip-based broadcasting service for sensor networks. In *Mobile Ad Hoc and Sensor Systems (MASS'06)*, pages 91–100, 2006.

[19] D. Li, K. D. Wong, Y. H. Hu, and A. M. Sayeed. Detection, classification and tracking of targets in distributed sensor networks. *IEEE Signal Processing Magazine*, 19(2):17–29, March 2002.

[20] N. Li and J. Hou. Topology control in heterogeneous wireless networks: Problems and solutions. In *Proc. of the 23rd IEEE INFOCOM*, 2004.

[21] D. S. Lun, E. Ahmed, L. R. Brothers, and J. A. DeBardelaben. Network coding for wireless applications. In *In Proc. of SPIE*, pages 258–260, 2005.

[22] D. S. Lun, N.Ratnakar, and R.Koetter. Achieving minimum-cost multicast: A decentralized approach based on network coding. In *In Proc. of IEEE INFOCOM*, 2005.

[23] P. M. Mavinkurve, H. Q. Ngo, and H.Mehra. Mip3s: Algorithms for power-conserving multicasting in static wireless ad hoc networks. In *Proc. of the 11th IEEE International Conference on Networks (ICON'03)*, 2003.

[24] S. Narang and A. Ortega. Lifting based wavelet transforms on graphs. In *Proc. of APSIPA Annual Summit and Conference (APSIPA ASC'09)*, 2009.

[25] S. Narang, G. Shen, and A. Ortega. Unidirectional graph-based wavelet transforms for efficient data gathering in sensor networks. In *In Proc. of ICASSP'10*.

[26] S. Narang, G. Shen, J. Perez-Trufero, and A.Ortega. Encodes: Energy-aware compression optimization using distributed energy-efficient set covers. Technical report, University of Southern California, April 2010.

[27] U. Nguyen. On multicast routing in wireless mesh networks. *Elsevier Journal of Computer Communications*, 31:1385–1399, 2008.

[28] S. Pattem, B. Krishnamachari, and R. Govindan. The impact of spatial correlation on routing with compression in wireless sensor networks. *ACM Transactions on Sensor Networks*, 4(4):60–66, August 2008.

[29] M. Rabbat, J. Haupt, A. Singh, and R. Nowak. Decentralized compression and predistribution via randomized gossiping. In *Proc. Information Processing in Sensor Networks*, April 2006.

[30] M. Rabbat and R. Nowak. Distributed optimization in sensor networks. In *IPSN'04*.

[31] G. Robins and A. Zelikovsky. Improved steiner tree approximation in graphs. In *In Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 770–779, 2000.

[32] P. M. Ruiz and A. F. Gomez-Skarmeta. Approximating optimal multicast trees in wireless multihop networks. In *10th IEEE Symposium on Computers and Communications*, pages 686–691, June 2005.

[33] K. Sayood. *Introduction to Data Compression*. Academic Press, 2nd edition, 2000.

[34] G. Shen, S. K. Narang, and A. Ortega. Adaptive distributed transforms for irregularly sampled wireless sensor networks. *ICASSP '09*, 0:2225–2228, 2009.

[35] G. Shen and A. Ortega. Transform-based distributed data gathering. *To Appear in IEEE Transactions on Signal Processing*. arXiv:0909.5177.

[36] G. Shen and A. Ortega. Optimized distributed 2D transforms for irregularly sampled sensor network grids using wavelet lifting. In *Proc. of ICASSP'08*, April 2008.

[37] K. Sohrabi, J. Gao, V. Ailawadhi, and G. J. Pottie. Protocols for self-organization of a wireless sensor network. *IEEE Personal Comm.*, 7(5), Oct. 2000.

[38] A. Sridharan and B. Krishnamachari. Max-min fair collision-free scheduling for wireless sensor networks. In *Proc. of IEEE IPCCC Workshop on Multi-hop Wireless Networks*, April 2004.

[39] W. Sweldens. The lifting scheme: A construction of second generation wavelets. Tech. 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.

[40] D. Ustebay, B. Oreshkin, M. Coates, and M. Rabbat. Rates of convergence for greedy gossip with eavesdropping. In *Proc. Allerton Conf. on Commnication, Control, and Computing*, 2008.

[41] 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*.

[42] 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.

[43] A. Wang and A. Chandraksan. Energy-efficient DSPs for wireless sensor networks. *IEEE Signal Processing Magazine*, 19(4):68–78, July 2002.

[44] R. Wattenhofer, L. Li, P. Bahl, and Y. Wang. Distributed topology control for power efficient operation in multihop wireless ad hoc networks. In *Proc of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2001.

[45] R. Wattenhofer and A. Zollinger. Xtc: A practical topology control algorithm for ad-hoc networks. In *Proceedings of the 4th International Workshop on*

Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN'04), 2004.

[46] J. E. Wieselthier, G. D. Nguyen, and A. Ephremides. Energy-efficient broadcast and multicast trees in wireless networks. *Mobile Networks and Applications*, 7:481–492, 2002.