Network Correlated Data Gathering with Explicit Communication: NP-Completeness and Algorithms*

Răzvan Cristescu[1], Baltasar Beferull-Lozano[1], Martin Vetterli[1,2], and Roger Wattenhofer[3]

[1]Laboratory for Audio-Visual Communications (LCAV),
Swiss Federal Institute of Technology (EPFL), Lausanne CH-1015, Switzerland
[2]Department of Electrical Engineering and Computer Science,
University of California at Berkeley, Berkeley CA 94720, USA
[3]Distributed Computing Group,
Department of Computer Science, ETH Zurich, Zurich CH-8092, Switzerland
URL: http://www.mics.org/
{Razvan.Cristescu, Baltasar.Beferull, Martin.Vetterli}@epfl.ch, wattenhofer@inf.ethz.ch

March 16, 2004

Abstract

We consider the problem of correlated data gathering by a network with a sink node and a tree based communication structure, where the goal is to minimize the total transmission cost of transporting the information collected by the nodes, to the sink node. For source coding of correlated data, we consider a joint entropy based coding model with explicit communication where coding is simple and the transmission structure optimization is difficult. We first formulate the optimization problem definition in the general case and then we study further a network setting where the entropy conditioning at nodes does not depend on the amount of side information, but only on its availability. We prove that even in this simple case, the optimization problem is NP-hard. We propose some efficient, scalable, and distributed heuristic approximation algorithms for solving this problem and show by numerical simulations that the total transmission cost can be significantly improved over direct transmission or the shortest path tree. We also present an approximation algorithm that provides a tree transmission structure with total cost within a constant factor from the optimal.

*The work presented in this paper was supported (in part) by the National Competence Center in Research on Mobile Information and Communications Systems (NCCR-MICS, http://www.mics.org), a center supported by the Swiss National Science Foundation under grant number 5005-67322. Parts of this work have been presented at the 23rd Conference of the IEEE Communications Society (INFOCOM 2004), and at the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2003).

1 Introduction

1.1 Correlated Data Gathering

Consider a number of distributed data sources with a certain correlation structure and which are located at the nodes of a network. A practical example of such a situation is the case of sensor networks that measure environmental data [1, 18, 20]. Collecting images from various sources into a common repository on the internet is another example of correlated data gathering. A number of links connect sources to each other, establishing a graph where sources are nodes and links are edges. The task is to send all the data to a particular node of the graph that is called *sink*. In the practical case of sensor networks, this node is denoted as base station. A typical transmission structure that is found in practice is the tree, that is, the data are sent from the nodes to the sink, using a tree which has the sink as a root. Since such structures are widely used in networks and lead to computationally efficient communication algorithms, we will restrict our analysis to tree transmission structures. The goal is to gather all data at the sink using this tree (subgraph of the original graph), while minimizing a cost functional (e.g. total flow cost). We refer to this problem as the *correlated data gathering problem*. This problem can be viewed as an instance of a network flow problem, but with an original twist: because the data is correlated, standard solutions may not be optimal, which leads to an original problem that combines the joint optimization of rate allocation and tree building.

An example is shown in Figure 1, where we have N nodes with sources $X_1, \ldots, X_N$, a sink S, and a number of edges that connect the sources. Intermediate nodes can be also used as relays in addition to measuring data. They aggregate their own data with the data received from other nodes, and at the same time, due to the correlation, the intermediate nodes can reduce the necessary rate to code their data. A very important task in this scenario is to find a tree transmission structure on the network graph that minimizes a cost of interest (e.g. flow cost [function(rate)] $\cdot$ [path weight], total distance, etc.). This leads to the question of how to construct efficient data gathering trees.

When the data measured at nodes are statistically independent, the problem becomes separable: because of the statistical independence, the choice of transmission structure does not affect the rate at each node. Namely, first, each node simply encodes its own data independently; then, well developed algorithms can be used to solve various network problems involving costs related to only the link weights (minimum and shortest path spanning tree).

However, in many situations, such as in typical sensor networks, data at nodes are *not* independent. Thus, due to the correlation that is present, it is expected that coding approaches that take this correlation into account (e.g. conditional coding), will outperform traditional approaches, for various cost functions of interest. Moreover, jointly exploiting the data structure and optimizing the transmission structure in the network, can provide substantial further improvements. Therefore, it is worth studying the interaction between the correlation of the data measured at nodes and the transmission structure that is used to transport these data to the sink.

An important practical instance of this type of problem can be found in sensor networks [1, 17, 18]: a number of sensors acquire measurements from the environment (e.g. temperature) which are typically correlated to each other, and these measurements are sent to a base station for decision or control purposes. Let $\mathbf{X} = (X_1, \ldots, X_N)$ be the vector formed by the

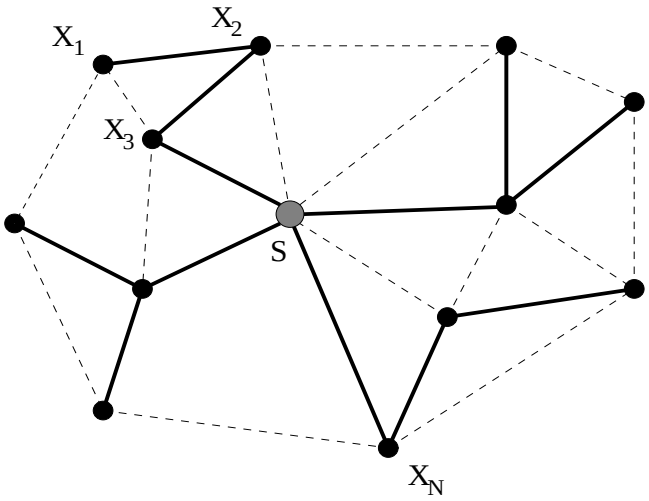


Figure 1: In this example, data from nodes $X_1, X_2, \ldots, X_N$ need to arrive at sink S. A rate supply R_i is allocated to each node X_i. In thick solid lines, a chosen tree transmission structure is shown. In thin dashed lines, the other possible links are shown.

random variables measured at the nodes $1, \ldots, N$. The samples taken at nodes are spatially correlated. We assume that the random variables are continuous and that there is a quantizer in each sensor (with the same resolution for each sensor). A rate allocation $(R_1, \ldots, R_N)$ (each R_i is expressed in bits) has to be assigned at the nodes so that the *quantized* measured information samples are described losslessly, so that they can be fully reconstructed at the sink. That information has to be transmitted through the links of the network to the designated base station. We abstract the communication structure to a connectivity graph with point-to-point links given by the edges of the graph (see Figure 1, where the edges are determined by either the transmission range of nodes, or the by the k-nearest neighborhood). In other words, instead of considering the full wireless multi-point case, we assume a simplified communication model with a medium access control (MAC) protocol, which makes sure that there are no collisions or interferences at a node. A meaningful cost function to minimize is the energy consumption, which is essentially given by the sum of products [function(rate)] $\cdot$ [link weight], for all the links and node rates used in the transmission. Here, the weight of the link between two nodes is a function of the distance d between the two nodes of the link (e.g. kd^{ν} or $k\exp(\nu d)$, with k, ν constants that depend on the transmission medium properties).

There are two complementary approaches that can be used in this problem. The first approach is to allow nodes to use joint coding of correlated data without explicit communication (this is possible by using random binning coding strategies, namely, using Slepian-Wolf coding [4, 19, 22]). With this approach, finding the optimal transmission structure turns out to be simple, because the joint problem of optimizing the transmission structure and the source coding becomes decoupled and can be solved in a separable manner; however data coding becomes complex and global knowledge of the network structure and the correlation structure is needed for an optimal solution. This approach has been treated in [5, 6], where in addition, scenarios including more than one sink are studied.

In the second approach, considered in this paper, nodes can exploit the data correlation only by receiving explicit side information from other nodes (for example, when other nodes

use a node as relay, their data is locally available at that relaying node). Thus, the correlation structure is exploited through communication and joint aggregate coding/decoding locally at each node. We call this approach the *explicit communication* approach. In this case, data coding can be performed in a simple way and relies only on locally available data as side information. However, optimizing the transmission structure becomes complex, as we show in this paper. Notice that in the explicit communication case it is not necessary to know the correlation structure a-priori. This is because the correlation structure is learned *explicitly* in a distributed manner through the explicit communication itself. This leads to a simple source coding, but the transmission structure optimization is hard.

1.2 Related Work

The problem of data gathering has been considered in previous work in the context of sensor networks. Let us briefly review some of the algorithms proposed so far.

In [10], the authors introduce the cluster based LEACH algorithm. In their model, the cluster head nodes compress data arriving from nodes that belong to the respective cluster, and send an aggregated packet to the base station. The work in [15] introduces the PEGASIS algorithm, that uses the [energy] $\times$ [delay] metric over the routing tree; their algorithms find chains of nodes instead of clusters. However, none of these works exploits the correlation present in the data.

In [11], data gathering is done using directed diffusion. Sensors measure events, creating gradients of information in their respective neighborhoods, while the base station requests data by broadcasting interests, meaning events relevant for the base station. The best paths of information flow on which interests fit gradients are reinforced. In order to reduce communication costs, data is aggregated on the way on aggregation trees. Similar work can be found in [8] and [13]. In [8], the authors address the problem of data gathering and compression at relay nodes by using the theory of concave costs applied to single source aggregation. The authors develop an elegant algorithm that finds good trees that simultaneously minimize several concave cost functions of interest. The main difference with our work is that in our case, due to the correlation structure and coding model we consider, the amount of aggregated information sent down the tree to the next hop from a particular node depends on the structure of the subtree whose parent is that node, whereas in [8] that amount only depends on the number of nodes in the subtree, and not on the particular links chosen to build that subtree.

1.3 Main Contributions

We first provide a formal definition of the problem of cost efficient data gathering with explicit communication in a sensor network. Namely, we study the case where joint coding of correlated data by the network is performed explicitly, that is, the reduction in rate by entropy coding due to the correlation is possible at a node only when side information is explicitly available (as relayed data from another node). We consider a simplified version of our general problem setting that results in an original flow optimization problem on a graph. We show some network examples where a joint treatment of rate allocation and transmission structure optimization provides important improvements over the shortest path tree. However, we prove that this optimization problem is NP-hard, by a non-trivial reduction from the min-set cover problem. Then,

we propose a set of distributed heuristic approximation algorithms that provide good solutions for this problem. We show experimentally how a combination of the shortest path tree and traveling salesman paths approximates well the solution given by simulated annealing, that is expected to provide results close to the optimum. Moreover, we present an approximation algorithm that provides tree transmission structure solutions within a constant from the optimal solution, for any possible graph instance (worst-case). We compare the various scenarios through numerical simulations to show how our algorithms provide important improvements in terms of total costs, as compared to the shortest path tree.

1.4 Outline of the Paper

In Section 2 we define the problem studied in this paper. In Section 3 we present a scenario that uses simplified assumptions for our problem setting, and we prove that even in this case, the corresponding optimization problem is NP-hard. In Section 4 we propose a set of heuristic approximation algorithms that provide good average improvements over direct transmission or the shortest path tree. In Section 5 we present an algorithm that generates a spanning tree with cost within a constant bound from the optimal solution. Then, in Section 6, we compare our proposed algorithms by numerical simulations. We provide our conclusions in Section 7.

2 Problem Formulation

We consider the problem of data gathering with a single sink, to which all the data has to be sent. Let $G = (V, E)$ be a weighted graph with $|V| = N + 1$. We denote by S the particular $(N + 1)$th node called sink. Except the sink, every node in the graph generates a source. Each edge $e = (i, j) \in E$ has a weight w_e. Since the data are correlated, depending on the chosen transmission structure, each node i has to transmit a certain rate R_i through the network to the sink. Let $f(x_e, w_e)$ be an arbitrary cost function of the total rate (flow) x_e going through a particular edge with weight w_e. Then the general *minimum cost data gathering tree* problem is defined as follows: find the spanning tree (ST) of the graph G that minimizes the cost function:

$$c_{ST} = \sum_{e \in ST} f(x_e, w_e), \tag{1}$$

under constraints

$$x_{i \to e} - \sum_{e \to i} x_e = R_i, \quad i = 1, \ldots, N; \qquad R_S = \sum_{i=1}^{N} R_i,$$

where we denote by $e \to i$ the set of edges entering node i, and by $i \to e$ the edge from node i to its parent in the tree. We restrict our discussion to functions $f(\cdot, \cdot)$ which are separable as the product of a function that depends only on the rate and another function that depends only on the link weights of the transmission structure[1]. Without loss of generality, we assume

[1]This corresponds to many practical settings (e.g. the [rate] · [path weight] cost function measures the transmission cost in wired networks, and the [exp(rate)] · [path weight] measures the battery consumption in wireless networks, where the [path weight] term is a function of the inter-node distances along a path).

$f(x, w) = x \cdot w$. Then, the expression (1) to be minimized can be rewritten as:

$$c_{ST} = \sum_{i \in V} R_i d_{ST}(i, S) \tag{2}$$

where $d_{ST}(i, S)$ is the total weight of the path connecting node i to S on the ST tree.

The important new feature that makes this problem different from classical network flow theory is the following: by changing the transmission structure, since we change the inter-node distances, both the set of rates $\{R_i\}_{i=1}^N$, which depends on the inter-node correlation, and the path weights $\{d_{ST}(i)\}_{i=1}^N$ are affected. Thus, the optimization of the set of rates and the path weights has to be done jointly, and it cannot be decoupled. We call this new problem the *minimum cost correlated data gathering tree* problem.

We now particularize the optimization problem (2) to the explicit communication based coding setting. In classical network transport theory, the amount of supply (rate in our case) at a node is fixed and independent of the communication links that are chosen to transport the various supplies. In particular, the supply provided by the ith node is independent of the nodes that are connected to the ith node through the chosen edges. In our problem formulation, an important novelty is that the supply at a given node *depends* on the incoming flow from other nodes that use that node as relay, and also on the transmission structure that is used for these nodes.

Consider again the example in Figure 1, where nodes have to communicate their correlated data to one sink. To reduce the complexity of local coding, we assume that each relay node forwards received packets without decoding/re-coding received information and they only perform compression by conditional entropy coding of its own measured data, given the received data from the nodes that are using it as intermediate relay node. Denote by $H(X)$ the entropy[2] of a discrete random variable X, and by $H(X|Y)$ the conditional entropy of a random variable X given that the random variable Y is known. If we consider node X_3, then the rate it has to supply depends on whether:

1. Neither X_1 nor X_2 use X_3 as relay. In this case X_1 uses a rate $H(X_1)$, X_2 uses a rate $H(X_2)$, and X_3 uses a rate $H(X_3)$.

2. Node X_1 uses X_2 as relay, and X_2 transmits its aggregate further to X_3 (this case is shown with solid line in Figure 1). In this case X_1 uses a rate $H(X_1)$, X_2 uses a rate $H(X_2|X_1)$, and X_3 uses a rate $H(X_3|X_1, X_2)$.

3. Both X_1 and X_2 use X_3 as relay. In this case X_1 uses a rate $H(X_1)$, X_2 uses a rate $H(X_2)$, and X_3 uses a rate $H(X_3|X_1, X_2)$.

In the first case, no side information is available at node X_3 from other nodes. Thus, node X_3 sends its entire amount of data on a path to the sink. In the second case, node X_3 does have side information available from node X_2. The information at these two nodes is not independent. Therefore node X_3 can reduce correspondingly the amount of data it sends further. It jointly codes its data with the data from node X_2 and sends the resulting coded data further. In the third case, the amount of side information available is even larger at node X_3, since two

[2]The entropy is a measure of uncertainty of a random variable [4]: $H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x)$, where $\mathcal{X}$ is the discrete alphabet of X.

nodes use it as relay. Thus, the data amount reduction at node X_3 is even larger because the conditional coding involves more sources.

It is clear that in either of these three cases, the optimal transmission structure might not be the shortest path tree. We show in the next Section 3 how the joint dependence of rates and path weights on the transmission structure actually makes our optimization problem NP-hard.

3 Complexity Analysis and Approximation Algorithms

For the sake of simplicity and clarity in our arguments, and without loss of generality in the complexity analysis, we use in this section a simplified model for the data correlation, which allows a clearer analysis of complexity, and for which we develop efficient heuristic approximation algorithms. As we show in this work, this model still completely preserves the original complexity of the optimization problem. Namely, in our model, data at each node are entropy coded with $H(X_i) = R$ bits if no side information is available from other nodes; but only $H(X_i|X_{j_1}, \ldots, X_{j_k}) = r \leq R$ bits, $\forall k$, are needed if the node i has side information available coming from at least another node, which uses node i as relay. Thus, our simplification is that r is constant and does not depend on the number of nodes on which conditioning is done. We denote by $\rho = 1 - r/R$ the correlation coefficient.

3.1 The Tradeoff between Shortest Path Tree and Traveling Salesman Path

In the case of uncorrelated data, if the cost for transmitting over an edge was proportional (by a fixed constant) to the Euclidean length of that edge, then the problem is trivial and the optimal communication structure is the edge connecting the node to the sink. However, for an arbitrary weight function on the edge, transmitting via relays may be better than direct transmission (for example, if the edge weight is d^ν, $\nu > 1$). In the case of correlated data, as it is the case in sensor networks, things become even more interesting, even for very simple networks, because the rates $\{R_i\}_{i=1}^N$ are affected by the choice of the transmission structure.

The example in Figure 2 shows that even in simple network cases, finding good correlated data gathering structures is not trivial at all. If the data were independent, the shortest path tree (*SPT*) would be optimal (see Figure 2 (a)). However, we see that in this example, if $\rho > 1/2$, the *SPT* is no longer optimal, since its cost is larger than the one corresponding to the gathering tree in Figure 2 (b).

Figure 3 shows one other simple network example, with N nodes equally-spaced on an unit length arc circle at distance D from the sink. It is straightforward to show that $\lim_{N\to\infty} \frac{c_{TSP}}{c_{SPT}} = (1-\rho)\left(\frac{1}{2D} + 1\right)$, where c_{TSP}, c_{SPT} are the total flow costs of the two corresponding trees. Consider the case when the number of nodes is very large and the correlation coefficient is arbitrarily close to unity. This means that a path passing through all the nodes and ending at the sink (a traveling salesman path, *TSP*) can be arbitrarily more cost efficient than the direct transmission which corresponds to the *SPT* in this case.

From these simple examples, it can be seen that the correlated data gathering problem with explicit communication is actually a hard optimization problem, in general. Formally, in

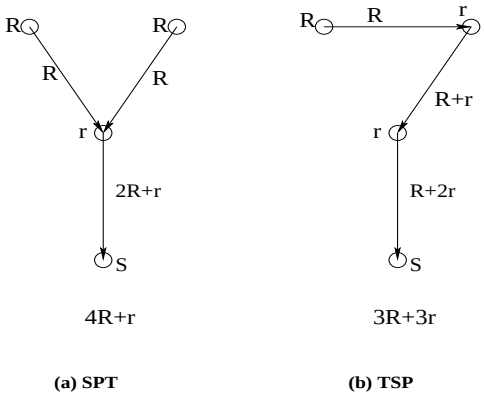


Figure 2: All edges have length 1. The *TSP* (b) outperforms the *SPT* (a) if $R > 2r$.

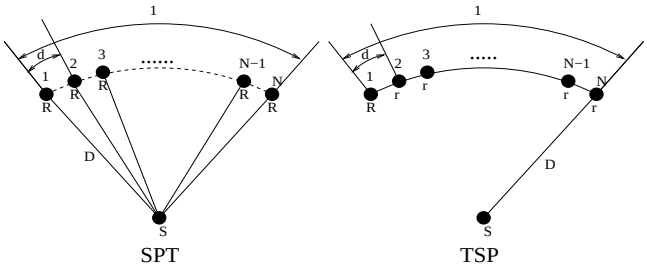


Figure 3: *SPT* vs. *TSP*.

terms of graph optimization, we can rewrite the minimization of (2) for the case of explicit communication as follows:

- **Given:** graph (V, E).
- **Find:** the spanning tree $ST = \{L, T\}$ with L leaves, T non-terminal nodes, $L \cup T = V$, $L \cap T = \emptyset$.
- **such that:**

$$ST = \arg\min_{\{L,T\}} \left(r \sum_{t \in T} d_{ST}(t, S) + R \sum_{l \in L} d_{ST}(l, S) \right)$$

where $d_{ST}(i, S)$ is the total weight of the path on the ST tree from node i to the sink S. In terms of the correlation coefficient, $\rho = 1 - r/R$:

$$ST = \arg\min_{L} \left((1 - \rho) \sum_{i \in V} d_{ST}(i, S) + \rho \sum_{l \in L} d_{ST}(l, S) \right) \tag{3}$$

Let us first look at the two extreme cases, that is $\rho \to 0$ and $\rho \to 1$. When $\rho \to 0$ (independent data), the optimal tree is the *SPT*, which is known to be solvable in polynomial time by e.g. a distributed Bellman-Ford algorithm. At the other extreme, when $\rho \to 1$ (data maximally correlated), the optimal solution is a spanning tree for which the sum of paths from the leaves to the sink is minimum. For this, the core information is taken from the leaf

nodes, and passing through all the in-tree nodes only adds an infinitesimally small amount of new information, since data is strongly correlated. It is straightforward to show that solving this problem is equivalent to solving the *multiple traveling salesman* optimization problem $(k-TSP)$[14], which is known to be NP-hard.

To the best of our knowledge, (3) is an original spanning tree optimization problem on a graph. In Section 3.2, we show that this problem is also NP-hard for the general case $0 < \rho \leq 1$. However, it is possible to design good approximation algorithms and we provide them in Section 4.

3.2 NP-Completeness

In order to prove the NP-hardness of the optimization problem given in (3), we show that the decision version of the problem is NP-complete. The decision version of our optimization problem is:

Definition 1 *Network data gathering tree cost decision problem.*

- Instance*: An undirected graph $G = (V, E)$ with weights w_e assigned to the edges $e \in E$, a positive integer M, and a particular node $S \in V$.*
- Question*: Does the graph admit a spanning tree ST such that, when assigning supplies $R_i = R$ to the leaf nodes and $R_i = r < R$ to the in-tree nodes in the spanning tree ST, the total cost of ST given by (3) is at most M?*

Theorem 1 (NP-completeness) *There is no polynomial time algorithm that solves the network data gathering tree cost problem, unless P=NP.*

Proof: See Appendix A (we use a non-trivial reduction from the min-set cover problem).

Since this problem is a particular version of the general problem given in (3), it follows by a trivial reduction that the general problem is also NP-hard.

Corollary 1 *Minimizing $\sum_{i \in V} R_i d_{ST}(i, S)$ with $R_i = f(\sum_{e \rightarrow i} x_e)$ for an arbitrary monotonic function $f(\cdot)$, is NP-hard.*

Note also that, in general, node i has information from all the nodes in the subtree $sbt(i)$ rooted at node i. Our simplified model is a particular case of this general entropy coding problem, where $H(X_i|\{X_j\}, j \in sbt(i))$ is approximated with $H(X_i|X_j)$, with j being a child of i. Then it can be shown easily that the NP-complexity of our simplified example extends also to this more general case by means of a trivial further reduction.

Note that the NP-hardness of the problem for a single sink generalizes by a straightforward reduction to the case of multiple sinks. However, the derivation of approximation algorithms for the multiple sinks case is significantly more difficult. For instance, the generalization of the *SPT/TSP* structure to multiple sinks is non-trivial due to the interactions between the approximated structures derived for each single sink in particular. This is because nodes that are leaves for a particular structure can be in-tree nodes for other structures, and thus their corresponding rate allocation cannot be uniquely determined. The study of approximation algorithms for the multiple sink case is a subject of our current research.

Arbitrary function of rate

Note that the NP-completeness result holds for any monotonically increasing function of the rate, since an arbitrary function only modifies the values of R and r, but not the multiplicative separable form of the cost function.

3.3 The Dual Problem: NP-Completeness of Broadcast of Correlated Data

The problem formulation for correlated data tree broadcast is essentially provided also by the simplified problem (3), with the difference that now $r > R$, that is, the amount of forwarded data diminishes as it is broadcast from a source node to the extremities of the network. The outer (leaf) nodes can use the data from their parent nodes to fully reconstruct their own data. Thus, in general, relays only need to send further to their children an amount of data equal to the entropy of their corresponding children, conditioned on their own measured data.

Proposition 1 *There is no polynomial time algorithm that solves the dual problem of correlated data broadcast (namely, $r > R$ in Theorem 1), unless P=NP.*

Proof: See Appendix B.

4 Heuristic Approximation Algorithms

In this section we introduce a set of approximation algorithms for solving problem (3).

4.1 Shortest path tree

SPT is computed by using the distributed Bellman-Ford algorithm for simultaneously determining the shortest paths from all nodes to the sink. If the data is independent, this is the optimum solution, but it is far from optimal if there are high correlations.

Algorithm 1 *Shortest Path Tree:*

- *Initialize $D_i = \infty$ and parent nodes with $par(i) = S, i = 1, \ldots, N$.*
- **While** *$par(i)$ changes for some i:*
  - $D_i = \min(D_j + d_{ij}), j = \{1 \ldots N\} \backslash \{i\}$.
  - $par(i) = \arg\min_j (D_j + d_{ij}), j = \{1 \ldots N\} \backslash \{i\}$.
- **Endwhile.**

Note that in the distributed version of the algorithm, the search for $par(i)$ can be done only over a neighborhood of node i (given by the transmission range, or the k-nearest nodes neighborhood).

4.2 Greedy algorithm

We start from an initial subtree composed only of the sink node. Then, we add successively, to the existing subtree, the node whose addition results in the minimum cost increment.

Algorithm 2 *Greedy algorithm:*

- *Let V_{ST} denote the nodes in ST. $V_{ST} = \{S\}$. Let $V_G = V \backslash V_{ST}$.*
- **While:** $V_G \neq \emptyset$
  - **Find:** $\{j_0, i_0\} = \arg\min_{\{j \in V_G, i \in V_{ST}\}} \left(R(d_{j,i} + d_{ST}(i,S)) + (r-R)isLeaf(i)d_{ST}(i,S)\right)$
  - $ST = ST \cup (i_0, j_0), V_{ST} = V_{ST} \cup \{j_0\}, V_G = V_G \backslash \{j_0\}$.
- **Endwhile.**

As expected, given the relationships between the problem in this paper and the TSP problem, greedy algorithms perform suboptimally (as we show experimentally in Section 6), in the same way as the greedy approximation algorithm for TSP provides a quite suboptimal solution. The reason is that far nodes are being left out, so they need to connect to the sink via a path with large weight.

4.3 Simulated annealing

We propose now a computationally heavy method which is known to provide results that are close to optimal for combinatorial problems involving a large number of variables, similar to the problem considered in this paper (e.g. *TSP*). This method was inspired by the *fitness landscape* concept used in evolutionary biology, physics of disordered systems and combinatorial optimization [21]. Its ingredients are: (a) a configuration space Z (finite set of possible *types*), (b) a move set Z (set of adjacencies among types), and (c) a fitness function $f : Z \rightarrow \Re$ (value assigned to each type). The goal is to optimize the fitness over the configuration space.

A very general heuristic optimization method is simulated annealing (SA) [9]. It is based on stochastically simulating the slow cooling of a system and it is closely related to the Gibbs field transition structures. It gives very good results when applied to another NP-hard combinatorial problem in graphs, the traveling salesman problem (TSP) [14, 21].

The fitness landscape formulation [21] of our problem is as follows: (a) the configuration space is the set of all spanning trees (completely defined by the parent relationship), (b) the move set is: one node changes its parent, (c) the fitness function is $g(ST) = R\sum_{l \in L} d_{ST}(l,S) + r\sum_{t \in T} d_{ST}(t,S)$. Our goal is to minimize the fitness over the set of spanning trees.

Algorithm 3 *Simulated annealing:*

- *Take a cooling schedule $T[k], k = 1, \ldots, K$.*
- *Initialize parent nodes with $par(i) = S, i = 1, \ldots, N$. Denote by $\mathcal{N}(i)$ the set of one-hop neighbors of i.*
- **While** $k < K$

- $k = k + 1$, $l = g(ST)$;
- *choose* $i, j \in \mathcal{N}(i)$, *at random such that deleting edge* $(i, par(i))$, *and adding edge* (i, j) *to the tree, does not form a cycle; let* ST' *be the newly generated spanning tree and let* $l' = g(ST')$ *be its corresponding fitness.*
- *make the change* $par(i) \leftarrow j$, *and assign* $ST \leftarrow ST'$ *with probability*

$$p = \begin{cases} 1, & \text{if } \ l' \leq l \\ \exp(-\frac{l'-l}{T[k]}), & \text{if } \ l' > l \end{cases} .$$

- **Endwhile.**

The main feature of this method is that it avoids getting stuck at a local minimum. With a correctly chosen cooling schedule and if the algorithm runs for a sufficient number of steps, the final version of the spanning tree ST is very close to optimal. The convergence of the method depends on the *ruggedness* and *neutrality* of the fitness landscape [21]. For $\rho = 0$ (SPT), our experiments show that it does provide the exact solution, and convergence is easy to achieve. When ρ is close to 1, the generated landscape is not smooth any longer, so convergence is difficult to obtain in a reasonable number of iterations. We obtained good results (iteration steps vs. ruggedness) with the Lundy and Mees schedule [16]:

$$T_k = \frac{T_{k-1}}{1 + \frac{T_0 - T_K}{K T_0 T_K}}.$$

A provably optimal, but slow schedule for T_k is $c/\log(1+k)$ [9], with c the maximum local minimum depth of the landscape.

However, in general, simulated annealing is usually hard to implement in a decentralized manner, and is computationally expensive. It does however provide a good benchmark close to optimal against which other heuristic algorithms can be tested.

4.4 Balanced *SPT / TSP* tree

We propose a heuristic approximation algorithm consisting of a combination of SPT and $k - TSP$, from the solutions obtained using simulated annealing. The solution provided by this algorithm consists of a SPT structure around the sink that has a certain radius and a set of TSP paths starting from each of the leaves of the SPT. Depending on the amount of correlation, that is the value of ρ, a certain radius for the SPT is more appropriate. We briefly describe the intuition why there is such a value for this radius. Since the leaf nodes contribute most to the cost ($R > r$), then in order to minimize the cost, the flows of R data coming from the leaves of the tree have to travel short paths to the sink (the SPT effect), but in the same time through as many nodes as possible, to reduce the total number of leaves (the TSP effect). On the other hand, when the correlation is large (r is small), the effect of transporting flows of r data through the tree is negligible, so it is essential to have as many in-tree nodes as possible, thus the TSP effect is more important, whereas when the correlation is small (r is large), it is more important that the data from in-tree nodes reach the sink on shortest paths, and thus the SPT effect becomes more pronounced.

Algorithm 4 *SPT/TSP balanced tree:*

- *Build the SPT for the nodes that are in a radius $q(\rho)$ from the sink. Denote this SPT by ST. The optimal choice for the radius $q(\rho)$ decreases with the increase of the correlation coefficient ρ.*
- *Let V_{ST} denote the nodes in ST. Let $V_{TS} = V \backslash V_{ST}$.*
- **While** $V_{TS} \neq \emptyset$
  - *Denote by L the set of leaves of ST.*
  - $\{i_0, l_0\} = \arg\min_{\{i \in L, l \in V_{TS}\}} (d(l,i) + d_{ST}(i,S))$.
  - $ST = ST \cup (i_0, l_0), V_{ST} = V_{ST} \cup \{i_0\}, V_{TS} = V_{TS} \backslash \{i_0\}$.

This is actually a suboptimal nearest neighbor approximation of the $k - TSP$, which is easily implementable in a distributed manner.

Square grid network graph: optimal radius for the SPT/TSP algorithm

Since the TSP problem is NP-complete, it is difficult to provide an analytical study of the dependence of the optimal SPT radius on the correlation structure for a general connectivity graph. Therefore, for analysis, we restrict our attention to a square grid graph and study in detail the structure of our SPT/TSP algorithm in this case. Namely, we study the dependence of the optimal radius on the correlation coefficient ρ for this graph.

Consider a square grid network with $(2n+1) \times (2n+1)$ nodes (see Figure 4). The SPT is build on the square area of $(2m+1) \times (2m+1)$ nodes around the sink. Note that the SPT subtree has $8m$ leaves. For the rest of the graph, equal length TSP paths are built. Namely, for each leaf of the SPT subtree, a TSP rooted at that leaf node is constructed, which spans $\texttt{floor}\,((2n+1)^2 - (2m+1)^2/(8m))$ of the nodes left outside the SPT subtree.

We plot in Figure 5 the total cost of the SPT/TSP tree as a function of the correlation coefficient ρ. We note that, as expected, the optimal SPT radius m decreases with the increase of the correlation coefficient ρ.

Next, we compute analytically the optimal 'radius' m/n of the SPT subtree as a function of the correlation coefficient $\rho = 1 - r/R$. After some computations, we obtain that the optimal m is a root of the following polynomial: $P(Z) = 3rZ^4 + (8r - 16R)Z^3 + (-r - 4rn + 4R - 4rn^2)Z^2 + (rn^4 + 2rn^3 + rn^2)$. This polynomial has 4 roots, but by solving it numerically, we find that only one of them is in the interval $[0, n]$. We plot this solution for the optimal radius in Figure 6. The discontinuity at $\rho = 0$ is due to the properties of the very particular regular grid structure that is analyzed. A particular interesting abrupt phenomenon is observed asymptotically: when n is sufficiently large, there is an optimal normalized radius for the SPT, which does not depend on the correlation coefficient ρ.

4.5 Leaves deletion approximation

This algorithm is a simplified version of the TSP/SPT algorithm. Namely, this algorithm constructs first the global SPT, and then uses one-hop TSP paths from the outer nodes

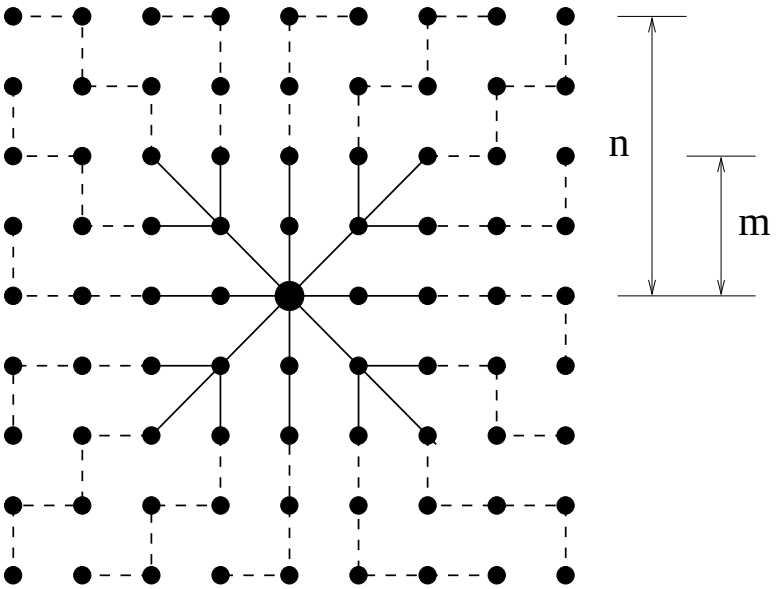


Figure 4: Square grid network: the SPT (solid lines) is built on the nodes in the $m \times m$ sub-grid around the sink (larger black dot). The rest of the nodes are spanned by TSPs (dashed lines) rooted in the leaves of the SPT.

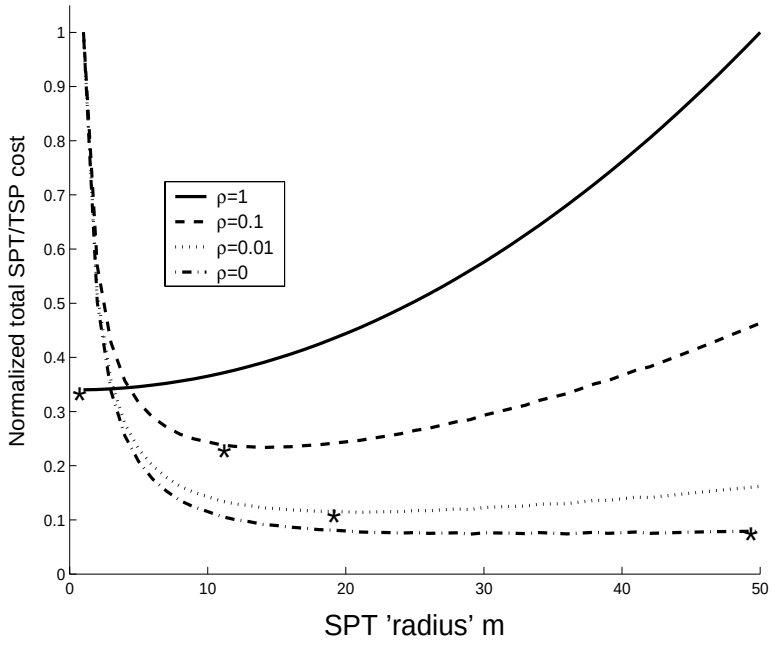


Figure 5: Square grid network: normalized cost of the SPT/TSP tree for a grid network of size $N = 101 \times 101$ nodes ($n = 50$) and several values of the correlation coefficient ρ. Note how the optimum value of the radius m increases from 0 to n as ρ decreases from 1 (high correlation) to 0 (no correlation).

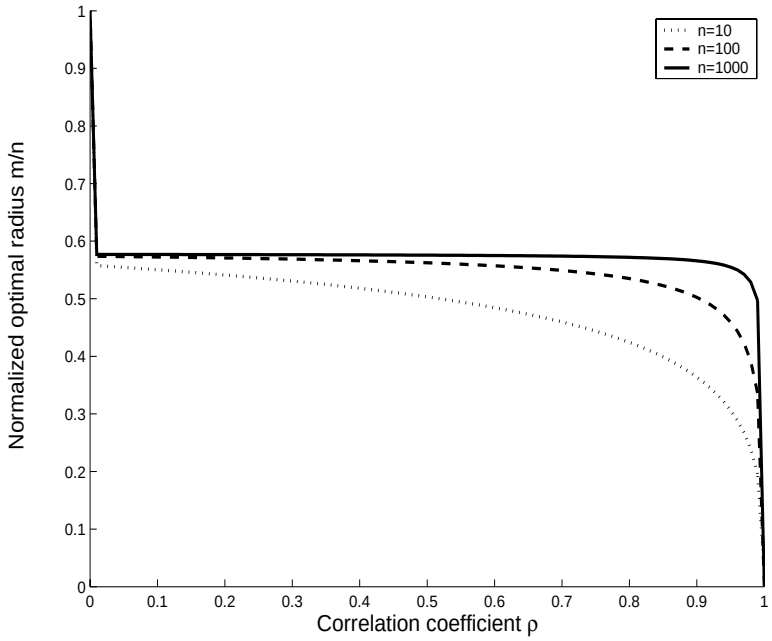


Figure 6: Square grid network: optimal radius of the SPT (normalized with respect to the radius of the square grid), as a function of the correlation coefficient ρ, for various sizes $N = (2n+1)^2$ of the network.

of the SPT. It is based on the observation that good cost improvements may be obtained mainly by making the leaf nodes change their parent node to some other leaf node in their neighborhood. This operation is done only if it reduces the total cost of the whole tree.

Algorithm 5 *Leaves deletion algorithm (LD):*

- *Initialize* $ST \leftarrow SPT$. *Each node* i *maintains its parent, number of children, and total distance* $d_{ST}(i,S)$ *on the current spanning tree to the sink. Let* $par(i)$ *denote the parent node of node* i.
- **While** *there is a decrease in cost:*
  - **For each** *leaf node* i*: Find the leaf node* $j \in \mathcal{N}(i)$ *that maximizes* $R(d_{ST}(i,S) + d_{ST}(j,S)) - (R(d_{i,j} + d_{ST}(j,S)) + rd_{ST}(j,S)) - I(i)$*, where* $I(i)$ *is an adjustment term indicating the cost lost by transforming single parent nodes into leaves. If the maximizing quantity is positive, then assign* $par(i) \leftarrow j$ *and update the corresponding distances on the tree to the sink, and number of children, for all the three nodes involved* $\{i,$ *former* $par(i),\ j\}$.
- **Endwhile**.

This algorithm involves a small number of iterations after SPT is computed, and is fully distributed. Note that a known good approximation for the geometric TSP is to start from the minimum spanning tree (MST) and eliminate the leaves by successively passing the traveling salesman path through them. Here, we can see that a simplified similar procedure provides good results in our case as well, which confirms the link between our problem and the TSP.

5 Strict Approximation Algorithms

In this section we present a strict approximation algorithm, that is an algorithm which guarantees a solution for which the cost is only a constant factor higher than the cost of an optimal solution. We start this section by giving a lower bound on the cost of an optimal solution.

Lemma 1 (Lower Bound) *The cost of the optimal solution c_{opt} is bounded from below by $c_{opt} \geq \max(r \cdot c_{SSP}, R \cdot c_{MST})$, where c_{SSP} is the sum of the costs of all the shortest paths to the sink, and c_{MST} is the cost of the minimum spanning tree of all the nodes, including the sink.*

Proof: Nodes in the network can either send their raw data directly to the sink, or use the raw data of other nodes to code their data, and then send their coded data to the sink. Let the nodes who send their data in the raw format be the set B. Let the nodes who code their data using using the raw data of node u be the set C_u. The set B and the sets C_u for all $u \in V$ form a partition over all nodes, that is: $V = B \cup \sum_{u \in B} C_u$.

After deciding how the set of nodes will be partitioned, an optimal algorithm will use the shortest paths (SP) to deliver the raw data from nodes in B to the sink. Similarly, the encoded data of nodes in set C_u will travel along shortest paths (SP) to the sink. Nodes from C_u need to encode their data using the raw data of node u, u being a node in set B. On the other hand the sink needs to decode the encoded data of nodes from C_u; to do so, the sink needs the raw data of node u too. The optimal way to distribute the raw data of u is given by the minimum spanning tree (MST) between the nodes in the set C_u, node u itself, and the sink. Summing up, the cost of the optimal algorithm is therefore

$$c_{opt} = \sum_{u \in B} R \cdot SP(u, sink) + \sum_{u \in B} \left(R \cdot MST(C_u, u, sink) + \sum_{v \in C_u} r \cdot SP(v, sink) \right).$$

We can bound this equation in two ways from below. Since the sets B and C_u form a partition of all nodes V, and since $r \leq R$, each node must transmit its data to the sink on the shortest path, at least in the coded form. Therefore the optimal cost contains at least the sum of the shortest paths (SSP) of the coded data:

$$\begin{aligned} c_{opt} &= \sum_{u \in B} R \cdot SP(u, sink) + \sum_{u \in B} \left(R \cdot MST(C_u, u, sink) + \sum_{v \in C_u} r \cdot SP(v, sink) \right) \\ &\geq \sum_{u \in B} r \cdot SP(u, sink) + \sum_{u \in B} \sum_{v \in C_u} r \cdot SP(v, sink) \\ &= \sum_{u \in V} r \cdot SP(u, sink) = r \cdot c_{SSP}. \end{aligned}$$

On the other hand, since B and C_u form a partition of all nodes V, the terms containing raw data (R) must include a spanning tree. Since the minimum spanning tree (MST) is the cheapest possible spanning tree, the cost of the optimal algorithm is also bounded from below by the cost of the MST, used to transmit the uncoded data. The lemma follows immediately.

■

In the following we present an approximation algorithm that is optimal up to a constant factor. The algorithm is based on the *shallow light tree* (SLT), a spanning tree that approximates both the MST and the shortest paths for a given node (e.g. the sink). The SLT was introduced in [2, 3]. Given a graph $G(V, E)$ and a positive number γ, the SLT has two properties:

- Its total cost is at most $1 + \sqrt{2}\gamma$ times the cost of the MST of the graph $G(V, E)$;
- The distance on the SLT between any node in V and the sink is at most $1 + \sqrt{2}/\gamma$ times the shortest path from that node to the sink.

For more details on the construction of the shallow light tree (SLT) we refer to [12].

The algorithm is as follows: First the SLT spanning tree is computed, the sink being the root of the SLT. Then the sink broadcasts its value R_{sink} to all its one-hop neighbor nodes in the SLT. When node v is receiving a value R_u from a neighbor u, node v encodes its locally measured data R_v using R_u, and transmits its encoded value r_v to the sink on the path given by the SLT. Then node v broadcasts its value R_v to all its one-hop neighbors but u; in other words to all its children but not its parent in the SLT. We call this the SLT algorithm.

The sink has its own data R available locally (or it can use the R data of its first-hop neighbors), and thus can perform recursive decoding of the gathered data, based on the encoded r values that it receives from all the nodes.

Theorem 2 *The SLT algorithm is a $2(1 + \sqrt{2})$-approximation of (3).*

Proof: The total cost of the SLT algorithm is given by

$$c_{SLT} = R \cdot c_{SLT} + \sum_{v \in V} r \cdot |SLT - Path(v, sink)|.$$

The first term follows from the fact that each node sends its raw data to all its children in the SLT. The second term corresponds to the sum of the shortest paths in the SLT. Using the SLT properties we have $c_{SLT} \leq R \cdot (1 + \sqrt{2}\gamma)c_{MST} + r \cdot (1 + \sqrt{2}/\gamma)c_{SSP}$. We choose $\gamma = \frac{1}{2\alpha}(-\beta + \sqrt{2}\beta + \sqrt{3\beta^2 - 2\sqrt{2}\beta^2 - 4\sqrt{2}\alpha\beta + 8\alpha\beta})$, with $\alpha = R \cdot c_{MST}$ and $\beta = r \cdot c_{SSP}$. Then

$$c_{SLT} = (1 + \sqrt{2})(-\beta + \sqrt{2}\beta + \sqrt{3\beta^2 - 2\sqrt{2}\beta^2 - 4\sqrt{2}\alpha\beta + 8\alpha\beta}).$$

Dividing c_{SLT} by $c_{opt} = \max(\alpha, \beta)$ as derived in Lemma 1, the second factor of c_{SLT} will be upper bounded by 2, and the approximation ratio will consequently be $(1 + \sqrt{2}) \cdot 2 \approx 4.828$. This ratio becomes tight at $\alpha \approx \beta$; if $\alpha \gg \beta$ or $\beta \gg \alpha$ the approximation ratio of the SLT algorithm is better. ∎

Thus SLT provides a worst-case bound for our problem. Figure 7 shows the best choice of γ for the SLT, found experimentally, as a function of the correlation coefficient ρ. Note that when the correlation is small $\rho \approx 0$, the optimal choice of γ (a large value) gives a result of the SLT which is close to SPT. On the contrary, for a large correlation $\rho \approx 0$, a good SLT should be close to the MST (value of γ close to 1), and the MST is known to approximate the TSP within a constant. These results for the best choice of the parameter γ for the SLT approximation are as expected, following our discussion in Section 3.

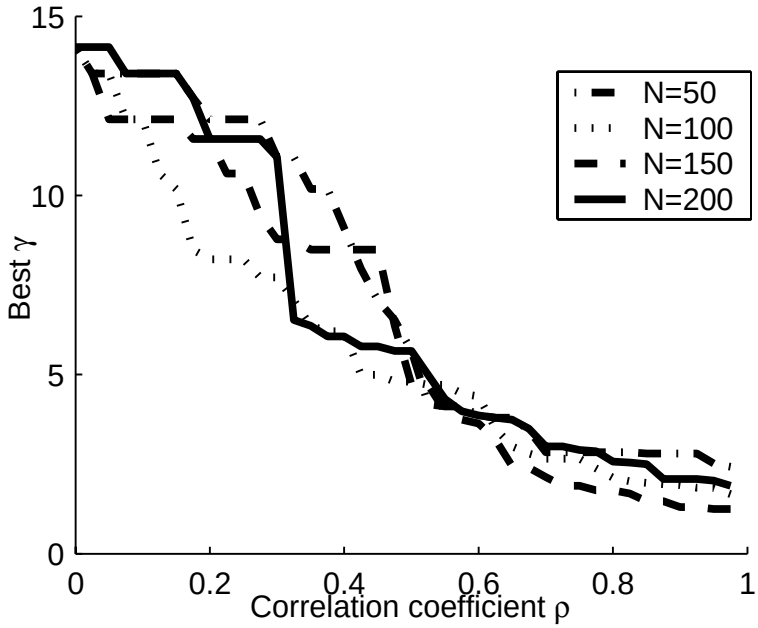


Figure 7: Best choice of the parameter γ, as a function of the correlation coefficient ρ. The average has been done over 20 random network instances for each pair (ρ, N).

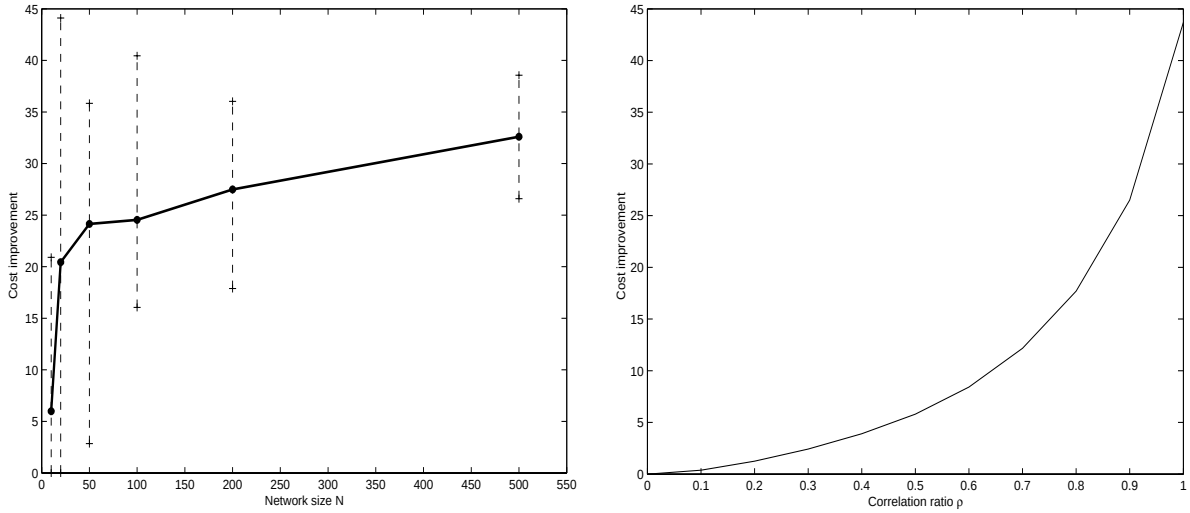
6 Numerical Simulations

Our simulations were done in MATLAB for a network of nodes randomly distributed on a 100×100 grid, with a value $\nu = 2$ for the power of the distance. We consider several sizes of the network, from $N = 10$ up to $N = 500$ nodes, and various values for the correlation coefficient ρ among the nodes, within the interval $\rho \in [0, 1]$. As mentioned before, the algorithm that is used for finding the SPT in a distributed manner is a distributed version of the Bellman-Ford algorithm, which runs in $\mathcal{O}(N|E|)$ steps. The actual speed of convergence depends on the degree of each node in the graph, which in turn depends on the range $\mathcal{N}(i)$ over which nodes search for neighbors. For the graph structures we consider, Bellman-Ford runs in an average of 50 steps for a network size of 500 nodes.

Our experiments show important average improvements of the LD algorithm over the SPT for nodes randomly distributed on a 100×100 grid (see Figures 8–11). The computational load of LD is small, namely at most 4 iteration steps after the SPT are required for its implementation, while the algorithm is still distributed. It also outperforms the greedy algorithm (see Figure 9). In this experiments, the sink is located at the center of the grid. Similar performances are obtained when the sink is situated outside the network area.

It can be seen that some clearly non-optimal patterns appear on the gathering tree solutions obtained using our LD heuristic algorithm. This is due to the fact that there are cases when leaf nodes do not have other close leaves on the graph, unless they choose leaves for which the corresponding connecting edge crosses over some already existing edges.

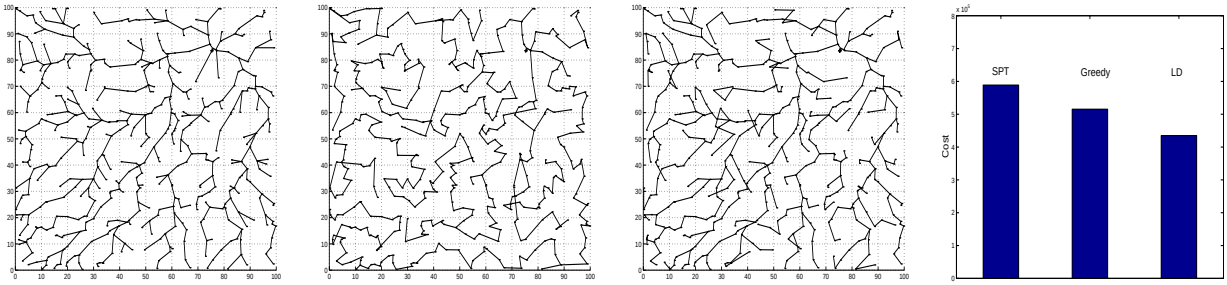
When comparing the various heuristic algorithms with the simulated annealing solution, which is expected to provide results close to optimal, we notice that our simple heuristic algorithms perform relatively well, while being completely distributed, scalable and efficient from a



(a) Cost improvement (in %) vs. N.

(b) Cost improvement (in %) vs. ρ.

Figure 8: Average total cost decrease $100 \cdot (\frac{c_{SPT}}{c_{LD}} - 1)$, in %, of leaves deletion (LD) over shortest path tree (SPT) for (a) $\rho = 0.9$ and $N = 10, 20, 50, 100, 200, 500$, and (b) $N = 200$ and $\rho = 0, 0.1, \ldots, 1$.



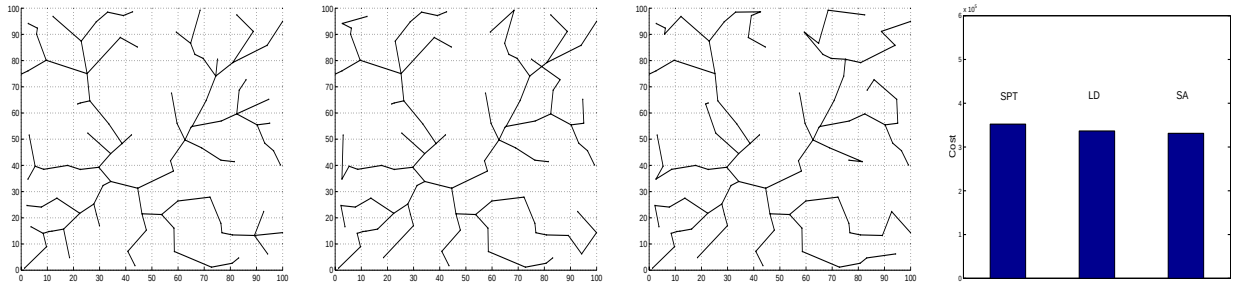
(a) SPT algorithm

(b) Greedy algorithm

(c) Leaves deletion algorithm

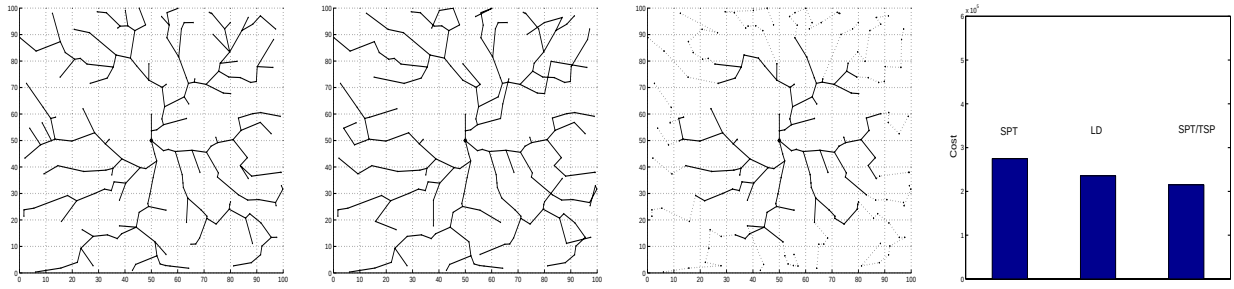
(d) Total cost

Figure 9: Data gathering tree algorithms on a network instance: $N = 500$, $\rho = 0.8$: (a) Shortest path tree (SPT), (b) Greedy algorithm, (c) Leaves deletion (LD), (d) Total cost. The total cost when the nodes transmit their data directly to the sink is one order of magnitude larger.



(a) *SPT* algorithm (b) Leaves deletion algorithm (c) Simulated annealing (d) Total cost

Figure 10: Data gathering tree algorithms on a network instance: $N = 100$, $\rho = 0.5$: (a) Shortest path tree (SPT), (b) Leaves deletion (LD), (c) Simulated annealing, (d) Total flow cost. Costs for this instance: SPT: 3.52e+6; LD: 3.36e+6; SA: 3.31e+5.



(a) *SPT* algorithm (b) Leaves deletion algorithm (c) *SPT*/*TSP* algorithm (d) Total cost

Figure 11: Approximated gathering trees on a network instance: $N = 200$, $\rho = 0.2$: (a) Shortest path tree (SPT), (b) Leaves deletion (LD), (c) SPT/TSP algorithm. Costs for this instance: *SPT*: 2.74e+5; *LD*: 2.36e+5; *SPT/TSP*: 2.15e+5.

complexity point of view (see Figure 10). Note that it is possible that simulated annealing did not provide the optimal solution either, but it is expected to do so with the right scheduling policy and with a long enough running time.

We show in Figure 11 some simulation results for the SPT/TSP algorithm. For networks with $\rho = 0.2$ and $N = 200$, the improvements are of the order of 10% over the LD algorithm. As the simulated annealing results suggest, a good tree solution has a small number of leaves, but at the same time short paths to the sink. Solutions for a network instance are shown in Figures 10–11. In Figure 11(c) we plot the branches in the SPT subtree in solid lines, and the branches added in the step involving TSP paths are shown in dashed lines.

Our experiments show important improvements of the LD and the SPT/TSP algorithms over SPT, in terms of average performance over randomly generated networks (see Figure 12).

For illustrative purposes, we show in Figure 13 the SLT tree and the SPT/TSP tree for a network instance with $N = 100$. In terms of total cost, as expected, from an average case point of view, the SPT/TSP algorithm performs better than the SLT algorithm (see Figure 14). In these results, the value of the radius $q(\rho)$ for the SPT/TSP has been chosen as in Figure 6, and for the SLT, the value of γ has been chosen as in Figure 7. Note that for small values of

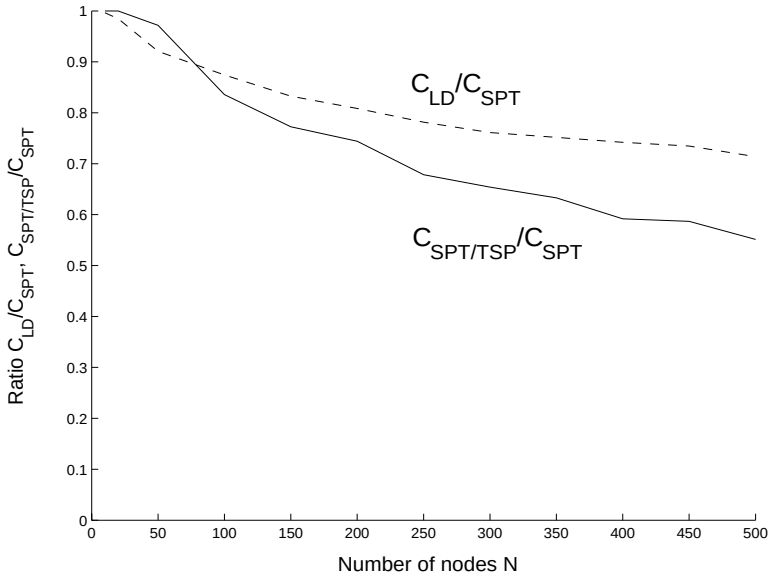


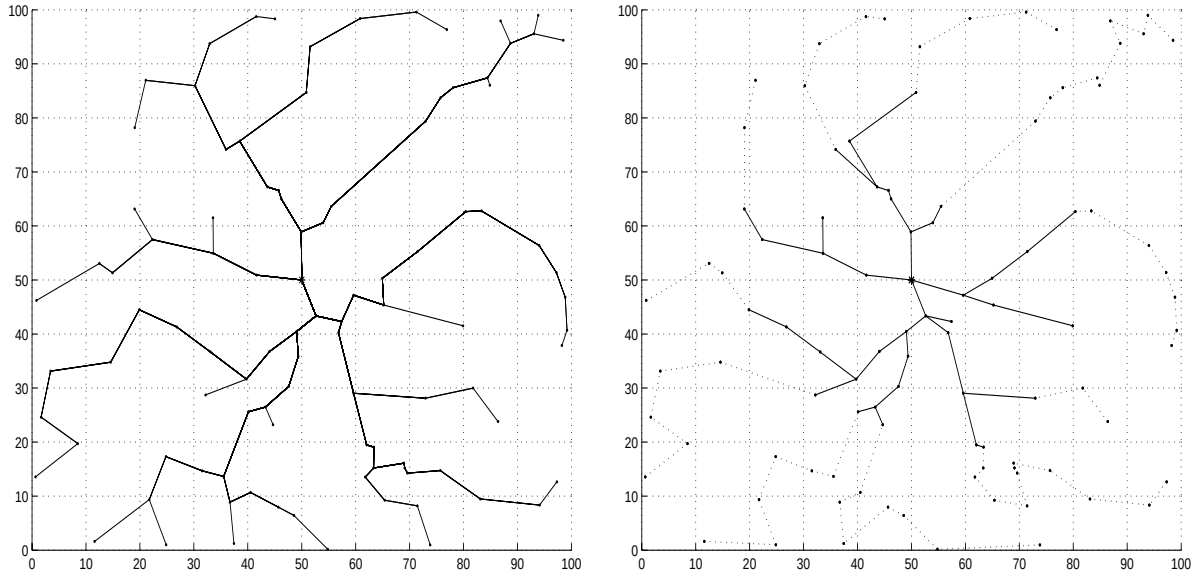
Figure 12: Average ratios of total costs between leaves deletion (LD) and SPT, and between balanced SPT/TSP and SPT: $\rho = 0.9$.

the correlation coefficient $\rho \approx 0$, the two trees perform similarly, since both algorithms provide solutions close to the SPT, which is the optimal solution when there is no correlation in the data. When the correlation coefficient ρ approaches 1, the SLT provides a solution close to the MST, and thus, the ratio between the costs provided by the two algorithms shows how well the MST approximates the TSP. Namely, the MST provides a constant approximation for the TSP in the worst case, while, by design, the SPT/TSP algorithm searches for better approximations for the TSP. We conclude that when the correlation is low, the two algorithms perform similarly, while when the correlation is important (this is the usual case in sensor networks), the SPT/TSP algorithm provides better results, since by definition it approximates better the multiple TSP than does the SLT.

7 Conclusions

In this paper, we formulate the network correlated data gathering tree problem with coding by explicit communication. Namely, we address an optimization problem that considers transmission structure optimization in networks where connectivity is modelled as a graph. A transmission tree structure implies both a certain rate allocation at the nodes and a certain transmission cost per bit between connected nodes. We first proved that the problem is NP-hard even for scenarios with several simplifying assumptions. We propose approximation algorithms for the transmission structure that provide significant gains over the shortest path tree. Moreover, our algorithms provide solutions close to the optimal, which is shown experimentally by comparing our approximation algorithms to a provably optimal but computationally heavy optimization method, namely, simulated annealing.

Future work include the derivation of approximation algorithms for general aggregation schemes (including the case of full entropy conditioning based on all nodes in the corresponding



(a) *SLT* algorithm (b) *SPT/TSP* algorithm

Figure 13: Approximated gathering trees on a network instance: $N = 100$, $\rho = 0.8$: (a) Shallow light tree (SLT), (b) SPT/TSP algorithm. Costs for this instance: *SLT*: 1.79e+005; *SPT/TSP*: 1.55e+5.

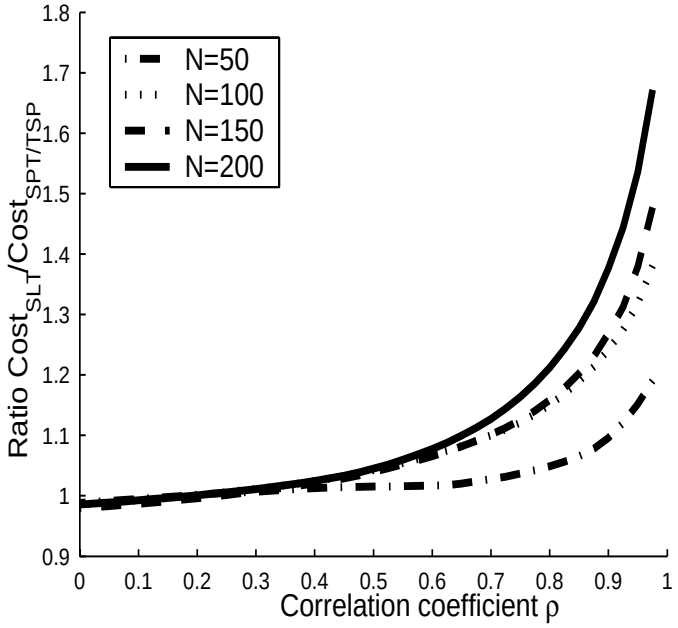


Figure 14: Average ratio of the SLT cost vs. the SPT/TSP cost. The average has been done over 20 random network instances for each pair (ρ, N), with values for γ as in Figure 7.

subtree at relay nodes), and the study of combined Slepian-Wolf and explicit communication data gathering approaches [6].

A Proof of Theorem 1

First, the decision version of our problem is in NP: a nondeterministic algorithm needs to guess the parent relationship (that is, specify the parent node for each of the nodes), and then find in polynomial time the nodes that are not parent nodes, assign to all nodes the number of bits corresponding to either leaf or in-tree node, and test that its total cost is less than the given value M.

Next, to prove the NP-hardness, we perform a reduction from the set cover problem [7], whose decision version is defined as follows:

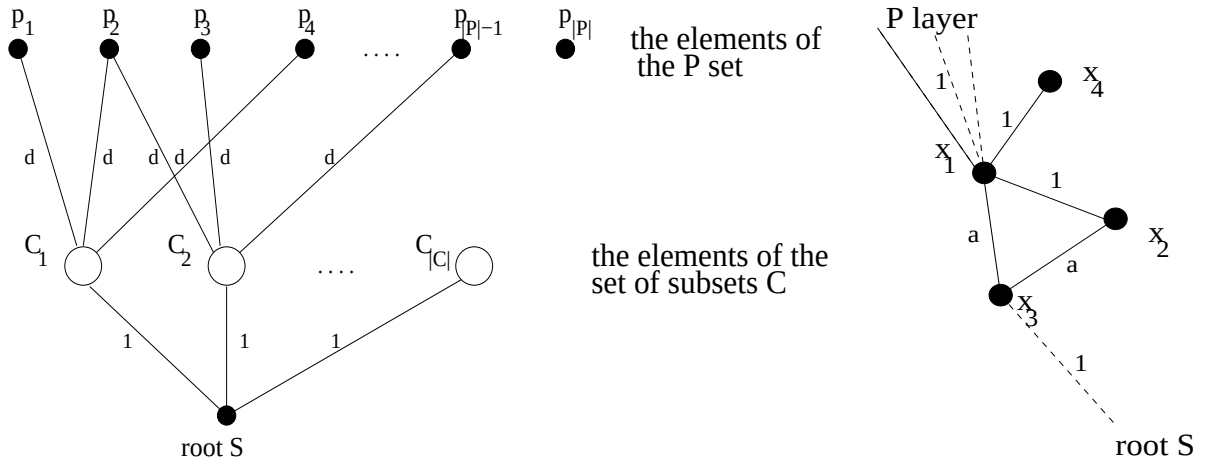
Definition 2 *Set cover.*

- Instance*: A collection C of subsets of a finite set P and an integer $0 < K \leq |C|$, with $|C|$ the cardinality of C.*
- Question*: Does C contain a subset $C' \subseteq C$ with $|C'| \leq K$, such that every element of P belongs to at least one of the subsets in C' (this is called a set cover for P)?*

For any instance of the set cover problem we build an instance of our decision problem. Figure 15(a) illustrates the construction of the graph instance for our problem. The resulting graph is formed of three layers: a sink node S, a layer corresponding to the subsets $C_i \in C$, and a layer corresponding to the elements $\{p_j\}$ of the set P. For each element $C_i \in C$ we build a structure formed by 4 nodes x_1, x_2, x_3, x_4, as in Figure 15(b) (there are four different nodes for each subset C_i, but we drop the superscript C_i of the nodes x for the sake of simplicity). This structure originates from our toy example in Section 3.1 and has properties linked with the tradeoffs observed there. The node x_3 is linked to the sink S, node x_4 is connected only to node x_1, and x_1, x_2, x_3 are all interconnected. Furthermore, we connect each structure $C_i \in C$ (namely the node x_1 from that structure) to only the nodes in the P layer that correspond to elements contained in C_i (example: in the instance in Figure 15(a), subset $C_1 = \{p_1, p_2, p_4\}, C_2 = \{p_2, p_3, p_{|P|-1}\}$ etc.) All the edges connecting the P layer to the C layer have a weight $d > 0$; for all C_i, the edges of type (x_1, x_3) and (x_2, x_3) have weight $a \geq 1$; the rest of the edges shown in Figure 15(a) have all weight 1. All other edges are assumed of infinite weight and are not plotted. Without loss of generality, we consider that in-tree nodes use $r = 1$ bits for coding their data, while leaf nodes use $R > 1$ bits.

The goal is to find a spanning tree for this graph, for which the cost in (3) is at most M. We now show that if $M = |P|(d+a+1)R + K(2aR+3R+a+2) + (|C|-K)(aR+3R+2a+4)$, for the positive integer $K \leq |C|$, then finding a spanning tree with cost at most M is equivalent to finding a set cover of cardinality K or less for the set P. Notice that the construction of our graph instance from the set cover instance can be performed in polynomial time.

With a large enough value chosen for d (i.e. $d > |C|(2aR + 3R + a + 2)/R$), a tree with cost at most M will contain exactly $|P|$ links between the layers P and C. That means that no p_j node is used as relay, so all $p_j \in P$ are necessarily leaf nodes. If some p_j node was used as relay, then the cost of the tree would contain R bits passing through more than $|P|$ such



(a) A graph instance

(b) Inner structure of each subset C_i

Figure 15: Reduction from the min-set cover problem.

links, which would result in a cost larger than M. This also implies that the only way the C_i structures can connect to the sink S is via their corresponding x_3 node, so all x_3's must be in-tree nodes. Furthermore, all x_4's nodes need to be connected to their corresponding x_1 node in order to belong to the tree, so necessarily all x_4-s are leaf nodes and all x_1's nodes are in-tree nodes. The only degrees of freedom are the choices of two out of the three edges interconnecting the nodes x_1, x_2, x_3, for each structure C_i.

The key idea of our proof is that, for properly chosen values for d and a, finding a tree with cost at most M means connecting the nodes in layer P to *at most* K nodes of layer C. If the tree needs to connect the layer P to more than K nodes in layer C, then the cost of the tree will necessarily be higher than M. The intuition is that 'detours' via the (x_1, x_2) edges are worthy from the point of view of cost reduction only if the flow that goes through node x_1 comes exclusively from node x_4 and no flow from the P layer goes through x_1. If some flow from the P layer joins as well, then the optimal path would use the edge (x_1, x_3) instead. In this latter case, we see now that for optimality, the edge (x_1, x_2) should not be used.

We choose a value of $a \geq 1$ such that $(a+2)/a < R < (a+2)/(a-1)$. Note that, for a given $R > 1$, it is always possible to choose a value for a that fulfills this condition.

With the given weights on the edges, if no p_j node is connected to a C_i structure, then since $R > (a+2)/a$, the optimal pattern (pattern 1, see Figure 16) for this structure contains the links $(x_4, x_1), (x_1, x_2), (x_2, x_3), (x_3, S)$, with cost $(a+3)R + (a+2) + (a+1) + 1$. The other possible structures contain either links $(x_4, x_1), (x_1, x_3), (x_2, x_3), (x_3, S)$ (pattern 2) with cost $(a+2)R + (a+1)R + (a+1) + 1$, or links $(x_4, x_1), (x_1, x_3), (x_2, x_1), (x_3, S)$ (pattern 3) with cost $(a+2)R + (a+2)R + (a+1) + 1$. They both are sub-optimal if $R > (a+2)/a$ (since pattern 2 is always better than pattern 3, we will consider only pattern 2 for the rest of our proof).

However, when $m \geq 1$ nodes $\{p_j\}_{j=1}^{m}$ from the P layer connect to x_1, for any of C_i's, the pattern 1 is no longer optimal, because it has a cost $m(d+a+2)R + (a+3)R + (a+2) + (a+1) + 1$. The alternative structure (pattern 2) has cost $m(d+a+1)R + (a+2)R + (a+1)R + (a+1) + 1$, which is more efficient if $m \geq 1$, and $R < (a+2)/(a-1)$. We notice that in an optimal tree the cost to transmit data from each p_j to the sink S is the same for all p_j's nodes (and equal

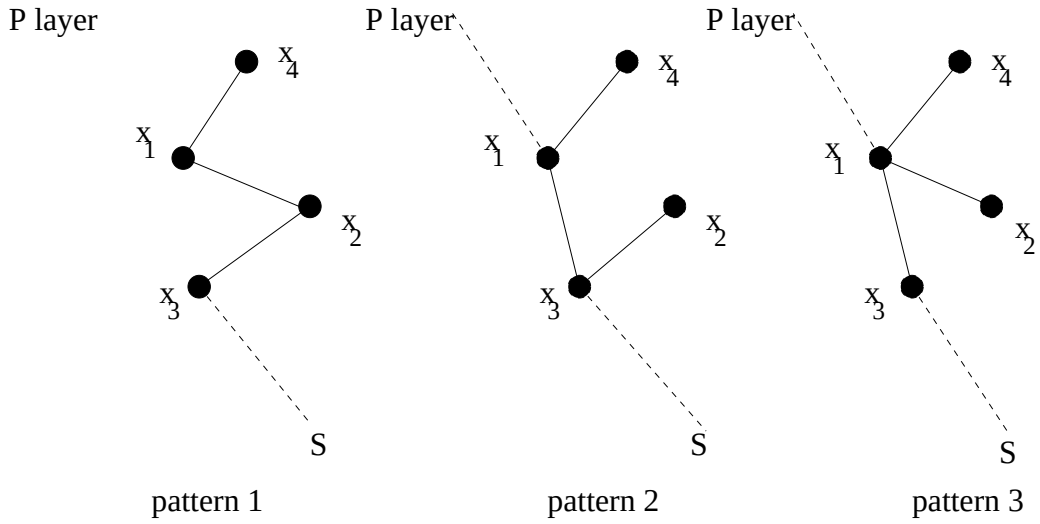


Figure 16: The three possible gathering patterns for the substructure C_i.

to $(d+a+1)R$). Therefore the goal is to keep minimal the part of the total cost corresponding to the rest of the nodes (i.e. nodes in layer C).

That means that to find a tree with cost less or equal to $|P|(d+a+1)R + K(2aR+3R+a+2)+(|C|-K)(aR+3R+2a+4)$ is equivalent to finding a set of K elements or less from the C layer to which all nodes in the set P connect. This is actually achieved by having at most K nodes of type x_1 used to connect to the p_j's nodes, which turns out to be equivalent to finding a set cover for the set P of size K or less, that is to solving the set cover problem.

Thus our decision problem is NP-complete and our optimization problem NP-hard. ■

B NP-Completeness Correlated Data Broadcast

We prove that the problem is NP with a reduction from 3-SAT. The reduction works as follows: for any 3-SAT instance, we build a 3-layered network: sink, variables, clauses (Figure 17). We link the sink to nodes corresponding to each of the variables, and add two nodes for each variable corresponding to the true and false possible values for the respective variable, and one more node for selecting *at least* one of the variables values. Then we add one more layer with one node for each clause, and link it to the corresponding true or false node, that is contained in that clause. We show that finding a minimum tree for this instance of the problem is equivalent to finding a satisfying assignment of the variables in the 3-SAT instance. We will do this by choosing such values for the edges so as to force the optimal tree to contain *one single branch*, corresponding to either the true or false node, per variable (that is, all clauses are linked to at most one of the two nodes of any variable).

Assign weight 1 to all edges except the ones connecting the clauses to the variables, which have weight d chosen large enough so an optimal tree will not pass through more than $|C|$ such links. Then, a 3-SAT instance is satisfiable if and only if the corresponding graph admits a data gathering tree of size $|C|\cdot R(d+2)+|V|\cdot 3R+|V|\cdot 2R+|V|\cdot 2r+|V|\cdot r$, where $|C|$ is the number of clauses and $|V|$ is the number of variables. If both T/F branches corresponding to the same variable need to be connected to the clause nodes, then one of the $2R$ terms is replaced with

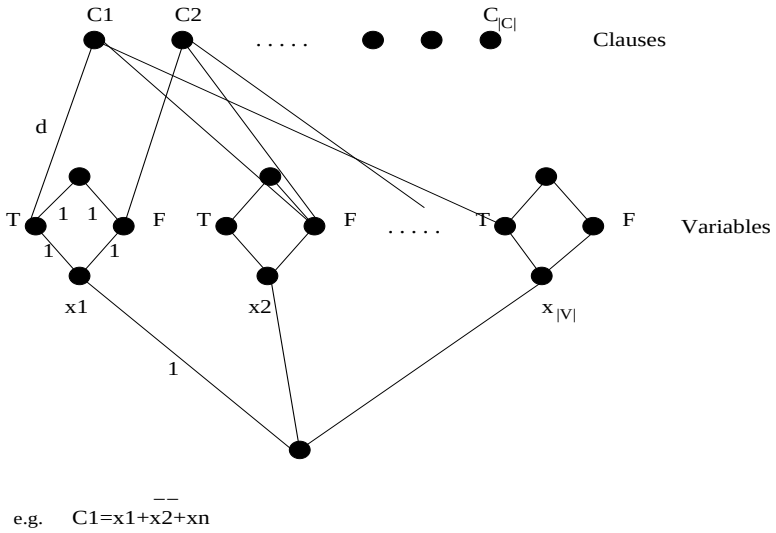


Figure 17: The reduction from any instance of 3-SAT to an instance of our problem.

a $2r$ term so the tree is no longer optimal. As the construction of the tree corresponding to the 3-SAT instance is polynomial, then our problem is at least as hard as 3-SAT, and thus NP-complete. ■

References

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A Survey on Sensor Networks. *IEEE Communications Magazine*, 40(8):102–116, 2002.

[2] B. Awerbuch, A. Baratz, and D. Peleg. Cost-Sensitive Analysis of Communication Protocols. In *Proceedings of the Ninth Annual ACM Symposium on Principles of Distributed Computing*, 1990.

[3] K. Bharat-Kumar and J. Jaffe. Routing to Multiple Destination in Computer Networks. *IEEE Trans. on Communications*, COM-31:343–351, 1983.

[4] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. Wiley Series in Telecommunications. John Wiley and Sons, Inc., 1991.

[5] R. Cristescu, B. Beferull-Lozano, and M. Vetterli. Networked Slepian-Wolf: Theory, Algorithms and Scaling Laws. *Submitted to IEEE Trans. on Inf. Th.*, 2003.

[6] R. Cristescu, B. Beferull-Lozano, and M. Vetterli. On Network Correlated Data Gathering. In *Proc. INFOCOM*, 2004.

[7] M.R. Garey and D.S Johnson. *Computers and Intractability*. W.H. Freeman, 1979.

[8] A. Goel and D. Estrin. Simultaneous Optimization for Concave Costs: Single Sink Aggregation or Single Source Buy-at-Bulk. In *ACM-SIAM Symposium on Discrete Algorithms*, 2003.

[9] B. Hajek. Cooling Schedules for Optimal Annealing. *Math. Oper. Res.*, (13):311–329, 1988.

[10] W. Rabiner Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In *Proc. of the 33rd International Conference on System Sciences (HICSS '00)*, January 2000.

[11] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed Diffusion for Wireless Sensor Networking. *IEEE/ACM Trans. on Networking*, 11(1), February 2003.

[12] S. Khuller, B. Raghavachari, and N. Young. Balancing Minimum Spanning and Shortest Path Trees. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 243–250, 1993.

[13] B. Krishnamachari, D. Estrin, and S. Wicker. Modelling Data Centric Routing in Wireless Sensor Networks. Technical Report 02-14, USC Computer Engineering, 2002.

[14] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys (ed.). *The Traveling Salesman Problem.* Wiley, 1990.

[15] S. Lindsey, C. S. Raghavendra, and K. Sivalingam. Data Gathering in Sensor Networks using the Energy*Delay Metric. In *Proc. of IPDPS Workshop on Issues in Wireless Networks and Mobile Computing*, April 2001.

[16] M. Lundy and A. Mees. Convergence of an Annealing Algorithm. *Mathematical Programming*, (34), 1986.

[17] C. Perkins. *Ad Hoc Networking.* Adison-Wesley Pub Co, 2000.

[18] G.J. Pottie and W.J. Kaiser. Wireless Integrated Sensor Networks. *Communications of the ACM*, 5(43):51–58, 2000.

[19] S. Pradhan and K. Ramchandran. Distributed Source Coding Using Syndromes (DISCUS): Design and Construction. In *Proc. IEEE Data Compression Conference*, March 1999.

[20] J. Rabaey, M.J. Ammer, J.L. da Silva, D. Patel, and S. Roundy. PicoRadio Supports Ad-Hoc Ultra-Low Power Wireless Networking. *IEEE Computer*, 33(7):42–48, 2000.

[21] C. Reidys and P. Stadler. Combinatorial Landscapes. *SIAM Review*, 44:3–54, 2002.

[22] D. Slepian and J.K. Wolf. Noiseless Coding of Correlated Information Sources. *IEEE Trans. Information Theory*, (IT-19):471–480, 1973.