Practical algorithms for gathering stored correlated data in a network

Ramin Khalili
Dept. of Computer Science
University of Massachusetts, Amherst, MA
Email: ramin@cs.umass.edu

Jim Kurose
Dept. of Computer Science
University of Massachusetts, Amherst, MA
Email: kurose@cs.umass.edu

***Abstract*—Many sensing systems remotely monitor/measure an environment at several sites, and then report these observations to a central site. We propose and investigate several practical algorithms for joint routing and compression of data files as they are forward from remote nodes to a central site, with the goal of minimizing the communication cost incurred. Our algorithms are practical in that they do not assume that nodes have *a priori* information about the correlation structure (and resulting compression gains) of the individual measurements at a given sensor or among multiple sensors. Instead, this correlation structure is learned as pieces of the files are routed and jointly compressed on their way to the sink, and routes are adaptively changed as the nodes learn more about the correlation structure of the data.**

I. Introduction

Increasingly, distributed sensing systems are being developed and deployed that monitor / measure an environment under study, and report these observations to a central site. Examples include in-situ monitoring of natural habitats [5], [12], [23], collaborative adaptive sensing of the atmosphere [3], multi-camera video surveillance [4], and measurement / monitoring of engineered systems such as transportation systems [2], and data networks [1]. Some of these systems operate in real-time, with individual measurements being forwarded to the central site as soon as they are taken. In other cases, a potentially large set of measurements are first stored at the remote sensor, and then forwarded in non-real-time as a group (e.g., as a file) to the central site. In either case - individual real-time measurements or bulk non-real-time measurements in files - data is routed among the sensors, and may be jointly compressed with locally stored data (which must also be transferred to the central site) as it is forwarded toward the central site.

In this paper, we propose and investigate several *practical* algorithms for joint routing and compression of data files as they are forward from remote nodes to a central site. The goal of these algorithms is to minimize the communication cost incurred in gathering these files at the central site. Such a problem has been studied in information theory literature such as [6], [7], [20], [22], that examined the distributed compression techniques, and [8], [14], [21] that examined the joint routing and compression techniques. However, the implementation of the proposed schemes in a practical setting is still an open problem since they require the complete knowledge of all source correlations *a priori* at each source. Our algorithms are practical in that they do not assume that a node has *a priori* information about the correlation structure (and resulting compression gains) of the individual measurements at a given sensor or among multiple sensors. Instead, this correlation structure can only be learned as data chunks are routed and jointly compressed on their way to the sink, with routes adaptively changing as the nodes learn more about the correlation structure of their data.

Our work thus differs from [8] which assumed that the correlation structure between data files is both known and identical for all pairs of nodes. It is also different from [14] which assumed that the joint pdf of the neighborhood sources data (*i.e.*, the source data correlation between local neighborhood nodes) is known. Moreover, in [19] the performance of routing with compression in wireless sensor networks is analyzed where the correlation structure between data sources has been empirically approximated as a function of distances between sources in the network. Our work is thus different, as in [19] the correlation structure between data sources is required to be estimated before starting the communication while our algorithms do not need such *a priori* information. Our work also differs from [7], [9]–[11], [16], [22] that establish information-theoretic bounds on the performance of any data gathering/compression algorithm under varying assumptions (e.g. a priori information about the correlation structure) but do not consider any specific routing or compression algorithms in particular.

With stored files, compression gains can arise from correlation in data gathered at a single given location (which we refer to as compression in time) and among measurements taken at ”nearby” sensors (correlation in space). As part of this study, we empirically evaluate the compressibility of two specific distributed data sources - in trace files of packets headers taken simultaneously at two measurement points in a university network, and in synthetic radar data files taken simultaneously at overlapping emulated radars operating in a numerically simulated atmosphere, using specific compression

This work was supported in part by the National Science Foundation under award number CNS-0519998 and under the Engineering Research Centers Program, Award number EEC-0313747. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the National Science Foundation.

978-1-4244-1777-3/08/$25.00 © IEEE

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE SECON 2008 proceedings.

algorithms.

Given the possibility of achieving lower data gathering costs by jointly compressing files from multiple sites, we present two techniques for iteratively estimating the joint compressibility of two files by dividing one of the files into a number of chunks, and iteratively sending these chunks from one node to the other. We find that it is possible to accurately estimate joint compressibility by transmitting only a small number of data chunks (i.e., a small amount of the file) from one site to another. This joint compressibility estimator is then embedded in two iterative local heuristic algorithms for joint routing and compression. We compare the performance of these heuristic algorithms to exact solutions (which are known to be NP-complete even in the case that the compressibility is known and identical at all nodes) for small size problems, and compare their performance with shortest path routing paths (which does not take the potential joint compression gains into account when determining the routing path, but does jointly compress files as they are routed along the SPT path). We find that for small problem sizes, the algorithms perform close to optimal, and for larger problem sizes can significantly decrease the cost of routing over shortest path routing.

The rest of this paper is structured as follows. In the following section we describe the network setting and assumptions for our study. In Section 3, we consider the simple case of a three-node network in which a single sink gathers data stored at two other nodes. We use this simple scenario to illustrate a polynomial fitting algorithm and a prediction algorithm for estimating joint compressibility of data "on the fly" as progressively more data chunks are received. We empirically study the joint compressibility of two specific data sources. The first source is network measurement data - an internet-scale sensing system [17], [18] in which network measurement devices gather data (e.g., traffic traces) of network traffic. The second source is meteorological radar data gathered from radars sensing the lower regions of the atmosphere [3]. We study the performance of the polynomial fitting and prediction algorithms for estimating the joint compressibility using both of these data sources. In section 4, we use the results from our three-node scenario as the basis for recursive heuristic algorithms for jointly routing and compressing data in larger network settings. Finally, Section 5 concludes this paper and discusses possible future research.

II. Network Setting

In this paper we consider a network that is not constrained in energy and thus the cost of compression in term of battery drain is negligible. We consider a network of N nodes connected via a set of edges $(i, j),\ i, j,\ i \neq j \in N$ in some arbitrary topology. Associated with each edge is a cost, $c_{i,j}$ for sending a unit of data over the edge from node i to node j. One of the N nodes is the designated as the destination or sink node. Each of the other $N-1$ nodes has a file, X_i, that must be routed to the destination node over the network. Let $|X_i|$ be the size of file X_i. If we apply a compression algorithm Com to file X_i, we create a compressed file $com(X_i)$ of size $Com(X_i) = |com(X_i)|$. In this paper we consider only the loss-less compression of data where the decrease from input rate to output rate (by compressing data) does not distort the transmitted data.

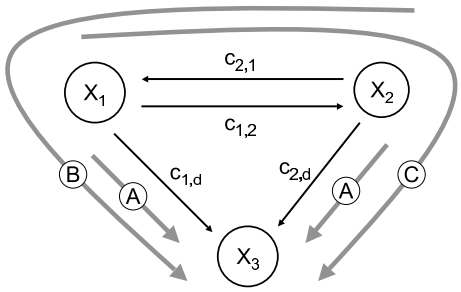


Fig. 1. Two sources scenario

Our goal will be to transfer (route) the $N-1$ data files to the destination at minimum cost. Costs are incurred as follows. If node i has only a single file, X_i, and compresses it and then sends it to direct neighbor j over link (i, j) then a cost $c_{i,j}Com(X_i)$ is incurred. If node j receives file X_i from node i (or more precisely, a compressed version of X_i which it can then decompress), and then jointly compresses X_i and its own file X_j and then sends the jointly compressed file to its neighbor, k, the incurred cost is $c_{j,k}Com(X_i, X_j)$, where $Com(X_i, X_j)$ is the size of the jointly compressed files X_i and X_j. We assume that $Com(X_i, X_j) \forall i, j, i \neq j$ is initially unknown and consider below algorithms for iteratively estimating $Com(X_i, X_j)$ as pieces of the file X_i are sent from i to j.

III. The Two Source Case

In order to gain some insight into the joint compression and routing problem with unknown correlation, let's first consider the simple two-source case shown in Figure 1. We assume that $Com(X_1)$ and $Com(X_2)$, the compressed sizes of files X_1 and X_2, are known by the nodes, since node i can locally calculate $Com(X_i)$ and send this information to its neighbor. As shown in Figure 1, there are three possible routing paths for transferring X_1 and X_2 to destination d :

Path A; In this case nodes 1 and 2 each transmit their locally compressed files $com(X_1)$ and $com(X_2)$, respectively, directly to the receiver. The corresponding transmission cost is equal to :

$$\mathcal{C}_A = c_{1,d}Com(X_1) + c_{2,d}Com(X_2). \quad (1)$$

Path B; In this case X_2 is first locally compressed at node 2 and sent to node 1 over link (2,1). Node 1 then jointly compresses X_1 and X_2 and sends $com(X_1, X_2)$ to the destination over link (1,d). The transmission cost is equal to :

$$\mathcal{C}_B = c_{2,1}Com(X_2) + c_{1,d}Com(X_1, X_2). \quad (2)$$

Path C; In this case, $com(X_1)$ is sent over link (1,2) to node 2, which then jointly compresses X_1 and X_2 and sends $com(X_1, X_2)$ to d over link (2,d). The transmission cost is

equal to :

$$\mathcal{C}_C = c_{1,2}Com(X_1) + c_{2,d}Com(X_1, X_2). \quad (3)$$

The optimal routing path is the path with minimum transmission cost. Therefore, for the simple scenario of two sources nodes (illustrated in Figure 1) the problem is simplified to a linear optimization problem and the optimal routing depends on the value of $Com(X_1, X_2)$, which is initially unknown.

Let

$$\rho^{'} = \frac{c_{1,d}Com(X_1) + (c_{2,d} - c_{2,1})Com(X_2)}{c_{1,d}}$$

and

$$\rho^{''} = \frac{c_{2,d}Com(X_2) + (c_{1,d} - c_{1,2})Com(X_1)}{c_{2,d}}.$$

$\rho^{'}$ is the value of $Com(X_1, X_2)$ when the cost of routing along paths A and B (equations 1 and 2) are equal. $\rho^{''}$ is the value of $Com(X_1, X_2)$ when the cost of routing along paths B and C are equal. Some simple manipulations show that if $Com(X_1, X_2) > \max\{\rho^{'}, \rho^{''}\}$ then A is the optimal path. Otherwise, there are sufficient joint compression gains, and the links costs are such that it is better to incur the cost of transmitting a file from node 1 to node 2 (or vice versa), and then jointly compressing X_1 and X_2 and sending the jointly compressed file to d over the direct link.

Without loss of generality, assume that B is the path corresponding to $\max\{\rho^{'}, \rho^{''}\}$, i.e., $\rho^{'} > \rho^{''}$. Let us define

$$\rho_{BC} = \frac{c_{1,2}Com(X_1) - c_{2,1}Com(X_2)}{c_{1,d} + c_{2,d}}.$$

If $\rho_{BC} < Com(X_1, X_2) < \rho^{'}$, B is the optimum path; otherwise C is the optimum path, see Figure 2. Note that both ρ_{BC} and $\rho^{'}$ can be easily calculated by the nodes since neither involves $Com(X_1, X_2)$. The challenge in evaluating the condition, $\rho_{BC} < Com(X_1, X_2) < \rho^{'}$, is that the size of the jointly compressed file, $Com(X_1, X_2)$, is unknown. Since we make no assumptions about *a priori* information about how the data in files X_1 and X_2 are correlated, the size of the jointly compressed files, X_1 and X_2, will have to be estimated. The following algorithm (for the simple two-node, single-destination scenario shown in Figure 1) chooses the optimal path by iteratively estimating a value for $Com(X_1, X_2)$. The algorithm divides the file at node 2 into M chunks and iteratively sends these chunks to node 1, until node 1 is able to form a sufficiently accurate estimate of $Com(X_1, X_2)$ to determine the optimal routing. We first present the algorithm, and then discuss its three main steps.

Algorithm 2SJRC: Two-Source Joint Routing and Compression

1) Compute the shortest path tree (SPT) for the network illustrated in Figure 1. If the SPT corresponds to routing scenarios B or C, transmit data to the destination using the routing defined by the SPT, jointly compressing the data at the intermediate node. Go to the End.

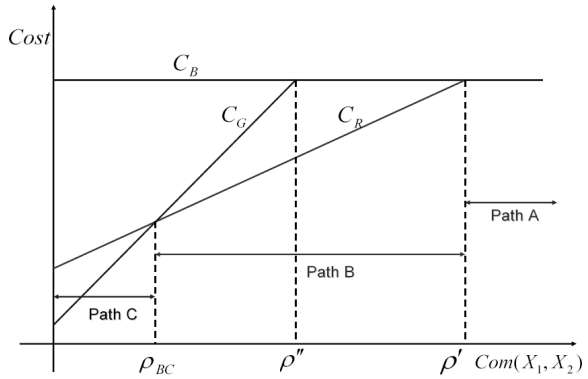


Fig. 2. The linear optimization problem

2) Divide X_2 into M partitions $X_2 = \{X_2^1, \cdots, X_2^M\}$. Set $i = 1$.
While $i \leq M$.
Node 2 sends the i^{th} partition of file X_2, $com(X_2^i)$, to node 1. Node 1 then estimates $Com(X_1, X_2)$ based on $Com(X_1)$ and the partitions $\{Com(X_2^j); j = 1 : i\}$ that have been received from node 2 so far. Let $\hat{\rho}_i$ denote node 1's estimated value for $Com(X_1, X_2)$ at step i. If the corresponding estimation error (ϵ_i) is lower than a required threshold (ϵ_T) go to 3; otherwise increase i by one. We will shortly explain how $\hat{\rho}_i$ and ϵ_i can be calculated for a set of data.
End while.
3) If $\rho_{BC} \leq \hat{\rho}_i \leq \rho^{'}$ go to (a), If $\hat{\rho}_i > \rho^{'}$ go to (b), and if $\hat{\rho}_i \leq \rho_{BC}$ go to (c). In the substeps below, A, B, and C refer to the three different possible routings shown in Figure 1.

a) B is the optimal routing. By "optimal routing" here, we mean that configuration B minimizes the costs among routing configurations A, B, and C, defined in equations 1, 2 and 3. In this case, send $com(X_2^{i+1}, \cdots, X_2^M)$ from S_2 to S_1 and $com(X_1, X_2)$ from S_1 to the destination. Note that $Com(X_2) \leq \sum_{j=1}^{M} Com(X_2^j)$. Go to the End.

b) A is the optimal routing. Let us denote the prefix of X_2 consisting of the first i chunks of X_2 as$\mathcal{X}_2^i = \{X_2^1, \cdots, X_2^i\}$. ***If*** $c_{2,d}Com(X_2^{i+1}, \cdots, X_2^M) + c_{1,d}Com(X_1, \mathcal{X}_2^i) \leq c_{1,d}Com(X_1) + c_{2,d}Com(X_2)$ we send $com(X_1, \mathcal{X}_2^i)$ from node 1 to the destination, and send the remainder of the file, $com(X_2^{i+1}, \cdots, X_2^M)$, which is at node 2 but has not yet been sent to node 1 directly from node 2 to the destination via link (2,d). ***Otherwise***, we send the entire file $com(X_1)$ from node 1 directly to the destination, and the entire file $com(X_2)$ from node 2 to the destination. Go to the End.

c) C is the optimal routing. ***If*** $c_{2,1}Com(X_2^{i+1}, \cdots, X_2^M) + c_{1,d}Com(X_1, X_2) \leq c_{1,2}Com(X_1) + c_{2,d}Com(X_1, X_2)$ we still continue

to send over B by sending $com(X_2^{i+1}, \cdots, X_2^M)$ from node 2 to node 1 and $com(X_1, X_2)$ from node 1 to the destination. ***Otherwise*** we transmit over path C by sending $com(X_1)$ from node 1 to node 2 and $com(X_1, X_2)$ from node 2 to the destination. Go to the End.

4) End of Algorithm 2SJRC

In the first step of the algorithm, we compute the shortest (least cost) path tree, e.g., using Dijkstra's algorithm, for the network shown in Figure 1. Since the SPT provides the least cost paths (on the basis of link weights alone, not taking into account the size of compressed data files) from each node to the destination, then if SPT path corresponds to routing scenario B, that routing scenario also has less cost than routing scenario A when taking compressed file sizes into account when determining cost in equations 1 - 3. This is true since $c_{2,1}+c_{1,d} \leq c_{2,d}$ and $Com(X_1, X_2) \leq Com(X_1)+Com(X_2)$ and thus $\mathcal{C}_B \leq \mathcal{C}_A$. Let us next compare $\mathcal{C}_B$ and $\mathcal{C}_C$. We have:

$$\begin{aligned} \mathcal{C}_B &= c_{2,1}Com(X_2) + c_{1,d}Com(X_1, X_2) \\ &\overset{a}{\leq} c_{2,1}Com(X_1, X_2) + c_{1,d}Com(X_1, X_2) \\ &\overset{b}{\leq} c_{2,d}Com(X_1, X_2) \\ &\leq c_{1,2}Com(X_1) + c_{2,d}Com(X_1, X_2) = \mathcal{C}_C \end{aligned}$$

where (a) comes from the fact that $Com(X_2) \leq Com(X_1, X_2)$ and (b) follows from $c_{2,1} + c_{1,d} \leq c_{2,d}$. Therefore B is the optimal routing scenario in the sense of minimizing the costs among configurations A, B ad C, as specified in equations 1 - 3. Following similar arguments, if the SPT corresponds to the routing scenario C then C would be the optimal routing scenario. Thus in the situation that the SPT tree (which is based only on link costs, and does not consider compressed file sizes) corresponds to routing scenarios B or C, the SPT is also the optimal routing for the costs specified in equations 1 through 3 for $\mathcal{C}_A, \mathcal{C}_B$ and $\mathcal{C}_C$.

In the second step of the algorithm, we iteratively estimate $\hat{\rho}_i$, the size of jointly compressed files X_1 and X_2 using one of the two estimation algorithms discussed below.

By step 3, we have an estimated value of $\hat{\rho}_i$, having already sent i chunks from node 2 to node 1. In step 3a, the optimal routing is B (i.e., node 2 routing to the destination via node 1). In this case, node 2 compresses the remaining chunks of its file and sends this over link (2,1) to node 1, which then sends the jointly compressed file $com(X_1, X_2)$ to the destination over link $(1, d)$. Step 3b is the case that the optimal path is configuration A, where nodes 1 and 2 send directly to the destination. Note, however, that node 2 has already sent the prefix $\mathcal{X}_2^i$ to node 1, and so it may now be more cost effective for node 1 to jointly compress the prefix with its own local file, X_1 and send the jointly compressed quantity $com(X_1, \mathcal{X}_2^i)$ to the destination, rather than having nodes 1 and 2 send their entire files directly to the destination.

Step 3c is the case that C is the optimal routing path. Once again, however, node 1 has already received the prefix $\mathcal{X}_2^i$. We thus test whether it may be advantageous for node 1 to make use of the prefix (given that is has already incurred the cost of receiving this prefix from node 2 as part of the estimation process for $\hat{\rho}_i$).

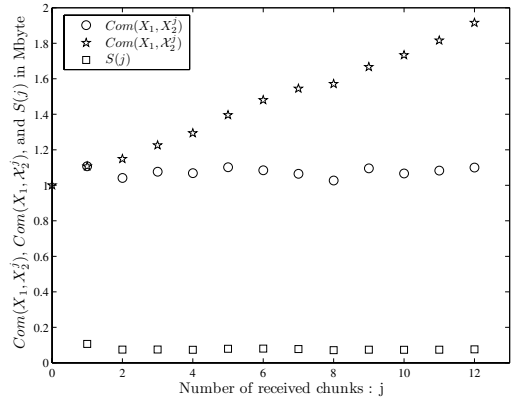


Fig. 3. $Com(X_1, \mathcal{X}_2^i)$, $Com(X_1, X_2^j)$, and $S(i)$ for $i \in \{1, 2, \ldots, 12\}$.

Let us now turn our attention to the problem of estimating $\hat{\rho}_i$, the estimated value for $Com(X_1, X_2)$ at step i, and how ϵ_i (the error associated with the estimated value) can be calculated.

Estimation of $\hat{\rho}_i$ with a Fitting approach

Recall that our goal is to estimate $Com(X_1, X_2)$ at node 1, having received $\mathcal{X}_2^i$, i chunks of the prefix of X_2. Consider the sequence of compressed file sizes $\{Com(X_1, \mathcal{X}_2^1), Com(X_1, \mathcal{X}_2^2), \ldots Com(X_1, \mathcal{X}_2^i)\}$, for prefix sizes of $1, 2, \ldots, i$. We define $Com(X_1, X_2^0 = \phi) = Com(X_1)$. The fitting approach we present to estimate $\hat{\rho}_i$ fits polynomial, $P(x)$ of degree n at these $i+1$ points in a least squares sense [13]. $\hat{\rho}_i$ is then taken as the value of the fitted polynomial at point $k = M$, i.e., $\hat{\rho}_i = P(x = M)$. The estimation error, ϵ_i, is calculated as the mean-square value of the distance between the computed values $\{Com(X_1, \mathcal{X}_2^j);\ j = 0 : i\}$ and the fitted polynomial divided by $\hat{\rho}_i$, *i.e.,* $\epsilon_i = \frac{1}{i\hat{\rho}_i} \sum_{j=1}^{i} \left| P(x = j) - Com(X_1, \mathcal{X}_2^j) \right|^2$. In this paper we consider that $n = 1$, however, higher degree can be used as well.

To illustrate the fitting approach, we show here the proceeding steps for two files containing a traces of packet headers, X_1 and X_2, with compression sizes $Com(X_1) = 1$ Mbytes and $Com(X_2) = 2$ Mbyte. These two trace files were collected simultaneously at a university-wide and a departmental gateway in a large university network. We expect there to be some correlation in these contents of these traces, since packets sourced in the departmental network and destined to the larger Internet may pass through the gateway and thus appear in both the departmental trace as well as the gateway trace. However, since the university has connections into the Internet via multiple providers, an Internet-bound packet from the department may not appear in the gateway trace.

We divide X_2 into 12 chunks in such a way that $Com(X_2^j) = \frac{2}{12} = 0.1667$ Mbyte for $j \in \{1, 12\}$. In Figure 3, the starred values plot the sequence of compressed file sizes

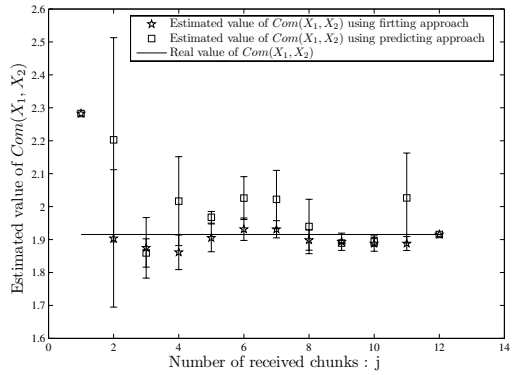


Fig. 4. Comparing the estimated value of $Com(X_1, X_2)$ using the fitting and predicting algorithms.

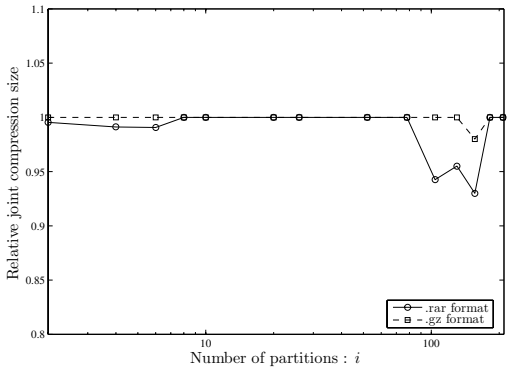


Fig. 5. Joint compression size for two individual radar trace files.

$\{Com(X_1), Com(X_1, \mathcal{X}_2^1), \ldots Com(X_1, \mathcal{X}_2^{12})\}$. The trend illustrated in Figure 3 is clearly linear. As explained before, in step i we have the first $i+1$ points and the fitting approach in this step consists of fitting a first degree polynomial $P(x)$ to these points in a least square sense. For example for $i = 7$ the polynomial fitted to points $\{Com(X_1), Com(X_1, \mathcal{X}_2^1), \cdots, Com(X_1, \mathcal{X}_2^7)\}$ has the form $P(k) = 0.077k + 1.004$. $\hat{\rho}_7 = 1.931$ Mbyte is the polynomial value at point $k = 12$ and $\epsilon_7 = \frac{1}{7\rho_7} \sum_{j=1}^{7} |P(k = j) - Com(X_1, \mathcal{X}_2^j)|^2 = 0.0174$ is the error associated with this estimation. In Figure 4 we show $\hat{\rho}_i$ and the norm of its associated residual error $\hat{\rho}_i \times \epsilon_i$ for $i \in \{1, 12\}$.

Estimation of $\hat{\rho}_i$ with a Predicting approach

In the second approach we estimate $Com(X_1, X_2)$ by predicting the average value of $Com(X_2|X_1)$ based on the sequence $Com(X_2^j|X_1)$, $j \in \{1, i\}$, where $Com(X_2^j|X_1) \approx Com(X_j^2, X_1) - Com(X_1)$. For this purpose we define the random process S as $S(i) = \frac{1}{i} \sum_{j=1}^{i} Com(X_2^j|X_1)$. Therefore $Com(X_1, X_2)$ is bounded by $Com(X_1) + M.S(M)$. This comes from the fact that $Com(X_1, X_2) \approx Com(X_1) + Com(X_2|X_1)$ and $Com(X_2|X_1) \leq \sum_{k=1}^{M} Com(X_2^k|X_1)$.

As S is a stationary process, classical prediction algorithms can be used to predict $S(M)$ from the time series of $\{S(1), S(2), \cdots, S(i)\}$ and the value of $\hat{\rho}_i$ is taken to be $Com(X_1) + M\tilde{S}_i(M)$, where $\tilde{S}_i(M)$ is the predicted value for $S(M)$ at step i. In this paper we use the stochastic least-mean-square (LMS) estimator which is a stochastic counterpart of the least-square-estimator used in the fitting approach (interested readers are encouraged to see [13] for more details about LMS estimators).

In Figure 3 we show the sequence of compressed file sizes $Com(X_1, X_2^1), Com(X_1, X_2^2), \ldots Com(X_1, X_2^{12})$ and the time sequence $S(i)$ for the two packet header trace files described earlier. At any step i, the series $S(1)$, $S(2)$, ..., $S(i)$ is used to predict $S(M)$ and consequently $\hat{\rho}_i$. For example if $i = 7$ we have $\tilde{S}_i(M) = 0.0852$ and therefore $\hat{\rho}_7 = 1 + 12 \times 0.0852 = 2.022$ with an associated error ϵ_7 bounded by 0.0426. The values of $\hat{\rho}_i$ and $\hat{\rho}_i \epsilon_i$, for $i \in \{1, 12\}$, are presented in Figure 4.

A. Evaluating Algorithm 2SJRC with fitting and predicting with real data.

In the previous section, we presented the 2SJRC algorithm for optimally routing data files from two sources to a common sink. At the heart of this algorithm was a process for iteratively estimating the joint compression size of these two files; we presented both a fitting approach and a predicting approach for estimating $\hat{\rho}_i$. We illustrated the operation of the algorithm using two packet header trace files. In this section, we perform a more extensive study of 2SJRC and its fitting and prediction approaches for estimating $\hat{\rho}_i$. We consider two different types of correlated data.

Our first source of spatially and temporally correlated data is radar data, which contains the reflectivity (a measure of the amount of scattering matter) in small unit volumes ("voxels") in the atmosphere over a period of time. These files were made available to us by the Center for Collaborative Adaptive Sensing of the Atmosphere [3]. Specifically, we consider the reflectivity data gathered at two radars with partially overlapping footprints. The radars themselves are emulated, operating in a numerically simulated atmosphere [15].

We choose the reflectivity trace files of two neighboring radar nodes, X_1 and X_2, collected during identical intervals of time with the respective size of $|X_1| = 455$ Kbyte and $|X_2| = 505$ Kbyte. We first apply the .gzip compression algorithm to compress each individual file and also jointly compress X_1 and X_2. The results show that using a gzip compression algorithm, we will not gain from the joint correlation between the data in the two files, as $Com(X_1) = 363$ Kbyte, $Com(X_2) = 253$ Kbyte, and $Com(X_1, X_2) = 616$ Kbyte, *i.e.* $Com(X_1, X_2) = Com(X_1) + Com(X_2)$. We also examined the reflectivity trace files collected during other time intervals and found similar conclusions - minimal or no gain was obtained using joint compression over the case of local compression. We then considered different compression algorithms (including zip, rar, and LZ7) to see if one of these compression algorithms might be able to gain from the joint correlation structure, but found similar results. For example for the trace file described above and using .rar compression, we have $Com(X_1) = 259$ Kbyte, $Com(X_2) = 179$ Kbyte, and $Com(X_1, X_2) = 438$ Kbyte.

We conjecture that the compression algorithms we have studied achieve significant local compression gains because the reflectivity data at an individual radar already has significant correlation in both time and space - an individual radar measures reflectivity in neighboring voxels (correlation in space) and sweeps through the same set of voxels over time (correlation in time). Jointly compressing files from two sites offers the possibility of taking advantage of correlated data due to overlap in the radar footprints (both radars are observing the exact same voxel in space), or proximity of the measured voxels (the voxels are different, but are physically close to each other). Joint compression gains clearly *are* possible; for example, the jointly compressed size of 616 Kbyte is significantly smaller than the sum of the two original file sizes. However, the joint compression of radar reflectivity data from two partially overlapping radars using generic compression algorithms does not appear to offer any significant advantage over simple local compression of the files.

Several generic compression algorithms, such as .rar and .gzip, have small coding memories and thus are not able to track the joint correlation structure in large-sized data sets. To see if we could overcome this drawback, we divided the trace files into smaller chunks and jointly compressed these smaller chunk files. We then computed $g^i = \{\frac{Com(X_1^j, X_2^j)}{Com(X_1^j)+Com(X_2^j)}, j \in \{1, i\}\}$ as a metric indicating the relative joint compression size of the chunked files, when files are divided into i partitions. Figure 5 plots the values of g^i as a function of the number of partitions, i, using rar and gzip formats (we performed the same process for gzip and LZ7, but since the results lay between the cases of .rar and .gzip, we do not discuss them here). Note that the x-axis is in logarithmic scale. We see that there is indeed a small gain of jointly compressing data (using .rar format) when we divide the files into small chunks (i.e., a large number of partitions), meaning that for smaller chunk sizes this compression algorithm is able to track and exploit the correlation between data. However, we also see that for very small partition sizes (e.g., $i > 160$) the gain again disappears. While we believe it may indeed be possible to develop application-specific compression algorithms for exploiting the joint correlation among radar reflectivity estimates stored at different locations - algorithms that may offer additional gains over simple local compression - we have found that generic, application-oblivious compression algorithms have not been able to realize this additional gain, i.e. "having" a priori information about the overlapping area between radar nodes will not provide any useful knowledge about the achievable gain of jointly compressing data. While the development of application-specific compression techniques is clearly of interest, it is well beyond the scope of this paper. Because of the limited or non-existent additional joint compression gains achievable with radar reflectivity data, we did not pursue an analysis of 2SJRC or the fitting and predicting approaches with this type of data.

Our second source of spatially and temporally correlated data is are trace files of packet headers captured during identical time intervals in a university network. One trace file (X_1) was taken at one of the universities two main gateways to the public Internet; the second trace file (X_2) was taken at the gateway to the Computer Science department. We apply different compression algorithms to examine the gain of jointly compressing X_1 and X_2. The results show that we will not gain from the joint correlation between the data in the two files using generic compression algorithms. We also computed g^i as depicted in figure 6. We see that there is a small gain of jointly compressing data when we divide the files into a large number of partitions however this gain is not considerable.

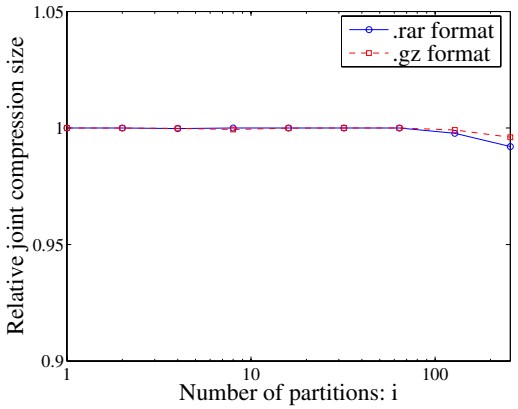


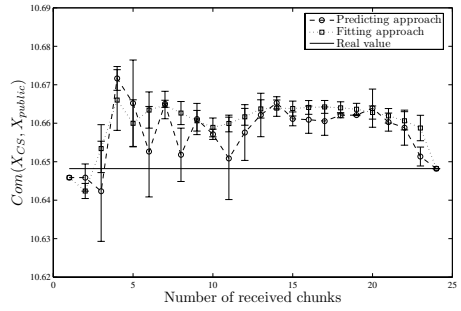
Fig. 6. Joint compression size for two individual header trace files.

Note that any packet that exits the CS gateway and enters the public Internet via the monitored university gateway (or vice versa) will result in a packet header that is essentially identical, except for a timestamp, being recorded in both locations. Specifically, the recorded packet headers at the CS gateway and university gateway will only differ in the hop count and IP header checksum, and thus differ in a known and deterministic (and hence reconstructable) manner. For example; from 4-hour long packets traces taken over several days during July 2006, we observed that the fraction of redundant traffic recorded in the department and university traces files, which we will refer to by parameter α, over the 4 hour period was approximately 0.16. At finer time scales the amount of redundant traffic varied, generally remaining within 25% of α, as discussed below.

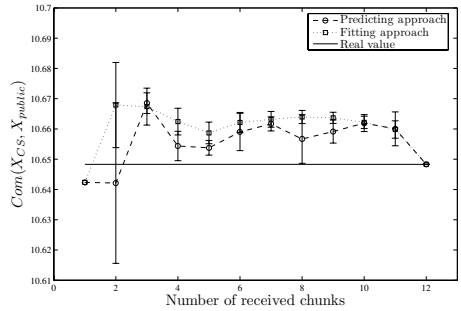
Given that a fraction α of the packet headers in the two trace files are redundant (and can thus be removed and replaced with a small has value for the packet header), and that there are *no* redundant packets within a single local trace file (since a given packet never passes the same measurement point twice), we approximate the relationship between the jointly compressed file size and the individually compressed file sizes as:

$$Com(X_1, X_2) = Com(X_1) + (1 - \alpha)Com(X_2)$$

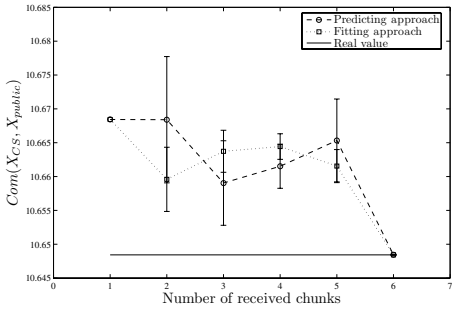
That is, when jointly compressed, the redundant packet headers can be removed, yielding compression gains above and beyond what is achieved by individual compression alone. The expression above should be valid regardless of the particular



(a) M=24

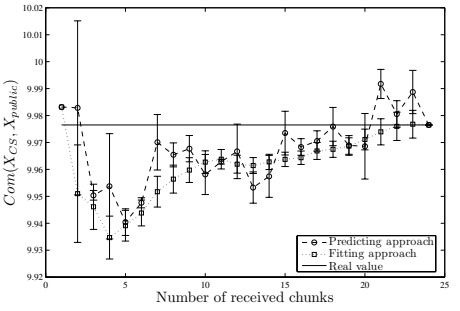


(b) M=12

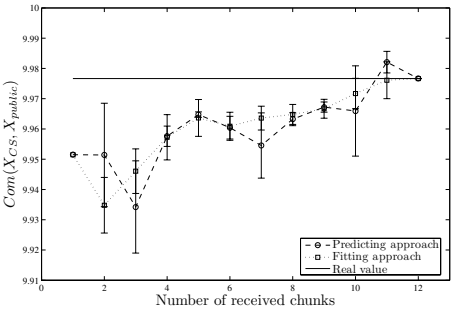


(c) M=6

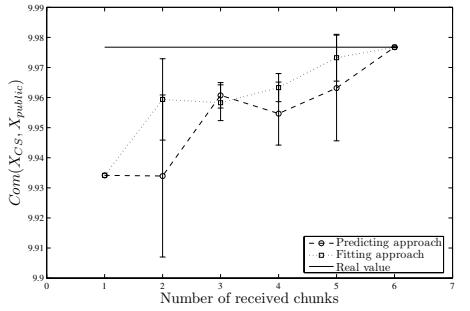
Fig. 7. Applying the predicting and fitting approaches for collected trace files at 12.07



(a) M=24



(b) M=12



(c) M=6

Fig. 8. Applying the predicting and fitting approaches for collected trace files at 24.07

compression algorithm used. Thus, in our numerical examples below, we used the uncompressed file sizes X_1 and X_2 for $Com(X_1)$ and $Com(X_2)$, respectively, but use our measured value for α for the file as a whole, and measured values for α_i for file chunk i in our analysis below. Note that α_i is not known *a priori* at nodes but only after receiving file chunk i. For the trace files collected at 12.07.2006, we have $Com(X_1) = 9.2$ Gbyte and $Com(X_2) = 1.73$ Gbyte and also files collected at 24.07.2006 with $Com(X_1) = 8.3$ Gbyte and $Com(X_2) = 1.997$ Gbyte.

We divide the department trace file into M chunks and we compute the sequence of compressed file sizes $\{Com(X_1, \mathcal{X}_2^j); j = 0 : M\}$ and the time series $\{S(1), S(2), \cdots, S(M)\}$ where the empirically observed values of α_i are used to determine the joint compressibility/redundancy in the ith block of packet header trace data. These sequence values are then used to estimate $\hat{\rho}_i$ in the fitting and predicting approaches. The evaluation results are shown in figures 7 and 8 for the trace files collected at 12.07.2006 and 24.07.2006 for different chunk sizes of $M = 6, 12, 24$. Note that after receiving the first partition we can estimate $\hat{\rho}_i$ with an error not higher than 0.5%. This confirms in some way our partitioning assumption, since a single partition of X_2 carry enough information for estimating the correlation structure between these two separated source data. In practice, we make our decision when the estimation error (ϵ_i) is lower than a required threshold ϵ_T. For example we can successfully estimate $\hat{\rho}_i$ with an estimation error of less than 0.1% ($\epsilon_T = 0.1\%$) if

we receive at the university gateway about 30% of the data measured at the department gateway. Therefore, in the worst case the cost of applying the step 2 of algorithm 2SJRC is not higher than 30% of the cost of transmitting X_2 through $(2,1)$ link. This property will be later used in evaluating our heuristics in the general case of larger scale networks. Note also that the predicting and the fitting approaches show equivalent performance.

IV. General case

In this section we consider the general scenario of a network with N data sources. As shown in [8], even with a known, identical correlation structure among all nodes, the optimal joint routing and compression problem is NP-hard. The problem becomes even more complex when the correlation structure is unknown by the sources. In this section, we thus propose and evaluate two approximate heuristic algorithms for solving the joint compression and routing problem in the general network setting. Our algorithms are extensions of the two-source, single-destination 2SJRC algorithm we examined in section 3, used together with the idea of recursive "leaf deletion" first proposed in [8]. An important difference between for N-source Joint Routing and Compression (NSJRC) algorithms we propose here and that in [8], is that [8] assumes that the correlation structure is both known *a priori* and is the same for any pair of network nodes. As in section 3, we assume that the correlation structure is unknown, can be different between any pairs of nodes, and will thus be estimated on the fly by the NSJRC algorithms, similar to how this was accomplished in our two-source 2SJRC algorithm.

We begin by having all nodes perform an SPT calculation over the network, e.g., using the Bellman-Ford algorithm. Each node thus knows the cost of the best path from any other node to the destination via this SPT. We'll also assume that each node i knows the communication costs, $\{c_{ij}\}$ for all of its links to all directly-attached neighbors j. A node i can thus compute the cost of sending $com(X_i)$ directly to the distination via the SPT path, a cost we will approximate as $c_{id}Com(X_i)$, where c_{id} is the sum of the link costs along the SPT path from i to d. Note that this cost is approximate as it assumes that no joint compression gains will be realized on the SPT path from i to d. Node i can also compute the cost of sending its file to neighboring node j, jointly compressing files X_i and X_j at j and then forwarding the compressed files from j to d as $c_{ij}Com(X_i) + c_{i,d}Com(X_i, X_j)$.

Following similar arguments as in the previous section, it is preferable for node i to route through node j if $c_{ij}Com(X_i) + c_{jd}Com(X_i, X_j) \leq c_{id}Com(X_i) + c_{jd}Com(X_j)$ or

$$\rho_{ij} \leq \frac{(c_{id} - c_{ij})Com(X_i) + c_{jd}Com(X_j)}{c_{id}} \qquad (4)$$

where $\rho_{ij} = Com(X_i, X_j)$. We refer to ρ_{ij} as the *joint compression threshold* for routing from i via j, since as long as the jointly compressed file size satisfies this threshold, we estimate that it is preferable for i to route to d via j.

We define T_i, the maximum joint compression threshold for node i as the maximum value of ρ_{ij} for all nodes j that are directly attached neighbors of i and also a leaf node in the SPT. We denote the neighbor j for which this joint compression threshold is maximized as N_i. Let also $par(i)$ denotes parent node of i in the SPT and $\mathcal{L}$ denotes the set of initial leaf nodes of the SPT. With this notation, we can now describe the NSJRC1 algorithm. We first present the algorithm and then discuss its operation.

Algorithm NSJRC1

- Initialize by computing the SPT. Find $\mathcal{L}$ for the SPT.
- **While** $\mathcal{L}$ is not empty :
  - For each leaf node $i \in \mathcal{L}$; find T_i, N_i, and $par(i)$. Distribute T_i to all other leaves in the SPT (we will see shortly that this information can also be distributed locally, allowing some of the computations below to proceed in parallel). Find the leaf node k that has the largest value, $T_k = \max_{i \in L} T_i$.
  - Divide X_k into M partition $X_k = \{X_k^1, \cdots, X_k^M\}$. Set $j = 1$.
    While $j \leq M$.
    Send j^{th} partition ($com(X_k^j)$) to N_k and find $\hat{\rho}_j$ (the estimated value for $Com(X_{N_k}, X_k)$ at step j) and ϵ_j (the error associated to this estimation). If ϵ_j is larger than the required threshold ϵ_T increase j by one; else go to the End while.
    End while.
  - If $\hat{\rho}_j \leq T_k$ route X_k to N_k; otherwise route X_k to $par(k)$.
  - Recompute a "trimmed" SPT by removing node k and link $(k, par(k))$ from the tree. (Note that trimming does not require recalculation of the SPT, but only a removal of one leaf node and one link.) Remove k from $\mathcal{L}$ as well. If $par(k)$ becomes a leaf node once the link $(k, par(k))$ is removed, add $par(k)$ to the leaf set $\mathcal{L}$.
- **End**.
- End of Algorithm NSJRC1.

The key idea behind the heuristic algorithm can be explained by first examining equation 4. Here, ρ_{ij} is the threshold value for the unknown quantity $Com(X_i, X_j)$ at which routing from i to j, jointly compressing X_i and X_j and then sending the jointly compressed files to d is estimated to be more cost effective than sending the two files directly from i and j with no joint compression. By taking the maximum over all neighbors, node i makes this threshold as less stringent a requirement as possible (i.e., allowing $Com(X_i, X_j)$ as large as possible).

We also note that the step of distributing T_i among all leaf nodes can be replaced by a local computation where each node distributes its value of T_i to its directly attached neighbors. A node receiving values of T_j from its neighbors can compare its own value of T_i with the received values of T_j and select itself for trimming if its value is larger than all received values. As

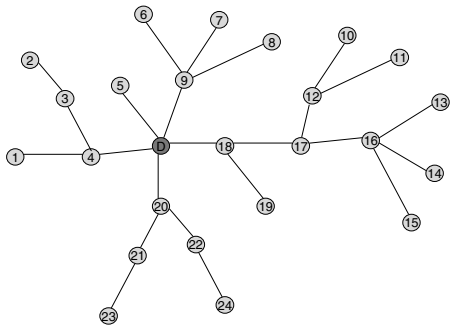


Fig. 9. Split parents, split children and leaf nodes in a SPT

the length of the paths over the SPT are typically less that 5, this algorithm involves at most $3-4$ supplementary iterations after the SPT is computed. Also note that this algorithm is fully distributed.

The following algorithm is a simplified version of algorithm NSJRC1 with less iterations. We once again build the SPT for a network of N nodes and each node again maintain its best path to the destination. Let's denote by $\mathcal{S}_p$ the set of nodes corresponding to splitting points in the SPT tree. We also define $\mathcal{S}_c$ as the set of nodes that are the direct children of nodes in $\mathcal{S}_p$. $\mathcal{L}$ is the set of leaf nodes in SPT.

For example for the SPT illustrated in figure 9; $\mathcal{L} = \{1,2,6,7,8,10,11,13,14,15,19,23,24\}$, $\mathcal{S}_c = \{1,3,5,6,7,8,9,12,16,17,18,20\}$, and $\mathcal{S}_p = \{4,9,12,16,17,18,20,d\}$. The main idea is to apply algorithm NSJRC1 just to nodes in $\mathcal{S}_c$; nodes that are in $\mathcal{L}$ but not in $\mathcal{S}_c$ will be routed over the SPT.

Algorithm NSJRC2

- Compute the initial SPT. Find $\mathcal{L}$, $\mathcal{S}_p$, and $\mathcal{S}_c$ for the built SPT.
- **While** $\mathcal{S}_c$ is not empty :
  - **While** there exist i in which $i \in \mathcal{L}$ but not in $\mathcal{S}_c$
    Route X_i to $par(i)$ and compute a trimmed SPT by removing node i and link $(i, par(i))$. Find $\mathcal{L}$ for the trimmed SPT.
  - **End**.
  - For each $j \in \mathcal{S}_c$; find T_j, N_j, and $par(j)$. Choose the node with maximum compression threshold (let's say node k) as the best leaf node to be trimmed from the SPT. Divide X_k into M partitions $X_k = \{X_k^1, \cdots, X_k^M\}$. Set $h = 1$.
    While $h \leq M$.
    Send h^{th} partition ($com(X_k^h)$) to N_k and find $\hat{\rho_h}$ and ϵ_h. If $\epsilon_h > \epsilon_T$ increase h by one; else go to the End while.
    End while.
  - If $\hat{\rho_h} \leq T_k$ rout X_k to N_k; else route X_k to $par(k)$. Trim the SPT by removing node k and link $(k, par(k))$. Find $\mathcal{S}_c$, $\mathcal{S}_p$, and $\mathcal{L}$ for the refreshed SPT.
- **End while**.
- End of Algorithm NSJRC2.

A. Comparing NSJRC1, NSJRC2, SPT and an optimal algorithm

In this section we first compare the performance of the NSJRC1 and NSJRC2 heuristic algorithms with the optimum algorithm for small problem sizes. We then investigate the performance of our algorithms for larger scale networks. We perform this comparison over a series of simulated network topologies, with synthetically generated link costs, and correlation structure, as discussed below.

We generate network scenarios as follows. We generate a network of N source nodes randomly (uniformly) distributed in a $D \times D$ square grid with a unique destination node in the middle point of the square. Each of the N source nodes in the network has a file of unit size (after local compression) to send to the destination. The cost of a link connecting two nodes is a random function of distance between the two nodes; there is no link between nodes at a distance larger than $100m$. Specifically, the link cost between two nodes is taken to be the distance between the two nodes divided by 100 multiplied by a random variable whose value is uniformly distributed in [0,1]. If a generated graph is not connected, it is discarded and another graph is generated.

The correlation between data sources is also chosen randomly based on the distances between nodes, with correlation decreasing with increasing distance. Specifically, the degree of correlation between two files at nodes i and j is taken to be $1 - \min\{\frac{distance(i,j)}{100}, 1\}$ multiplied by a random variable whose value is uniformly distributed in [0,1]. If two data files, X_i and X_j have a degree of correlation $z_{i,j}$, then $Com(X_i, X_j) = Com(X_i) + (1 - z_{i,j})Com(X_j)$ (note that $z_{i,j} = z_{j,i}$). This model is relative to the specific compression algorithm proposed in section 3.1 for collected trace data at the university and department gateways.

In our simulations, when node i determines that there is a potential gain in sending a file to node j (for joint compression of X_i and X_j) if the value of $Com(X_i, X_j)$ is sufficiently small, we assume that it must send 30% of its file to node j in order to make the decision of whether or not it indeed the correct choice to send the file to the destination via i. We chose this value of 30 % based on our analysis in Section 3 that showed that generally a node can accurately estimate $\hat{\rho_i}$ by the time 30 % of the file has been sent from one node to another. Whether or not this turns out to be a good decision will depend, of course, on the correlation structure of the data, as discussed above.

Figure 10 compares the performance of the NSJRC1 and NSJRC2 heuristic algorithms, SPT routing, and optimal routing (computed via brute force enumeration) for small-size networks of up to 10 source nodes distributed in a $20m \times 20m$ square grid ($D = 20m$). Since the transmission range of nodes is fixed to be $100m$, we have fully connected networks in which each node has a link to every other node, with a cost that depends on distance, as described above. Every point on a curve in Figure 10 corresponds to the average cost

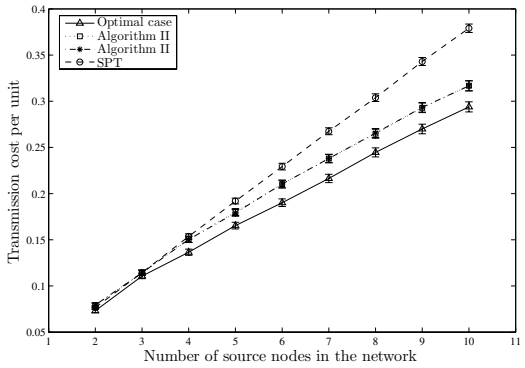


Fig. 10. Comparing the average transmission cost per bits of the optimal solution, SPT span- ning tree, NSJRC1 and NSJRC2.

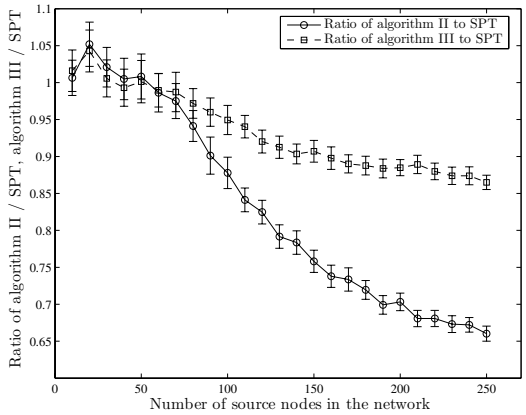


Fig. 11. Average ratios of total cost between heuristics NSJRC1 and NSJRC2 and SPT.

over at least 200 randomly-generated scenarios, and shows the 95% confidence intervals. Generally, we find that our heuristic algorithms perform close to optimal for these small problem sizes, and offer better performance than routing over the SPT tree.

Figure 11 compares the performance of the NSJRC1 and NSJRC2 algorithms and SPT routing, for larger network sizes of up to 250 nodes. Here, the y-axis value represents the ratio of the cost of routing via the NSJRC1- or JSNRC2-computed paths versus SPT routing. The results show that the proposed heuristics clearly outperform traditional SPT routing (which does not take the potential joint compression gains into account when determining the SPT path, but does jointly compress files as they are routed along the SPT path), especially when the number of source nodes in the network increases. This is because the average correlation between nodes increases as number of nodes in the network increase, while for lower density networks (when we have a smaller number of nodes in the network) we have less correlation among network nodes. This also explains the negative gain for sparse networks when $N < 50$, since there will be an incurred penalty for routing the initial file prefix to another node, only to find that the direct SPT path was indeed the best path to use. Finally, comparing the performance of algorithms NSJRC1 and NSJRC2, one can see that for dense networks, NSJRC1 provides better performance. This results from the fact that $|\mathcal{S}_c|$ increases more slowly than the number of source nodes in the network and NSJRC2 is a simplified version of NSJRC1 that applies the leaf deletion heuristic improvement only to nodes in $\mathcal{S}_c$, with leaf nodes that are not in $\mathcal{S}_c$ being routed over SPT.

V. Conclusion

In this paper, we have proposed and evaluated several *practical* algorithms for joint routing and compression of data files as they are forward from remote nodes to a central site, with the goal of minimizing the communication cost incurred. The simulation results showed that for small network sizes, our algorithms perform close to optimal, and for larger network sizes can significantly decrease the cost of routing over shortest path routing where paths are chosen independently of the correlation structure of data.

References

[1] http://ipmon.sprint.com/index.php.

[2] http://thth.berkeley.edu/tab-db/committeeinfo.php?tcid=10.

[3] http://www.casa.umass.edu/.

[4] http://www.cs.cmu.edu/ vsam/index.html.

[5] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao. Habitat monitoring: Application driver for wireless communications technology. *ACM Sigcomm Workshop*, April 2001.

[6] T. P. Coleman, A. H. Lee, M. Medard, and M. Effros. Low complexity approaches to slepian-wolf near-lossless distributed data compression. *IEEE. Trans. Inform. Theory*, 52:3546–3561, August 2006.

[7] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley Sons, New York, 1991.

[8] R. Cristescu, B. Beferull-Lozano, and M. Vetterli. On network correlated data gathering. *IEEE Infocom*, 2004.

[9] R. Cristescu, B. Beferull-Lozano, and M. Vetterli. Networked slepian-wolf: Theory, algorithms and scaling laws. *IEEE. Trans. Inform. Theory*, 51:4057– 4073, Dec. 2005.

[10] T. Han. Slepian-wolf-cover theorem for networks of channels. *Information and Control*, 47:67–83, 1980.

[11] T. Ho, M. Medard, M. Effros, and D. R. Karger. Network coding for correlated sources. *CISS*, 2004.

[12] W. Hu, V. N. Tran, N. Bulusu, C. tung Chou, S. Jha, and A. Taylor. The design and evaluation of a hybrid sensor network for cane-toad monitoring. *Information Processing in Sensor Networks*, April 2005.

[13] T. Kailath, A. Sayed, and B. Hassibi. *Linear Estimation*. Prentice-Hall, 2000.

[14] J. Liu, M. Adler, D. Towsley, and C. Zhang. On optimal communication cost for gathering correlated data through wireless sensor networks. *International Conference on Mobile Computing and Networking*, 2006.

[15] S. J. Lord, E. Kalnay, R. Daley, G. D. Emmitt, and R. Atlas. Using osses in the design of the future generation of integrated observing systems. *Symposium on Integrated Observation Systems*, pages 45–47, 1977.

[16] D. S. Lun, M. Mdard, T. Ho, , and R. Koetter. Network coding with a cost criterion. *ISITA*, 2004.

[17] R. N. Murty and M. Welsh. Towards a dependable architecture for internet-scale sensing. *Workshop on Hot Topics in Dependability*, 2006.

[18] S. Nath, A. Deshpande, Y. Ke, P. B. Gibbons, B. Karp, and S. Seshan. Irisnet: An architecture for internet-scale sensing services. *VLDB*, '03.

[19] S. Pattem, B. Krishnmachari, and R. Govindan. The impact of spatial correlation on routing with compression in wireless sensor networks. *Symposium on Information Processing in Sensor Networks*, 2004.

[20] S. S. Pradhan and K. Ramchandran. Distributed source coding using syndromes (discus): design and construction. *IEEE. Trans. Inform. Theory*, 49:626–643, March 2003.

[21] A. Scaglione and S. D. Servetto. On the interdependence of routing and data compression in multi-hop sensor networks. *ACM MobiCom*, 2002.

[22] D. Slepian and J. K. Wolf. Noiseles coding of correlated information sources. *IEEE. Trans. Inform. Theory*, 19:471–480, 1973.

[23] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin. Habitat monitoring with sensor networks. *Communication of the ACM*, 47(6):34–40, 2004.