Energy-Efficient Data Preservation in Sensor Networks With Spatial Correlation

Setu Taase, B.S, Bin Tang, Ph.D.

Computer Science, California State University, Dominguez Hills.

INTRODUCTION

Abstract:
Many sensor network applications require deployment in challenging environments. In these situations, it is not always possible to deploy base stations in or near the sensor field to collect sensory data. Therefore, the overflow data of the source nodes is first offloaded to other nodes inside the network, and is then collected when uploading opportunities become available. We call this process *data preservation in sensor networks*. In this paper, we take into account spatial correlation that exist in sensory data, and study how to minimize the total energy consumption during data preservation. We call this problem *data preservation problem with data correlation*. We show that with proper transformation, this problem is equivalent to minimum cost flow problem, which can be solved optimally and efficiently. Via simulations, we show that it outperforms an efficient greedy algorithm.

Background:

- We focus on one special kind of sensor network that is deployed in challenging environments.
- Each sensor node has limited storage capacity and a finite, non-replenishing battery power supply.
- The system conserves energy by eliminating redundancies in the system and offloading one copy of each data item for collection.

OBJECTIVES

1. We must first create a randomly generated virtual Sensor Network Model with nodes placed at arbitrary x-y coordinates. The system will be represented by the graph G(V,E), where V = the set of all Nodes and E = the set of Edges between Nodes. The minimum number of nodes as well as corresponding edges are dependent on an inputted transmission range between nodes.
2. From our graph we can output both entered and randomly generated data. Such data includes the energy used during transmissions between nodes, the number of data types as well as the number of copies of each data type, as well as the coordinates of each individual node and their respective neighbors (within a distance equal to the transmission range).
3. 3. Create a converted **Data Spatial Correlation Model** which is represented by a new graph G'(V', E'), where V' = the new set of Nodes with the addition of virtual (imaginary) nodes and E' = new corresponding set of edges.
4. 4. Generate and input data into a text file that is formatted as an input file for an Minimum Cost Flow (MCF) program.

MATERIALS AND METHODS

Network Model

We use Java programming language to create a program that generates a Sensor Network Model that takes inputs of network grid area (x-y), desired transmission range and number of nodes, number of data generating nodes (DG) and data items per data generator, number of data types and copies per type, and finally, storage capacity per storage node. Nodes are placed randomly on a grid and data items are generated randomly to DG across the network. Figure 1 shows a typical representation of a physical grid.

Nodes within the transmission range of each other will be labeled "neighbors" and the energy consumption between them is calculated via ((Eelec*k)*2) + (Eamp*k*(distance*distance)).

The data is then outputted to a text file to await conversion.

Conversion to Data Preservation with Spatial Correlation Model

In this program we employ Visual C++ to create a graph converter which essentially reads input from the network model and converts it into our new Data Preservation Model.

In addition to the data generating nodes (which we now call *source nodes*) and storage nodes, our converter generates a source to send data, imaginary nodes that act as the data types, and a sink to retrieve and store data. Figure 2 gives a representation of our new model.

The data is printed to a text file in the form of minimum cost flow input which defines each edge of our new model by adjacent nodes, capacity of the nodes and cost of transmission from one to the other. We use Djikstra's algorithm here to determine the shortest path between source and storage nodes.

Minimum Cost Flow – cs2

Using a Visual Studio command prompt, we set a Windows compiler in order to execute an existing program that reads in our converted data and determines minimum cost flow. This program is called cs2.

By reading each edge from the inputted file and taking into account the cost of each transmission, our program finds the shortest path and determines the least amount of energy required. We refer to this as the minimum cost flow.

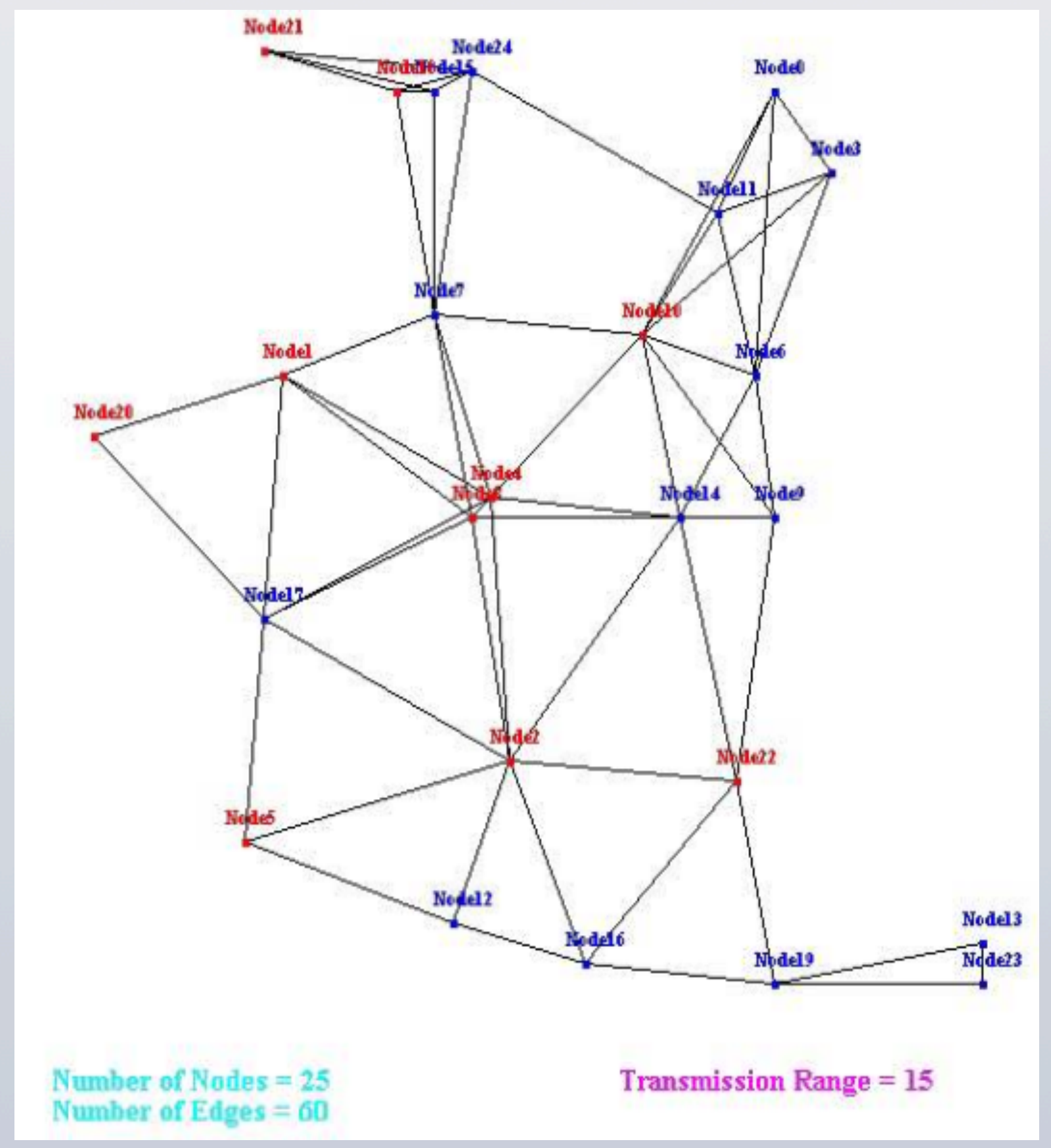

Figure 1.

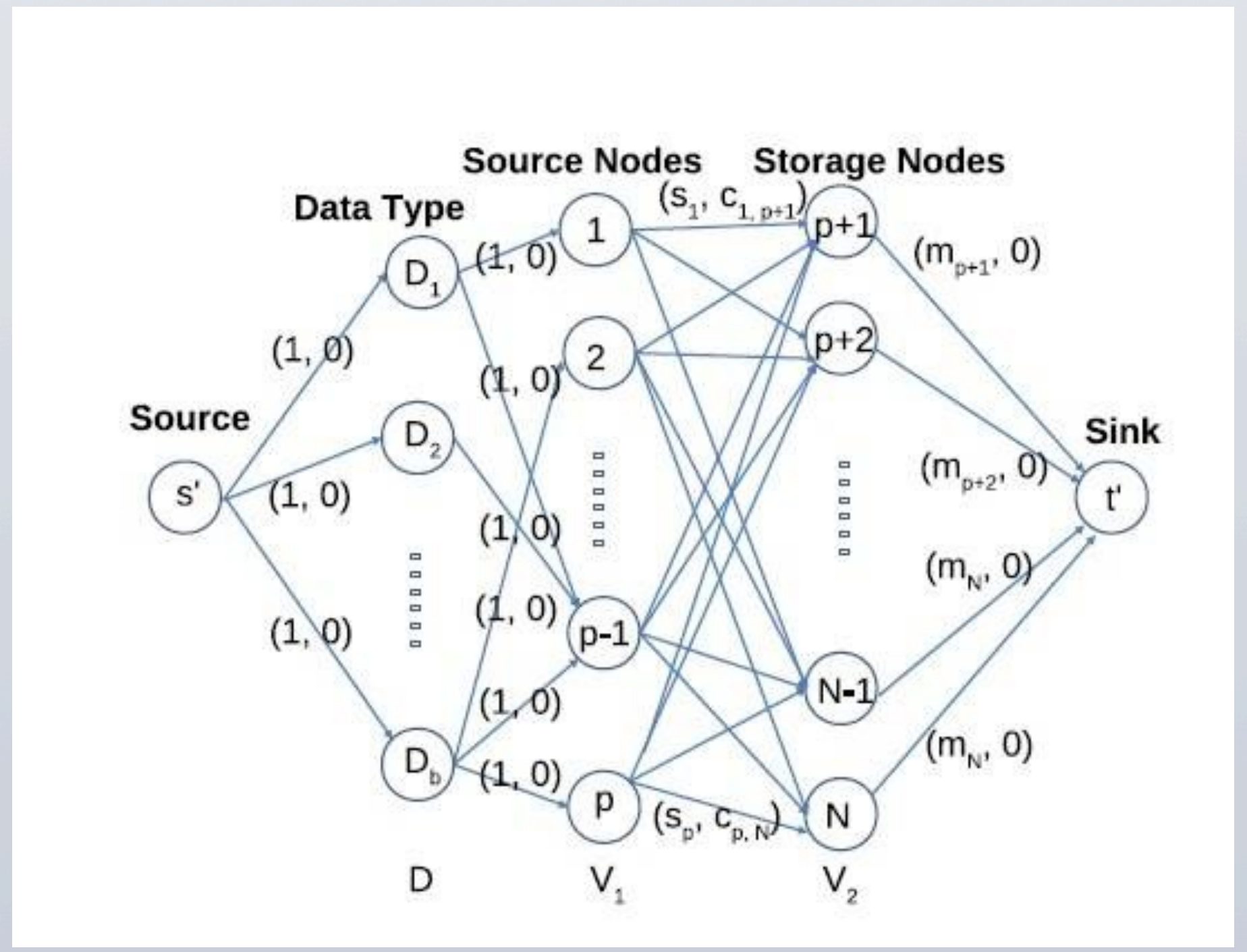

Figure 2.

RESULTS

Minimum Energy Consumption

We generate our network model and give the following inputs.

- Area of Grid = 50x50
- Transmission Range = {15, 20, 25, 30 , 35}
- Total Number of Nodes = 25
- Data Generators = 10, Storage Nodes = 15
- Data Items per DG = 10
- Data Types = 10, Copies per Type = {1, 2, 4, 6, 8, 10}
- Storage Capacity = 8

Tests are run using all methods described and we note the resulting cost in energy consumption. Figure 3 represents our data with varying transmission ranges and copies/data-type.

Figure 4 is the projected performance comparison between Optimal and Greedy Algorithms.

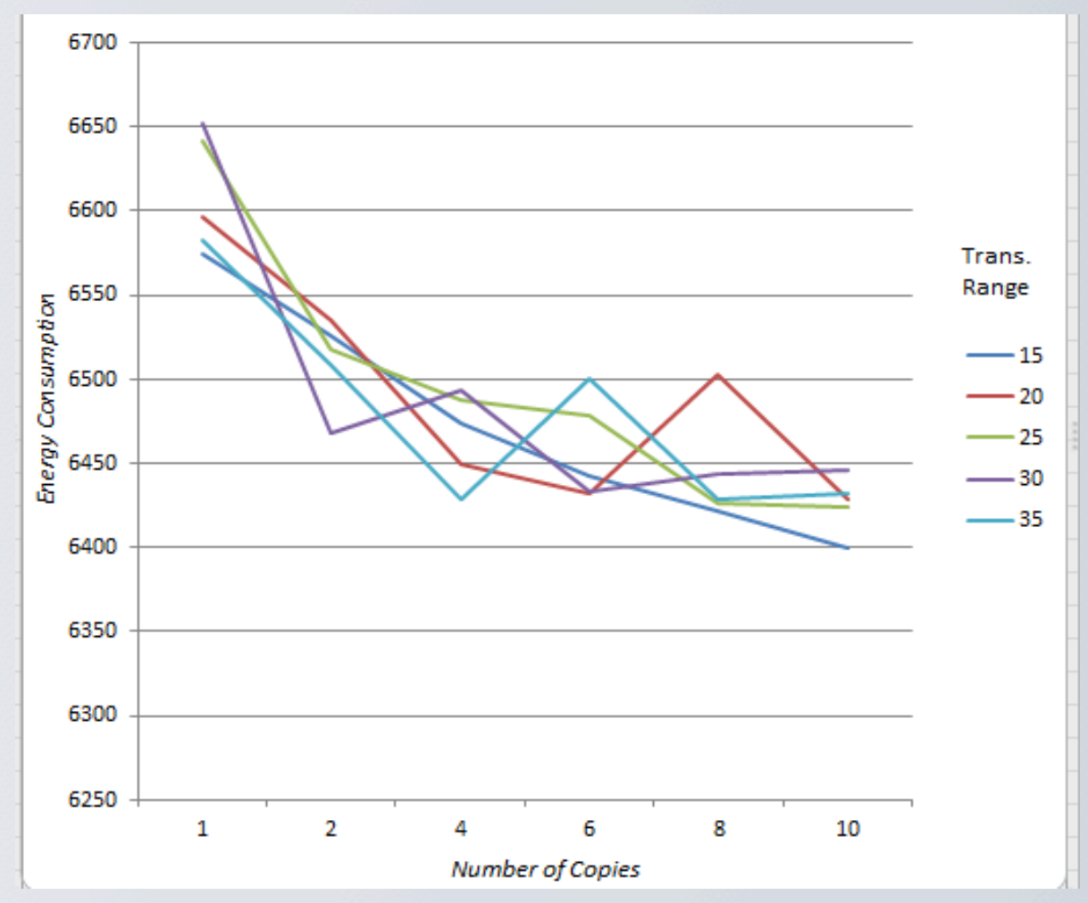

Figure 3

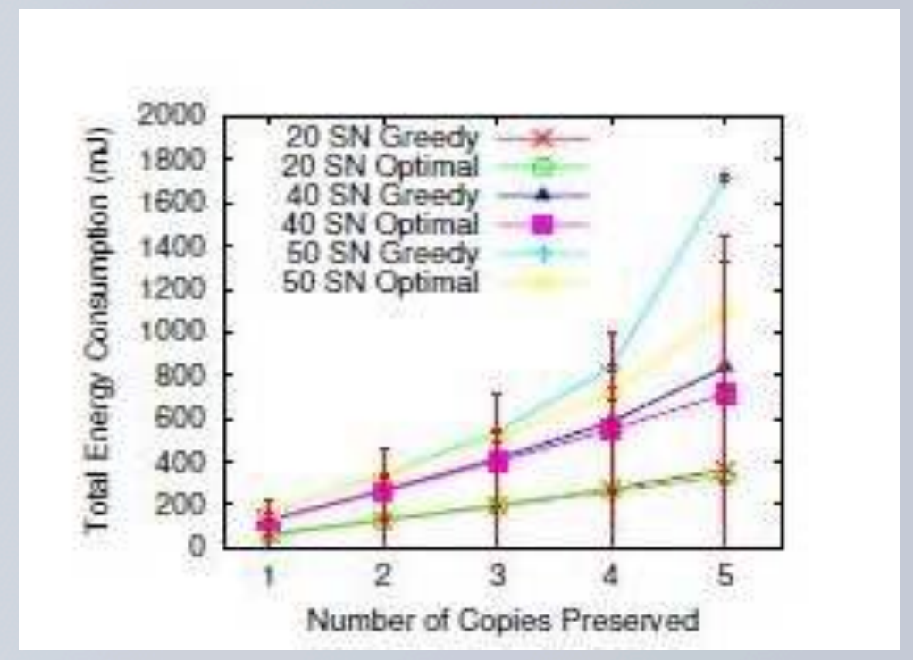

Figure 4

CONCLUSIONS

- With smaller transmission ranges, we determine that the average energy consumption increases as the number of copies decreases
- The larger the transmission range, the more random (non-linear) the energy consumption as the number of copies decreases. This may be due to the fact that many direct paths from source to storage are becoming available with a higher range, so that these transitions require fewer hops.. even as the number of copies decreases.
- We maintain that our algorithm outperforms standard Greedy Algorithms and leave more exhaustive testing to future work.

ACKOWLEDGEMENTS

I would like to thank my supervisor and mentor, Dr. Bin Tang, for his guidance and patience throughout this research. I would also like to acknowledge Mr. Javier Lopez and Mr. Preston Crary for their contributions to our efforts. Finally I would like to thank LSAMP and the NSF for their continued support of our work.