DATA REDISTRIBUTION PROBLEM IN DATA INTENSIVE SENSOR NETWORKS

A Thesis by

Rohini Kurkal

Bachelor of Technology, DVRCET, India, 2007

Submitted to the Department of Electrical Engineering and Computer Science

And the faculty of the Graduate School of

Wichita State University

in partial fulfillment of

the requirements for the degree of

Master of Science

December 2009

DATA REDISTRIBUTION PROBLEM IN DATA INTENSIVE SENSOR NETWORKS

The following faculty members have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

_____

Bin Tang, Committee Chair

_____

Bayram Yildirim, Committee Member

_____

Neeraj Jaggi, Committee Member

DEDICATION

This thesis is dedicated to my parents and sister

ACKNOWLEDGEMENTS

I would like to express deepest appreciation and sincere thanks to my advisor Dr. Bin Tang for his guidance and remarkable patience all through my research and valuable suggestions that helped to complete my thesis successfully. I'm grateful to him for his generous help, guidance, and patience throughout my Master's program.

I would like to thank Dr. Neeraj Jaggi's suggestions, which greatly improved the quality of my thesis. I would like to thank Dr. Bayram Yildirim for offering his helpful suggestions and time on my thesis

I would also like to thank Mr. Keenan Jackson who helped me in my coursework and advised me thorough out my coursework.

I shall always be indebted to my parents for making me capable to complete this thesis. I am also thankful to my friends Dharma Teja Nukarapu, Archana Sridharan who helped me and supported me morally and made my stay at WSU memorable. Finally I would like to dedicate this thesis to my dad who has been my ideal inspiration throughout my life.

ABSTRACT

Data redistribution problem has become a key challenge in the data intensive sensor networks (DISNs), wherein large volume of sensory data are sensed and generated from some sensor nodes about their surrounding physical world. Due to the resource constraints of sensor nodes there is a need to redistribute (offload) the generated data to the nodes with free storage space. However, such data redistribution, if not managed well, could be a serious energy drain not only to the data generators' battery power but also to other sensor nodes involved in the redistribution process. We implement the data redistribution algorithms, which deal with redistribution of generated data and strive to minimize the energy consumption incurred by the data redistribution, while fully utilizing the storage capacity in the DISNs. We first show that our redistribution problem is equivalent to the balanced assignment problem, which can be solved with well-known Hungarian algorithm. However, the Hungarian algorithm gives $O(N\overline{m})^3$ time complexity where N is the total number of sensor nodes in the network and $\overline{m}$ is the average storage capacity of each node. We design a fully distributed, highly scalable, and efficient data distributed mechanism, which is also adaptable to network dynamics such as dynamic data generating. We show both analytically and experimentally, our proposed distributed mechanism achieves best results. The goal of the thesis is to maximize the storage utilization of the sensor network and minimize the energy consumption required for the whole process of data redistribution. We focus on the in-network data redistribution where the data is redistributed between the highly utilized nodes and lightly utilized nodes.

TABLE OF CONTENTS

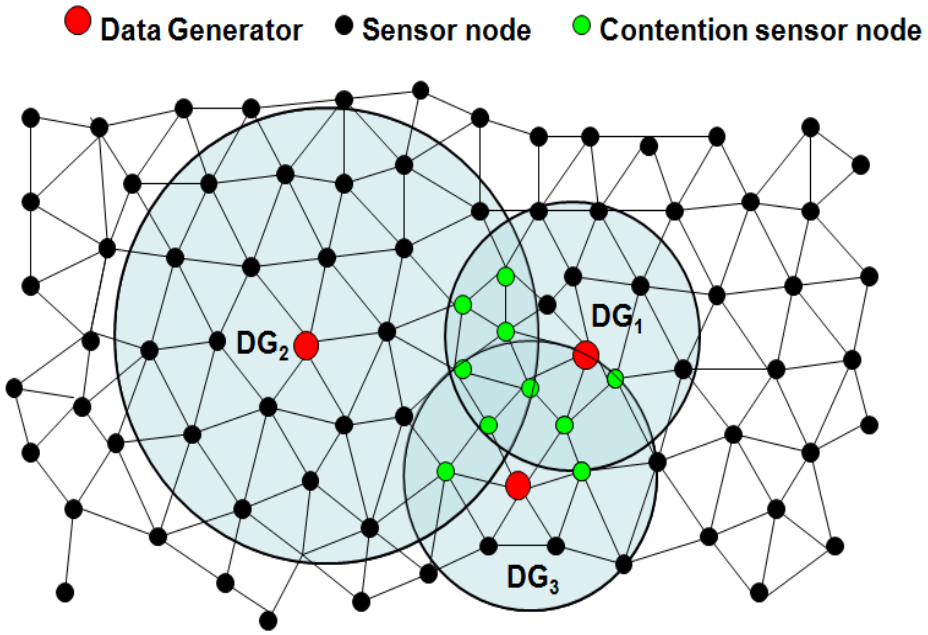
Chapter-1

INTRODUCTION

We study how to redistribute the large amount of data into the network to fully utilize the storage capacity of all the sensor nodes, while at the same time, minimizing energy consumption incurred by the data redistribution. Note that we are not concerned with the data retrieval, which can be done using data mules [15] or by human operators manually. Since data redistribution is energy expensive wireless communication, a data generator obviously prefers to offload the data to other sensors close to it. When there are very few data generators distant from each other's or the amount of data to offload is small, this problem becomes trivial – each data generator can perform a breadth first search (BFS) ordering of distance to other sensor nodes and offload data to its one-hop neighbors first, then two-hop neighbors and so on. However, when data generators are close to each other, or the amount of generated data is comparable to the amount of available storage space in the networks, redistribution contention arises. As shown in Figure 1, data generators DG1, DG2, and DG3 have conflicting offloading data storage area with each other. We call the sensor nodes, which are pursued by multiple data generators *contention sensor nodes*. The challenge is how to resolve such contention for data generators while still achieving optimal energy-efficient data redistribution.

Two questions have to be answered:

- How does a node know it is in the contention area?
- Which data generator should go to get the contention node (especially, when one DG's contention set is a subset of the other)?



.

Fig.1. Data redistribution problem in sensor networks.

Specifically, our data redistribution problem and algorithms should address the following questions: In an offline and centralized version where the data generators and the amount of data to be redistributed by each are already known, how does each data generator choose the destination nodes to redistribute their data? How much data to redistribute? In an online and distributed environment, where nodes do not know the network topologies and data are generated dynamically, how do data generators coordinate with each other so not to redistribute to the same area (or nodes) in the sensor networks? To answer the first question, we formulate the data redistribution problem as a graph-theoretical problem and show that it is equivalent to the assignment problem, which has polynomial time optimal solution. For the second question, we consider the data generators as electrical point charges and model the sensor network as the

electrostatic potential field in physics. We study the data redistribution as the movement of electric particle in the potential field consequently.

1.1 Contribution of Thesis

The main results and contributions are as follows:

1) To the best of our knowledge, our work is the first one to formally formulate and study the data redistribution problem in sensor networks.

2) We show that the data redistribution problem is equivalent to the classic balanced assignment problem, which can be solved optimally in polynomial time using Hungarian algorithm [7].

3) We propose heuristic algorithms to optimize Hungarian algorithm in the aspect of reducing the number of sensor nodes involved.

4) We design a novel distributed algorithm by applying potential function of Electrostatics in Physics. Specifically, we implement a distributed protocol to establish potential fields to facilitate the data redistribution.

5) Through our own simulator (written in C language), we show that our distributed algorithm is competitive to the centralized algorithm.

Chapter-2

LITERATURE REVIEW

2.1 Background and Motivation

2.1.1 Data Intensive Sensor Networks (DISNs)

Wireless sensor networks consist of spatially distributed small computing, communicating and sensing devices that have limited resources (processing speed, battery power, storage capacity, and communication bandwidth), which allow us to interact with the physical world around us. It has become a reality that the sensor network applications are no longer limited to just ambient sensing (e.g. light or temperature) or environmental and weather monitoring. With the emergence of a rich collection of sensory sources such as video cameras, microphones, RFID readers, telescopes and seismometers, a whole new array of data-intensive sensing applications have been researched and developed recently. They include multimedia surveillance networks [5], visual and acoustic sensor networks [10, 18], underwater or ocean seismic sensor networks [1, 9, 20] and geophysical monitoring [12, 21]. In such data intensive sensor networks (DISNs), large amount of sensed data about the physical environment are generated continuously from some sensor nodes called *data generators*.

2.1.2 Challenges

One of the major research problems in data intensive sensor network is how to manage the large amount of generated data and the mechanism to store these data for future retrieval or analysis. We call this problem as data redistribution problem. Large amount of data loss can be avoided by offloading data at the right time and to the right neighboring nodes.

The need for data storage management arises primarily in class of sensor networks where information collected is not relayed to observers in real-time. The storage determines useful

lifetime and coverage of network. As once the available storage space is exhausted, a sensor can no longer collect and store data locally. Thus, data redistribution plays vital role where the rate of data generation is not uniform at sensors, some sensors may run out of storage space while space remains available at other nodes. By redistributing the data we can stabilize the network and balance the data among the sensor nodes. We study how to redistribute the data to the under-loaded nodes and select a best possible destination sets for offloading the data from the over-loaded nodes.

Despite the advances in large lower-power flash memory such as parallel NAND flash technology [2], storage is still a serious resource constraint in data intensive sensor networks. According to [10], an acoustic sensor that has a 1GB flash memory and is designed to sample the entire audible spectrum will run out of its storage in 7 hours. Therefore, a major challenge in DISNs is that how to temporarily store the massive amount of generated data inside the sensor networks under limited storage capacity and battery power.

2.2 Related Work

More recently, new cooperative storage system for sensor networks called EnviroStore has been proposed by Luo et al. [11] to maximize the utilization of the network's data storage capacity. To the best of our knowledge, [11] is the only work to study data redistribution to maximize data storage capacity in sensor networks. The storage system of EnviroStore is mainly designed for the disconnected operation in sensor networks where sensor nodes are deployed without any connected path to a base station. They came up with two data redistribution mechanisms. One is called *in-network data redistribution*, wherein data are migrated from nodes that are highly loaded (in terms of storage capacity) to nodes that are not. The other is called *cross-partition data redistribution*, wherein data are offloaded from overloaded network partitions to under

loaded partitions using mobile data mules. Both data redistribution mechanisms are heuristic-based. We focus on the problem of in-network data redistribution. We formulate the data redistribution problem and design algorithms to achieve both storage maximization and energy minimization.

The *data migration problem* has been studied extensively in the field of parallel computing [14, 17] and disk storage [6]. They mainly study how to schedule workload and move associated data from source processors to destination processors, or change one storage configuration into another, to better respond to the data demand changes for the purpose of load balancing.

Our problem concerns with the data redistribution energy minimization while fully utilizing storage capacity in wireless sensor networks. Our problem bears a resemblance to the graph Voronoi diagram problem [3] in the sense of "areas of influence". Graph Voronoi diagram is the graph theory equivalency of the classical Voronoi diagram in computational geometry. Graph Voronoi diagram characterizes regions of proximity in graphs based on shortest paths between nodes. Yet there are two differences between our data redistribution problem and graph Voronoi diagram problem. First, in data redistribution problem, the node has weight, which indicates the size of the data to be redistributed. Second, graph Voronoi diagram does not consider the "capacity" of each node, which in our problem signifies the available storage space of sensor nodes.

We compare our problem with *generalized assignment problem* proposed by Shmoys and Tardos[16]. It is stated as follows. There are a set of jobs and a set of machines – each job is to be processed by exactly one machine. Processing job j on machine i requires processing time p_{ij} and incurs a cost of c_{ij} . Machine i has T_i processing time available. The objective of the generalized assignment problem is to minimize the total cost under the processing time

constraint of each machine. Shmoys and Tardos design a polynomial-time algorithm which, given a value C , either proves that no feasible schedule of cost C exists, or finds a schedule of cost at most C where machine i is available for at most $2T_i$ processing time. To map the generalized assignment problem to our data redistribution problem, c_{ij} is the shortest distance between the data generator of data item i and node j; each node has T_i available storage space; storing data item j on node i occupies d_j of i's storage space. Being a graph-theoretical problem, our problem differs from generalized assignment problem. Moreover, Shmoys and Tardos relaxed the processing time constraint of each node. In our problem, the storage constraint each node is a stringent constraint and thus can n relaxed.

There are a number of works that have adopted the idea of potential functions in sensor networks (see [19] for a good survey paper). They either study how to route packets from source to destination to avoid congestion in anycast [8] or multipath routing [13], or study the placement of mobile sinks in wireless sensor networks for energy balancing. In all those problems, there are particular traffic sources and sinks. In our data redistribution problem, we have traffic sources (data generators) while trying to find the sinks (offloaded nodes). The goal is to maximize storage utilization while reducing redistribution energy cost, which is different from above problems.

2.3 Organization of thesis

The rest of the thesis is organized as follows. In Chapter 3, we formalize the data redistribution problem and illustrate it with a simple example and prove that the data redistribution problem is equivalent to the balanced assignment problem. We present our data redistribution algorithm called potential field based distributed algorithm. In Chapter 4, we compare our proposed

centralized algorithms and our distributed algorithm with Hungarian algorithm, and present our analysis. Finally we make conclusion and point out our future work in Chapter 5.

Chapter-3

Proposed Model

3.1 Data Redistribution Problem

In our data redistribution problem, there are sensor nodes, which generate continuous large amount of data and other sensor nodes, which do not generate any sensory data. Sensory data is a sequence of raw data item, which are of unit size. As the sensor nodes have limited storage capacity, the data generators when fully utilize their local storage has a need to redistribute the additional data to the sensor nodes with available storage space.

The objective of the data redistribution problem is to redistribute/offload the data items from the data generators to other nodes in order to maximize the overall storage capacity of the entire network while minimizing the power consumption of the entire network. We measure the power consumption as a metric of number of hops (i.e. the distance from the data generators to all the nodes to which the data items have been redistributed). We choose the optimal path and thus reducing the overall cost of the network. We state the power consumption as the total redistribution cost of the network.

3.2 Problem Formulation

We represent our sensor network graph as G (V, E) where V = {1, 2, …, i,…, N} is the set of nodes and E is the set of edges. Two nodes can communicate directly with each other if there exists an edge between them. We use d_{ij} to denote the shortest path distance (in number of hops) between two sensor nodes i and j. We use s_i to denote the number of data items node i needs to redistribute and m_i the available free storage space (in number of data items) at node i. If $s_i > 0$, then $m_i = 0$, meaning node i is full and it cannot store data items offloaded by other nodes. Node i is a data generator in this case. If $s_i = 0$, then node i can store m_i data items offloaded from other

nodes.

Each data generator redistributes one data item at a time. For our energy cost model, we use the number of hops a data item transmits to approximate the energy consumption of redistributing the data item. This can be explained by the equal size of the data items and uniform distribution of sensor nodes, according to the First order radio model [4]: for a k-bit data over distance d, the transmission energy. $E_{Tx}(k,d) = E_{elec}* k+ \epsilon_{amp} *k*d^2$, the receiving energy. $E_{Rx}(k)= E_{elec}*k$, where E_{elec} and ϵ_{amp} are constants . The *redistribution cost* for data generator *i* (with s_i number of data items to redistribute) is the sum of the number of hops to redistribute all s_i data items. The *total redistribution cost* of the sensor networks is defined as the sum of the redistribution cost of all the data generators s. The goal of the problem is to redistribute the data items from the data generators into the network to maximize the storage utilization with minimum redistribution cost. Without loss of generality, we assume that the total size of the data items to be redistributed is less than or equal to the size of the total available storage space in the network.

Let I denote the set of data items to be redistributed in the whole network, and let $S(i)$, where $i \in I$, be data item i's data generator. A redistribution function is defined as $r : I \rightarrow V$ indicating data item $i \in I$ is distributed to node $r(i) \in V$ via the shortest path between $S(i)$ and $r(i)$. Our goal is to find such a redistribution function r to minimize the total redistribution cost:

$$\sum_{i \in I} d_{S(i)r(i)}$$

Under the storage constraint that:

$|\{i \mid i \in I , r(i) = j \}| \leq mj$ For all $j \in V$.

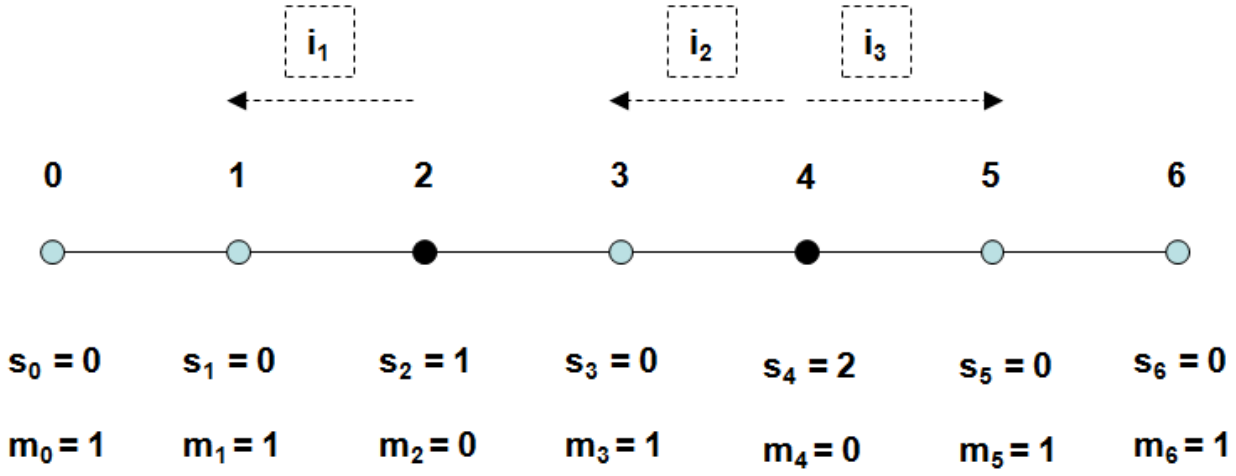


Fig. 2. Illustrating data redistribution problem under storage constraint.

Below we give a simple example to illustrate the data redistribution problem under storage constraint.

EXAMPLE 1: Figure 2 illustrates the above described data redistribution problem in a small linear sensor network with seven sensors, each graph edge has a unit weight. The storage capacity and the number of data items to be redistributed of each sensor are shown in Figure 2. Node 2 has one data item, i_1, to redistribute; node 4 has two data items, i_2 and i_3, to redistribute. The minimum cost solution is node 2 sends i_1 to node 1, while node 4 sends i_2 and i_3 to nodes 3 and 5, respectively. Total redistribution cost is 3 hops.

3.3 Assignment Problem and Hungarian algorithm

We prove that our data redistribution problem is a balanced assignment problem and introduce the Hungarian algorithm, which is an optimal solution for assignment problem.

Balanced assignment problem: It can be stated as a problem where n individuals are assigned to n different jobs based on the cost for doing the respective jobs. The main idea here is to

minimize the total cost of completing these jobs by individuals. Below we show that our data redistribution cost is equivalent to the balanced assignment problem.

Theorem 1: The data redistribution problem is equivalent to assignment problem.

Proof: There are p data generators in the sensor network, with s_1, $s_{2,\ldots,}$ s_p data items respectively. For a network with N nodes, there are $q = N\text{-}p$ regular sensor nodes that has available storage space, with m_1, m_2, ..., m_q storage space respectively. We do the following transformations. First, since each data item has one unit memory space, we subdivide data generator node i into s_i (number of data items it has) nodes, each corresponding to one of its data items to redistribute. Second, for each sensor node i that has m_i units of available storage spaces, we subdivide node i into m_i number of nodes, each corresponding to one unit memory space of node i. As shown in Figure 3, we get a bipartite graph, where the left hand side is the set of data items and the right hand side is the set of available unit storage spaces. There is an edge between any node (data item) on the left and any node (unit memory space) on the right (we do not plot them for clarity), and the distance between them is the shortest distance (in terms of number of hops) between each data item's data generator and the regular sensor node the unit memory space belongs to.

If the number of data items is less than the number of the total available memory spaces in the sensor nodes (note that we assume the total size of the data items to be distributed is less than or equal to the size of the total available storage space in the network), we add dummy nodes on the left with edge cost zero to make it balanced assignment problem, which can be solved by Hungarian algorithm illustrated below.

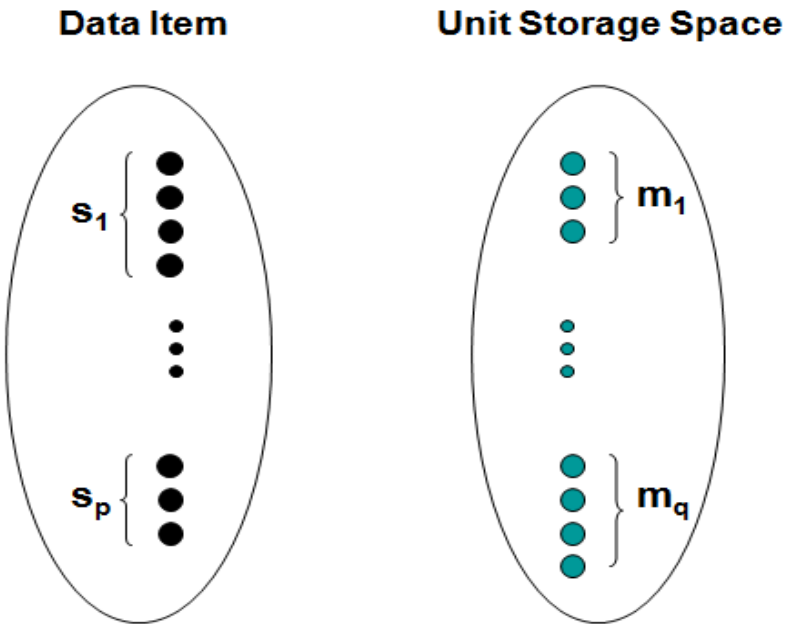


Fig. 3. Assignment problem.

Hungarian algorithm:

The algorithm is stated below:

We construct a cost matrix c_{ij} with i rows and j columns where $1 \leq i,j \leq n$, we consider n as the total network size. The steps are as follows:

Step 1. (Finding reduced cost matrix) Find the minimum element in each row of cost matrix and subtract the value from each row and then find the minimum element in each column and subtract the value from each column. The resultant matrix is the reduced cost matrix.

Step 2. Cover all the zeros in the reduced cost matrix with minimum number of lines. Let m be the lines required to cover all zeros.

- If $m<n$, find minimum uncovered element k and subtract it from every uncovered element. Add k to each element that is covered by two lines. Continue step 2.
- If $m=n$, go to step3.

Step 3. Starting with top row to make assignment till last row. Iterate all the above steps until we get unique assignments.

The time complexity of Hungarian algorithm is $O(n^3)$. In our data redistribution problem wherein N sensor nodes exist and the average storage capacity of sensor node is $\overline{m}$, the time complexity is $O(N\,\overline{m})^3$.

3.4 Potential Field-Based Distributed Algorithm (PDA)

In this section, we present our potential field-based data redistribution model. First, we give an overview introduction of the basic idea. Next, we introduce the definition of potential fields in sensor network context and describe how data items are redistributed along those fields. Third, we discuss convergence issue. Finally, we give a detailed explanation of the protocol.

Let's start with the Example 1 in Figure 2 again. The minimum cost solution is that node 2 sends its data item to node 1, while node 4 sends one data item to nodes 3 and the other one to node 5. However, in a distributed environment, node 2 obviously does not know the whole topology. From its perspective, it does not matter whether offloading its data to either node 3 or node 1 since both have the same redistribution cost. However, we show below the concept of *potential* to further characterize the difference among the sensor nodes and to help sensor nodes (including data generators) to make right decision as to which neighbor nodes to offload data.

Potential field model: We study the data redistribution using the analogy that the whole sensor network is an electric potential field, wherein each data generator is an electric charge. For data generator S_i with s_i data items to offload, it has a positive electric charge of s_i. For arbitrary node j, its potential due to data generator S_i, denoted as ø(i,j), is characterized as

$$k_0 \frac{s_i}{d_{s_i j}}$$

, Where k_0 is a constant and $\frac{s_i}{d_{s_i j}}$ is the distance (in terms of number of hops) between node j and data generator *Si*. According to superposition principle reference, the field of the whole sensor network is the linear superposition of all individual fields of the data generators. Therefore for arbitrary node j, its total potential, defined as *ø(j)*, is:

$$\sum_{i=1}^{p} \emptyset(i,j) = \sum_{i=1}^{p} k_0 \frac{s_i}{d_{s_i j}}$$

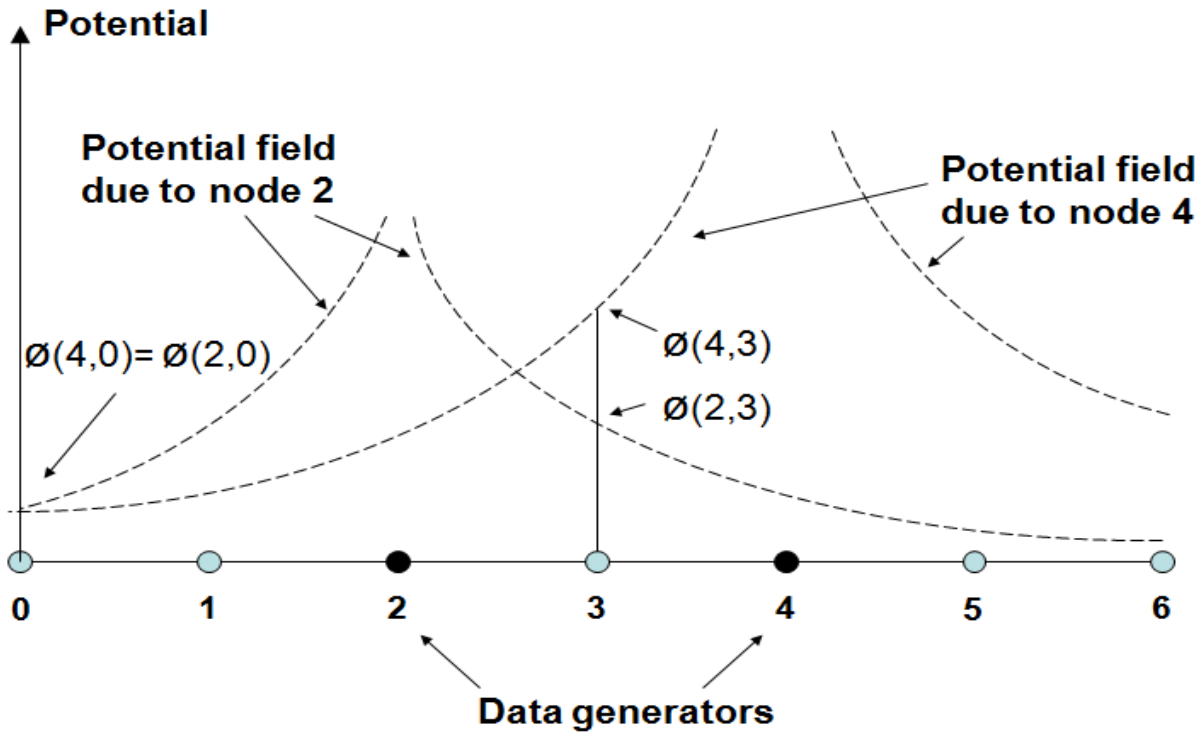


Fig. 4. Potential field in the sensor networks.

Figure 4 shows the individual potential field due to data generators node2 and node 4, for the linear sensor network depicted in Figure 2. For example it shows that ø(4,3) = 2 × ø(2,3) and ø(4,0) = ø(2,0). It can be seen that for node 2's two neighbors, node 1 and node 3, their total

potentials ø(1) < ø(3). All such information are utilized by the data generators and sensor nodes to make intelligent decisions in the process of data redistribution.

Potential field based data redistribution.

With above preparation, we need to further answer two questions. First, how can we build up the potentials that can effectively guide the data redistribution? Second, how can data generators coordinate with each other when there are contention sensor nodes? (i.e., multiple data generators offload to the same sensor nodes.) In the following, we address the above questions, which finally lead to a fully distributed data redistribution algorithm.

We perform data redistribution by forwarding data items towards the steepest gradient along the potential field. This is analogous to the movement of electrical particles in electrostatic field. By following the steepest gradient, data items eventually reach their equilibrium status. The steepest gradient at each node is determined by comparing its Potential value with that of its neighbors, and it is towards the neighbor with the lowest potential value. Similarly, we can assume there is a "force" existing between any node and its neighbor nodes, which pushing the data items out of the data generators and into the network. Now consider a data item to be redistributed from S_i. First, S_i will always offload its data items to its closest nodes. Among the nodes with the same number of hops to S_i, S_i first offloads to the sensor node that has the minimum potential (due to the observation that the higher the potential of the node, the more offloading contention of it). This explains why in Example 1, data generator 2 sends its data to node 1, instead of node 3, as shown in Figure 4.

Potential field based data redistribution protocol:

The PDA happens in iteration. Each iteration consists of below three stages:

1) **Advertisement Stage.** For data generator that has data items to offload, say S_i, it floods a

message to the network with its ID and number of data items to offload (s_i). The algorithm stops when all the data generators have offloaded their data items. A time-to-live (TTL) value is included in the message indicating how many hops the message has traveled. The TTL value serves two purposes. First, it allows every receiver sensor to determine its distance (in terms of number of hops) to the data generator who initiated the flooding. Second, it allows limiting the flooding scope by only rebroadcasting messages, which have a TTL value greater than zero. This reduces the communication overhead. However, it does affect the accuracy of the data generator information received by other sensors.

2) **Storage commitment Stage**. For each regular sensor *j* with available storage space *m*, on receiving such message from S_i, it does the followings steps:

a) Computes its potential value due to the data generator S_i, $\emptyset\ (i,j) = \frac{s_k}{d_{s_k j}}$. Note we omit k_0 for clarity.

b) Finds the data generator that gives the maximum potential value. Suppose such data generator is S_k, that is, $k = argmax_{1 \le i \le p} \emptyset\ (i, j)$ ($\emptyset(i,j) = 0$ if data generator j did not receive S_i's message)

c) *Commits* one unit of storage space to S_k. Updates $S_k = S_k - 1$ and $\emptyset\ (i,j) = \frac{s_k}{d_{s_k j}}$

d) If j has committed all its available storage space, goes to Step e) below. Otherwise, goes back to Step b).

e) Sends a message to each data generator which it committed storage space to, say s_i, with the number of storage space it committed, c_{ij}.

3) **Data Offloading Stage**. After receiving the commitment message, each data generator S_i does the following steps:

a) Compares the total number of received commitment, $c_i = \sum_{j \in V} c_{ij}$, with the number of data items to offload, s_i. If $c_i \leq s_i$, s_i can completely satisfy all the commitments and thus sends to each sensor node the amount of data the sensor node committed to store. Otherwise, s_i offloads data to the closest among all the committed sensors.

b) Updates its left data items to be offloaded for next round. If it still has data to offload, goes back to Stage 1).

We see from above that in each iteration, each sensor commits all its storage space. However, it could be the case that the sensor's commitment is not satisfied by its intended data generator. In this case, the sensor will participate in the data redistribution in the next round. Note that for either sensor nodes that no longer have storage spaces available or data generators that no longer have data items to offload; they do not participate in the next iteration.

Discussion of PDA: PDA does not reply upon the knowledge of the initial and remaining storage capacity of other sensors. Further PDA does not reply upon a routing table and routing protocol support. So it is suitable to dynamic environment where maintaining a full fledged routing table is difficult.

Performance and convergence analysis of PDA

Below we show the convergence and performance analysis of PDA.

Theorem 2: The PDA will stops in at most *p* rounds, where *p* is the number of data generators in the network.

Proof: We first show that in each round, at least one data generator would receive commitment more than its number of data items to be offloaded, by way of contradiction. Assume that in the first iteration of PDA, none of the data generator receives more commitment than number of its

data items. That is,

$$\sum_j c_{ij} \leq s_i$$

Sum up among all the data generators, we get

$$\sum_i \sum_j c_{ij} < \sum_i s_i$$

Which is $\sum_i \sum_j c_{ij} < \sum_i s_i$

$$\sum_j m_j < \sum_i s_i$$

This contradicts with the assumption that the total size of the data items to be distributed of all the data generators is less than or equal to the size of the total available storage space in the network. Therefore, at each round, at least one data generator will finish offloading. It takes at most *p* rounds, where *p* is the number of data generators in the network.

EXAMPLE 2: Figure 5 illustrates the working of the potential field based data redistribution for the linear sensor network shown in Figure 2. Initially the data generators, node 2 and node 4 advertise its ID and the amount of data to be redistributed to the entire network. Each sensor node calculates the potential value due to each data generator. In storage commitment stage, each sensor node commits to the data generator with maximum potential value. Here node 1 commits to node 2. Node 3, node 5 and node 6 commits to node 4.Whereas node 0 randomly commits to any of the data generator as it has same potential value for both the data generators. In the data offloading stage, node 2 selects node 1 to offload its data items and node 4 selects node 3 and node 5 to offload its data items.

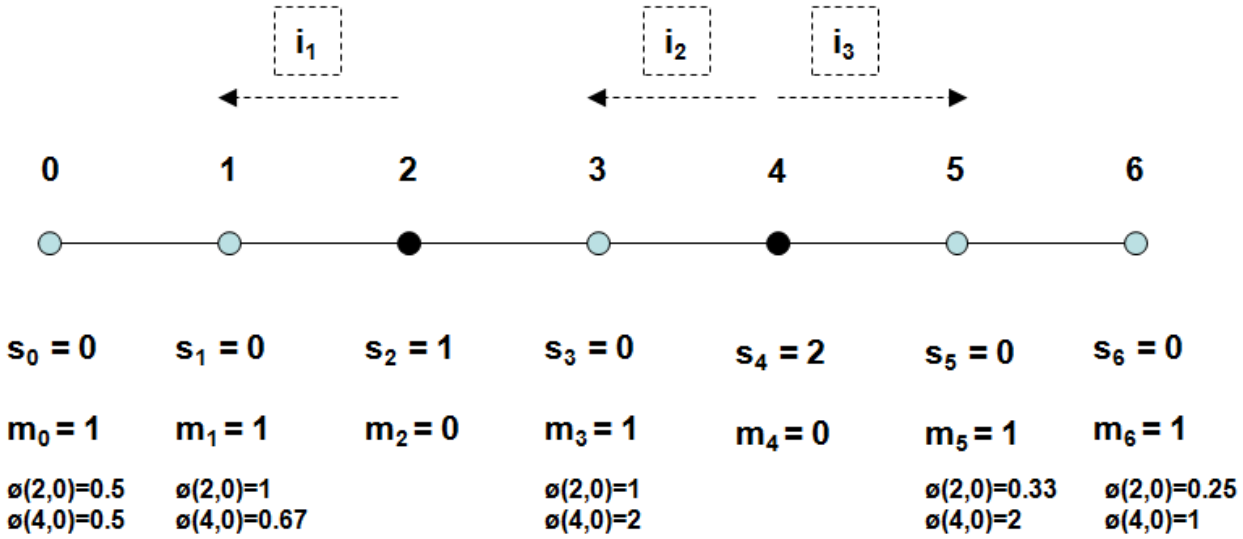


Fig.5. Potential field based data redistribution example.

Chapter 4

Performance Evaluation

To evaluate the performance of our proposed PDA algorithm, we have run an extensive set of simulations. In this simulation study, we have considered some scenarios for performance evaluation. For the purpose of our comparison, besides Hungarian Algorithm (HA), we also present several centralized heuristics, viz., Random Algorithm (RA), Greedy Algorithm (GA), and Cooperative Algorithm (CA).

4.1 Centralized Heuristics

Random Algorithm (RA)

In RA, each data generator selects a node randomly and asks if it has storage space. If yes, it offloads the data items to the node otherwise, it continues to ask another random node until all its data items are offloaded. The performance of the Random Algorithm is worse when compared to other algorithm. The redistribution cost involved in RA is very high as the randomly selected node can be far away from the data generators. Thus, it is better to avoid RA for data redistribution.

Greedy Algorithm (GA)

In GA, each data generator always asks its closest nodes first. Here the priority of the data generators offloading the data items plays important role. The data generators with lower id offload its data items completely at first place and the nodes with highest id offload its data items at the end. GA performs better when compared to RA as the data generators select the nodes with least hop number from itself. But when compared to Hungarian algorithm it performs worse as the data generators with highest id have to offload its data items to the farthest node, which result in high redistribution cost.

Cooperative Algorithm (CA)

In CA, each data generators cooperate with each other for the redistribution of data items. Here each data generator is given equal opportunity to offload its data items one by one until it completely offloads its data items. At first the data generator with lower id offloads one of its data item to other nodes and then the node with higher id offloads one of its data item to rest of the nodes with available storage space. This continues until all the data generators offload the data items completely. As a result, the nodes with highest id are also given priority to offload its data items to its nearest nodes and thus reducing the total redistribution cost of the network. CA performs better than GA and RA.

4.2 Simulation setup

We simulate the results using our own simulator (written in C language). The sensor nodes are deployed in a grid topology. For our simulation we consider three scenarios:

- The data generators are distributed at the center of the network.
- The data generators are at one corner of the network.
- The data generators are randomly distributed over the network.

The data items to be offloaded by the data generators are also considered according to below cases:

- The data generators with same number of data items to be redistributed.
- The data generators with different number of data items to be redistributed.

Below Table 1 shows the parameters we have used for the purpose of comparison in our simulations.

Parameters	Value
Network size	20 x 20 Grid, 100 x 100 Grid
Maximum data to be redistributed	40-99
Number of data generators	4, 20-80

Table 1: Simulation Parameters.

Data generators at the center of the network with different amount of data to be redistributed. In Figure 6 and 7, we deploy 400 sensor nodes evenly on a 20 × 20 network. The transmission range is unit one and each node has at most four one-hop neighbors. Nodes in black (nodes at (8, 10) and (10, 10)) are the data generators, which are two hops away from each other. The nodes in red represent the destination set for the data generator at (8, 10) and blue nodes are the destination set for the data generator at (10, 10). The data generator with lower id has 90 data items to redistribute whereas the data generator with higher id has 40 data items. The total redistribution cost for GA is 662 and for Hungarian algorithm the cost is 630. We observe that Hungarian algorithm is optimal as the cost involved in the redistribution is less when compared to GA.

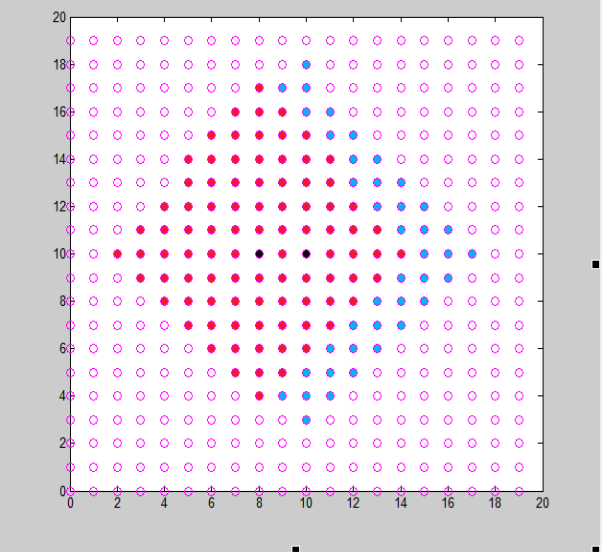


Fig.6. Redistribution with cost of 662 in GA.

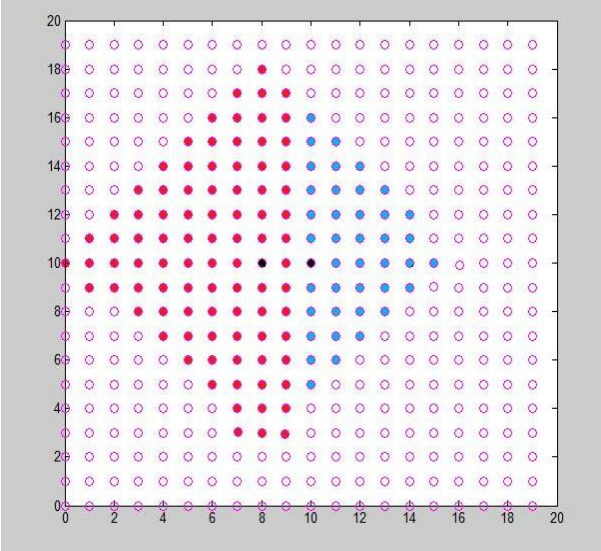


Fig.7. Redistribution with cost of 630 in HA.

Data generators at the center of the network with same amount of data to be redistributed. We deploy 400 sensor nodes on a 20 × 20 network out of which we consider four nodes as the data generators. Figure 8-12 shows the visual plot for the GA, HA, CA, RA and PDA respectively. Nodes in black (nodes at (8, 10), (12, 10), (8, 9), (12, 9)) are the data generators. The nodes in red represent the destination set for the data generator at (8, 10), blue nodes are the destination set for the data generator at (12, 10), yellow nodes represent the destination set for the data generator at (8, 9) and grey is the destination set for the data generator at (12, 9). Each of the data generators has 99 data items to be redistributed.

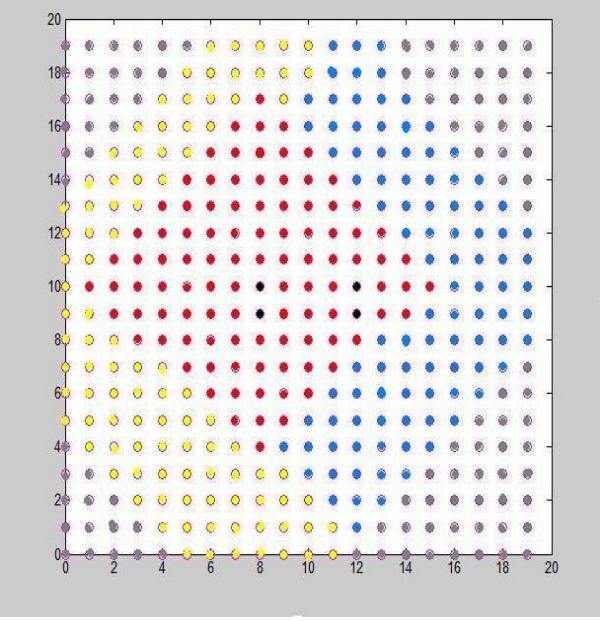


Fig.8. Redistribution with cost of 3524 in GA.

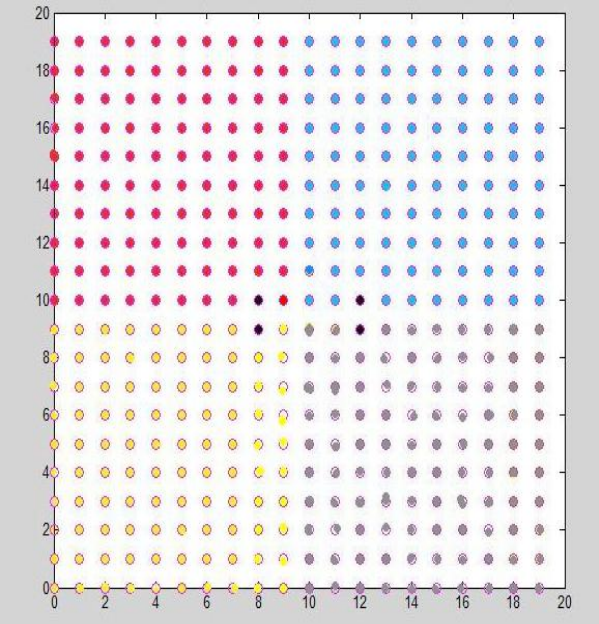


Fig.9. Redistribution with cost of 3160 in HA.

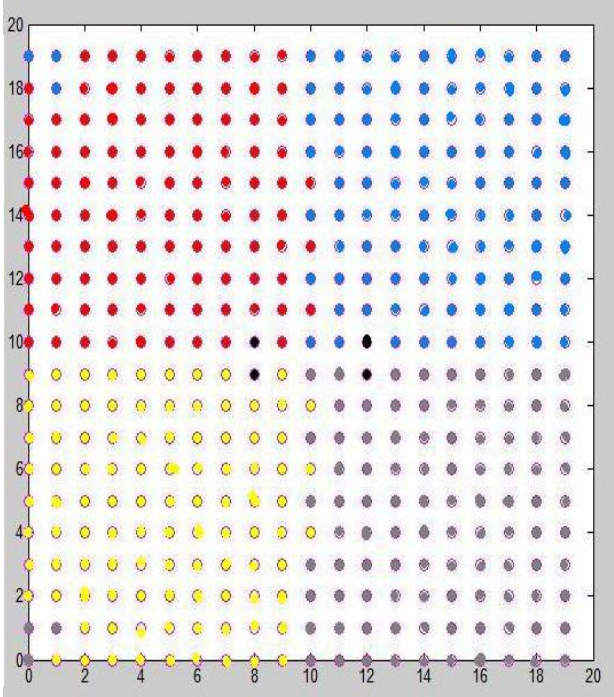


Fig.10. Redistribution with cost of 3200 in CA.

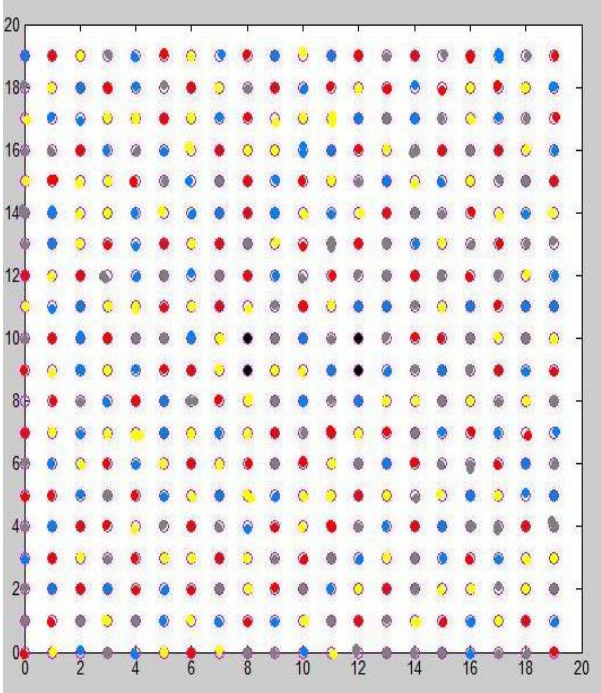


Fig.11. Redistribution with cost of 5235 in RA.

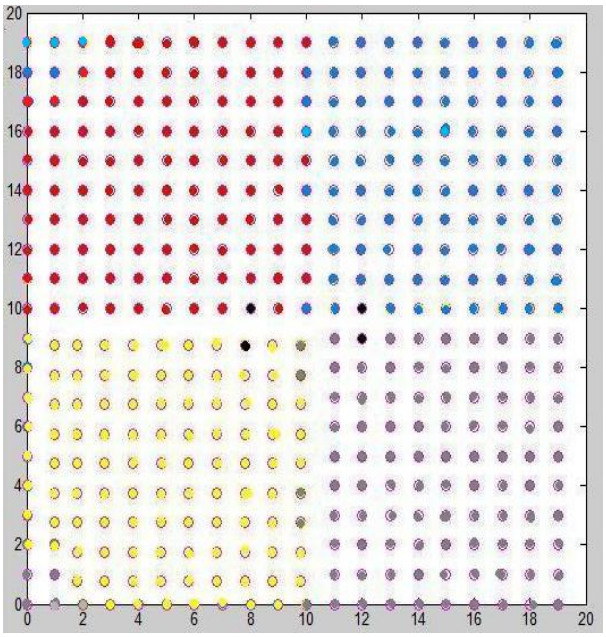


Fig.12. Redistribution with cost of 3205 in PDA.

Data generators at one corner of the network. Figure 13-17 shows the visual plot of GA, HA, CA, RA and PDA. Here we consider the nodes (0, 19), (3, 19), (0, 17) and (3, 17) as the data generators and we marked them as the black nodes. The nodes in red represent the destination set for the data generator at (0, 19), blue nodes are the destination set for the data generator at (3, 19), yellow nodes represent destination set for the data generator at (0, 17) and grey is the destination set for the data generator at (3, 17).

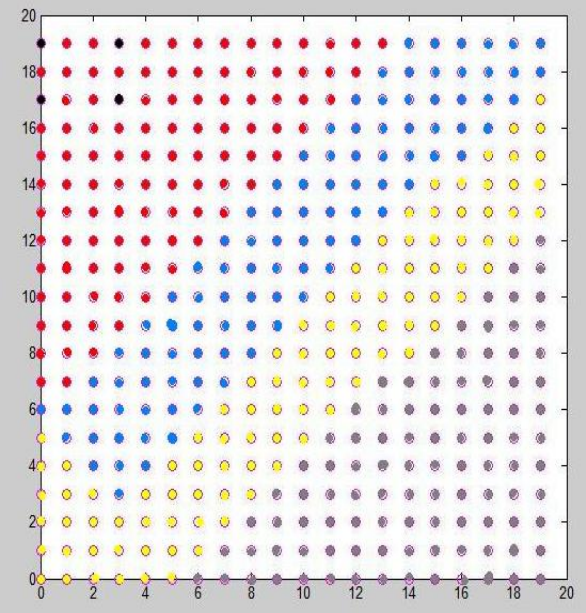


Fig.13. Redistribution with cost of 6780 in GA.

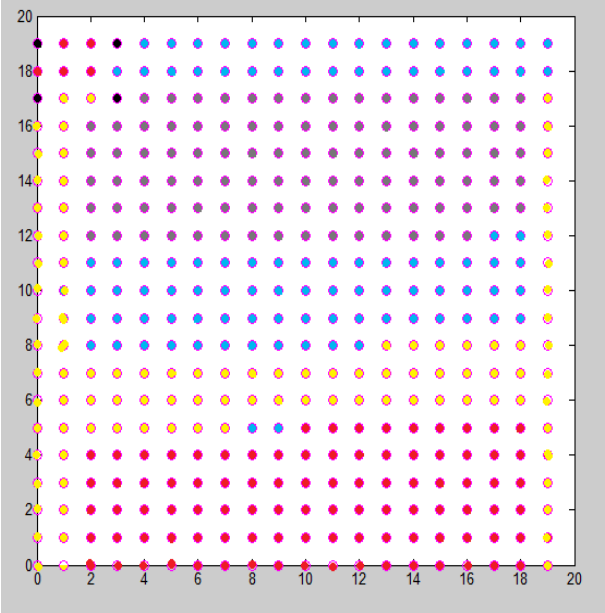


Fig.14. Redistribution with cost of 6600 in HA.

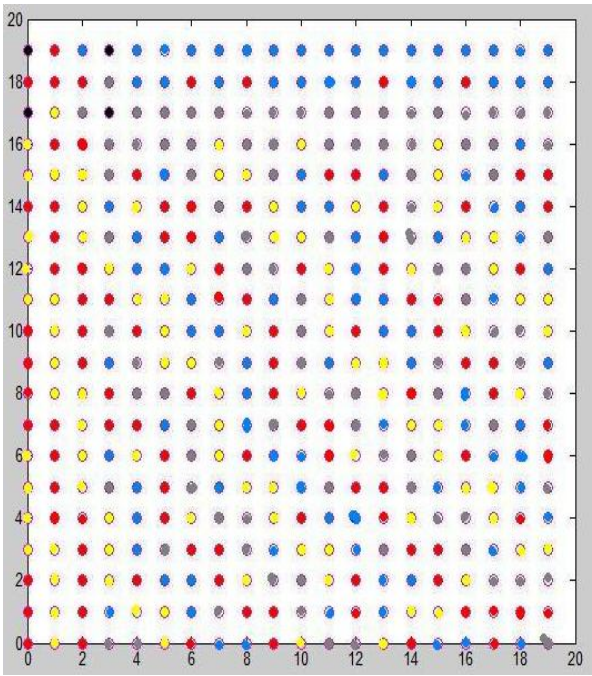


Fig.15. Redistribution with cost of 6690 in CA.

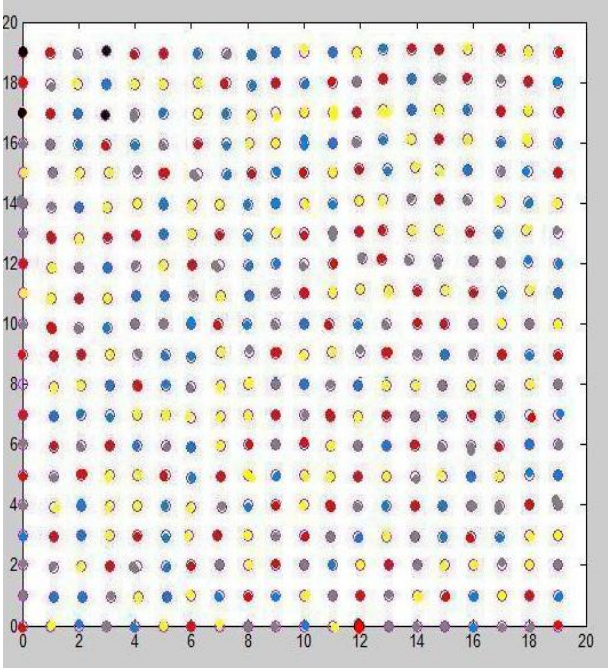


Fig.16. Redistribution with cost of 8770 in RA.

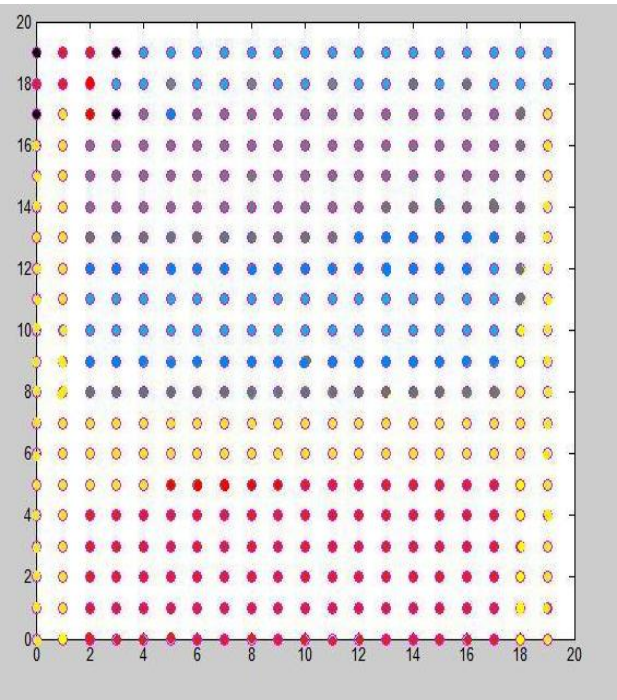


Fig.17. Redistribution with cost of 6633 in PDA.

Data generators randomly distributed over the network. Figure 18-22, shows the visual plot of GA, Hungarian algorithm, CA, RA and PDA respectively. Here we consider nodes (0, 18), (5, 7), (18, 0) and (12, 10) as data generators and we marked them as black nodes. The nodes in red represent the destination set for the data generator at (0, 18), blue nodes are the destination set for the data generator at (12, 10), yellow nodes represent destination set for the data generator at (5, 7) and grey is the destination set for the data generator at (18, 0).

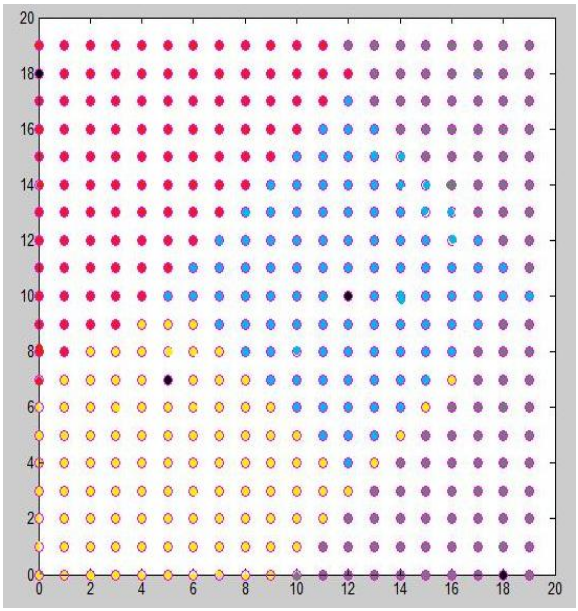


Fig.18. Redistribution with cost of 2964 in GA.

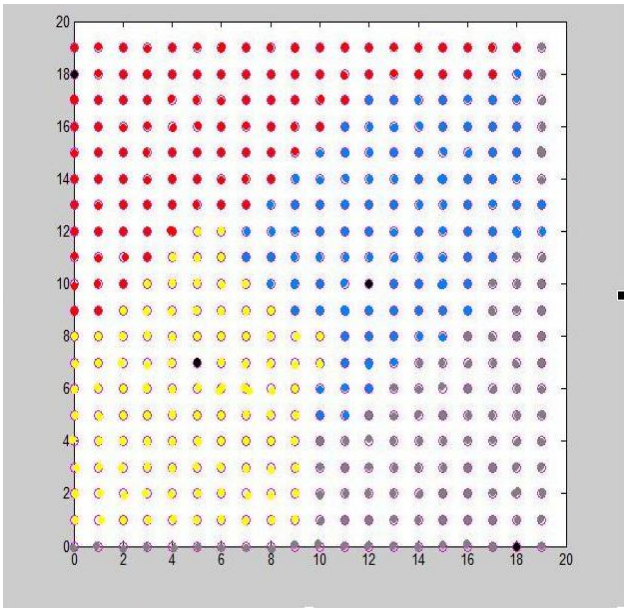


Fig.19. Redistribution with cost of 2698 in HA)

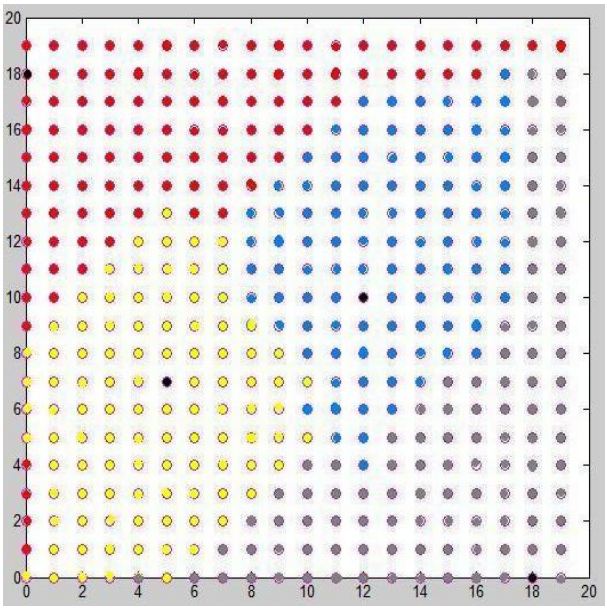


Fig.20. Redistribution with cost of 2790 in CA.

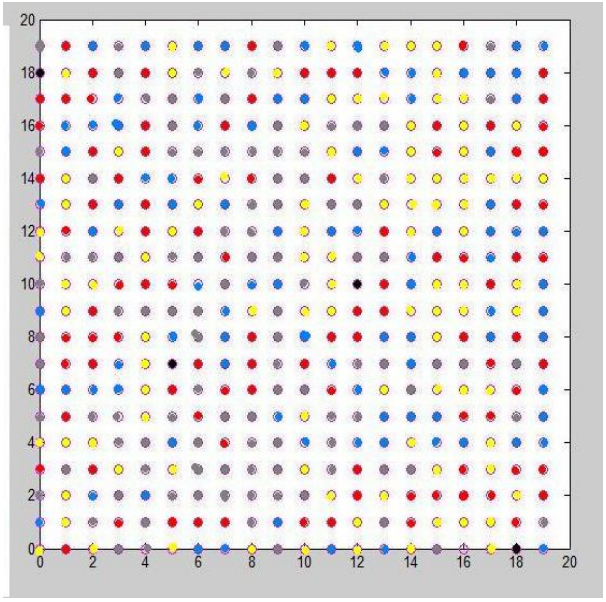


Fig.21. Redistribution with cost of 5318 in RA.

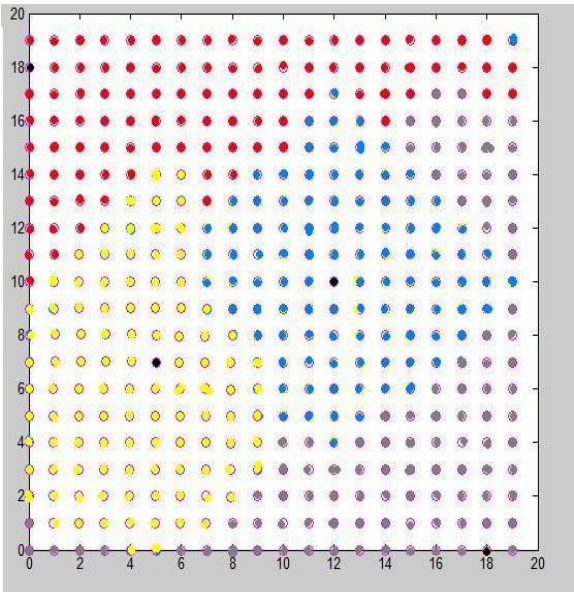


Fig.22. Redistribution with cost of 2742 in PDA.

Observation. Table 2-4 is the summary of total redistribution cost of all the data generators at the center of the network, at one corner and randomly distributed data generators respectively. We have observed with our simulations that our proposed Potential field based distributed algorithm performs better when compared to GA, CA and RA and it is comparable to the Hungarian algorithm, which is a centralized algorithm. In some cases even CA performs close to Hungarian algorithm and PDA as there is cooperation between the data generators to offload the data items. From the above figures we can visualize that our PDA algorithm performs close to the optimal Hungarian algorithm. Thus, we minimize the overall energy consumption of the network and maximize the storage utilization of the network using PDA. We calculate the total redistribution cost as the total sum of the hop numbers from the data generators to their respective destination set.

Table 2: Redistribution cost for the data generators at the center of network.

Algorithm	Redistribution Cost (In terms of hops)
Greedy algorithm	3524
Hungarian algorithm	3160
Cooperative algorithm	3200
Random algorithm	5235
Potential field based distributed algorithm	3205

Table 3: Redistribution cost for the data generators at one corner of network.

Algorithm	Redistribution Cost (In terms of hops)
Greedy algorithm	6780
Hungarian algorithm	6600
Cooperative algorithm	6690
Random algorithm	8770
Potential field based distributed algorithm	6633

Table 4.Redistribution cost for the data generators randomly distributed over the network.

Algorithm	Redistribution Cost (In terms of hops)
Greedy algorithm	2964
Hungarian algorithm	2698
Cooperative algorithm	2790
Random algorithm	5318
Potential field based distributed algorithm	2742

In figure 23-25, we show the total redistribution cost of the four data generators located at one corner, center and randomly distributed over 20 × 20 network.

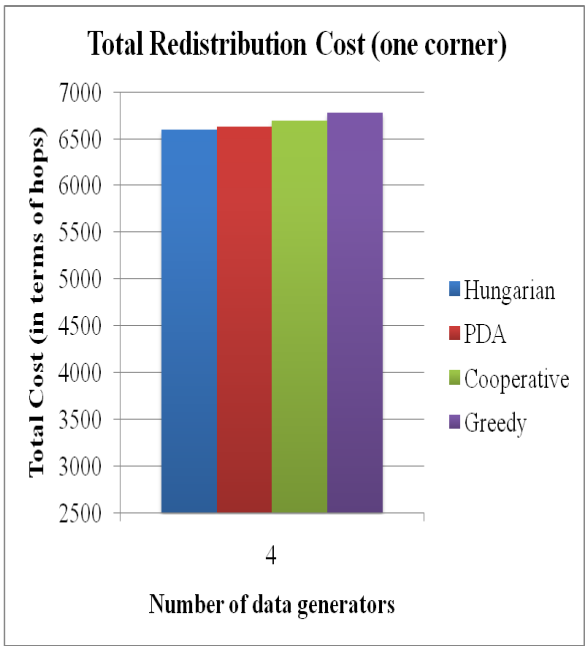


Fig. 23 Data generators located at one corner.

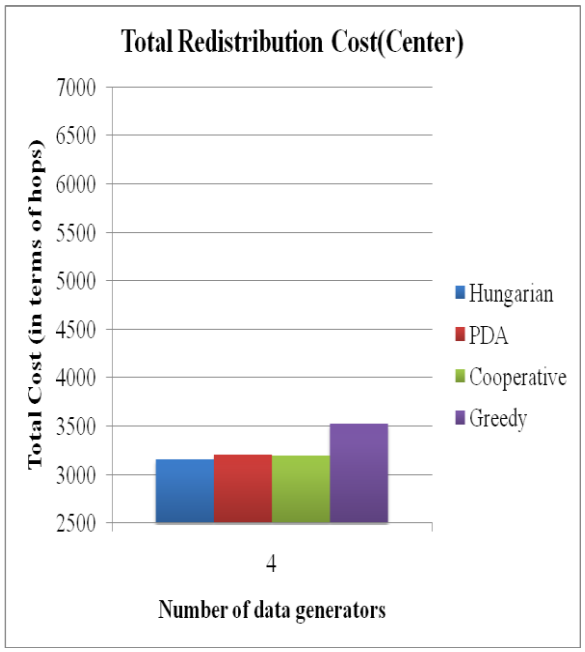


Fig. 24 Data generators located at center.

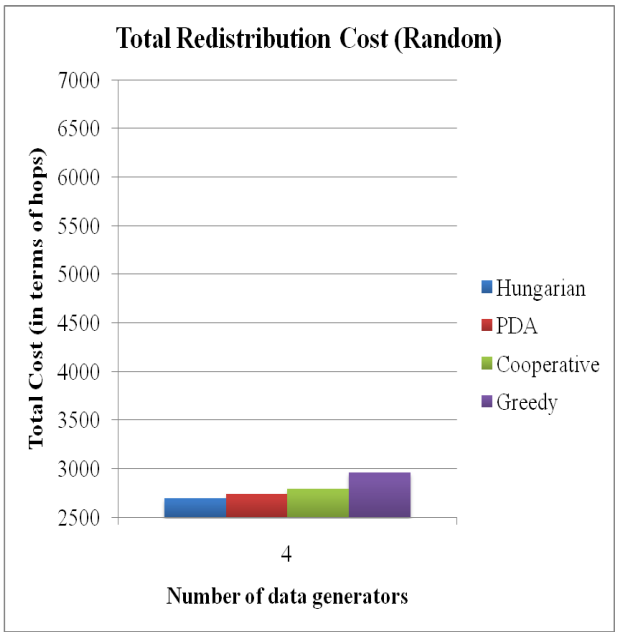


Fig. 25 Data generators randomly distributed.

Varying number of data generators and data items to be redistributed. In Figure 26 we vary the number of data generators deployed in a 100 × 100 network and also vary the maximum amount of data items to be redistributed over the entire network. We vary the data generators from 20, 40, 60 to 80 and maximum data items from 50, 70 to 90. We observe that PDA performs comparable to Hungarian algorithm and Greedy Algorithm performs worse than CA. The difference between the redistribution algorithms is seen clearly when there are more number of data generators with highest amount of data items to be redistributed. As the data generators and the maximum data items increases there arises the need of a suitable algorithm to avoid the contention and to reduce the total redistribution cost of the entire network.

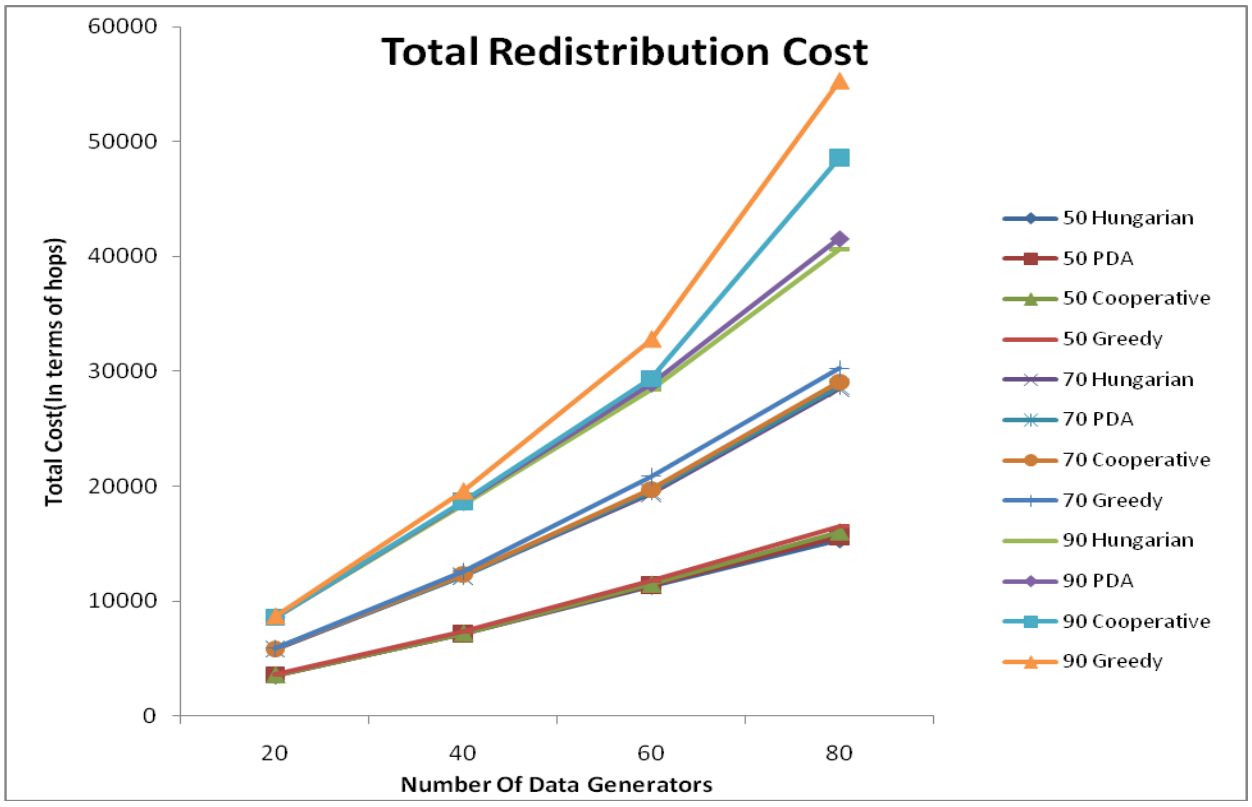


Fig.26.Total redistribution cost for 100 × 100 network with varying data generators and varying data items to be redistributed.

In Figure 27, we show how the RA performs worse and it not suitable for the redistribution of the data items in any size of network. Here we consider 100 × 100 network and vary the data generators as well as the maximum data to be redistributed. We compare RA with GA as the latter performs worse when compared to Hungarian algorithm, PDA and CA because the data generator with lower id selects nearest neighbors for offloading the data where the data generators with higher id incurs high redistribution cost but performs very well when compared to RA.

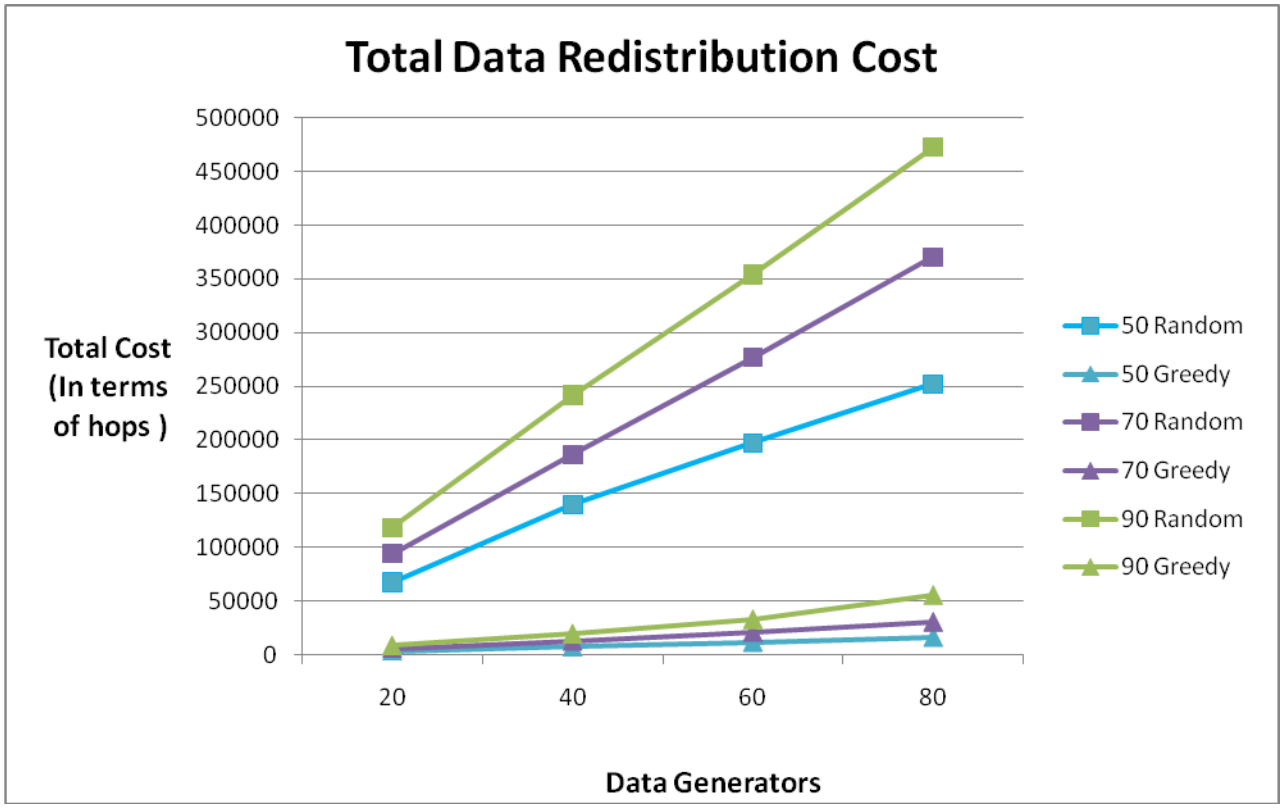


Fig.27. Total redistribution cost for 100 × 100 network for GA and RA.

Performance differential calculations. Figure 28 shows the performance percentage differential of GA, CA, PDA and RA for a network of 20 × 20 with four data generators at corner, center and random locations each with 99 data items to redistribute (Figures 8-12).

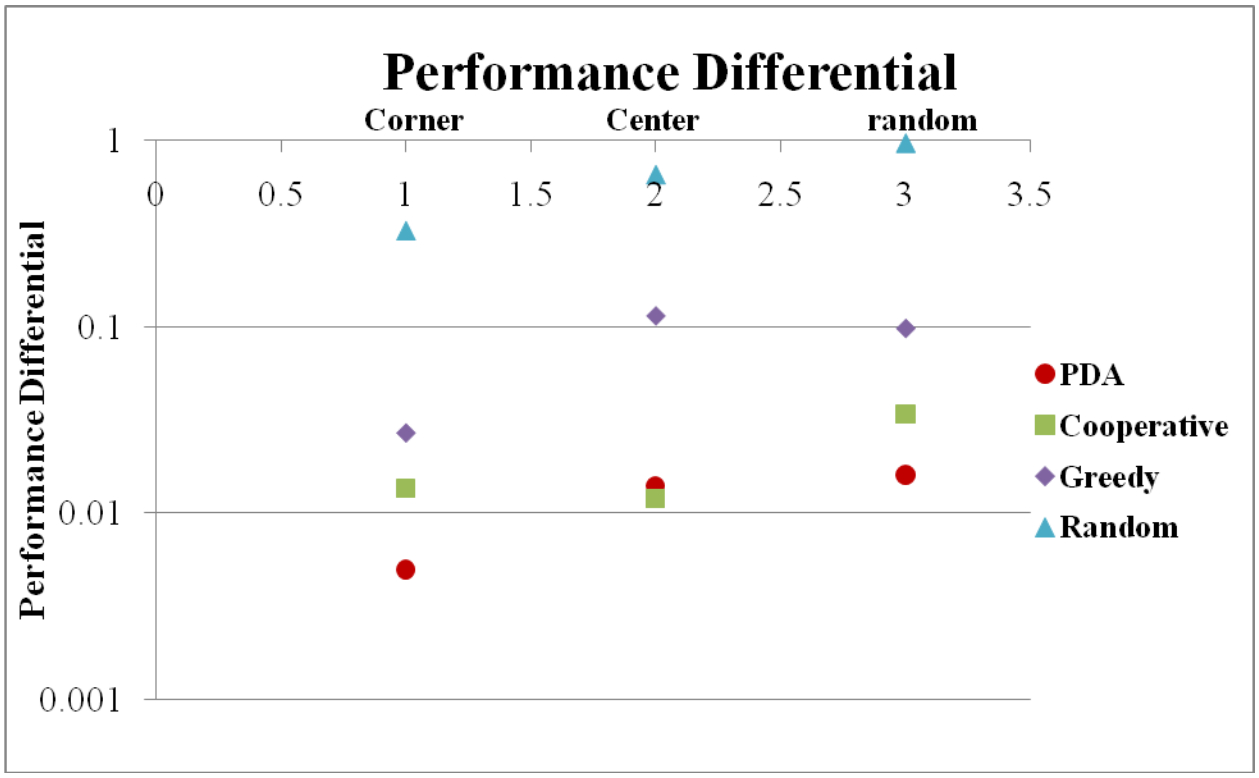


Fig. 28. Performance differential calculation for 20 × 20 network.

Observation: Here we observe that our PDA algorithm performs well when compared to all other algorithms in all the scenarios as it is completely distributed. PDA and CA are close to each other when the data generators are at the center of the network. RA performs worse when compared to all the algorithms.

In figures 29-31, we show the performance percentage differential of GA, CA, PDA and RA for 100 × 100 network by varying number of data items to be redistributed and by varying number of data generators from 20, 40, 60, and 80 for the Figure 26.

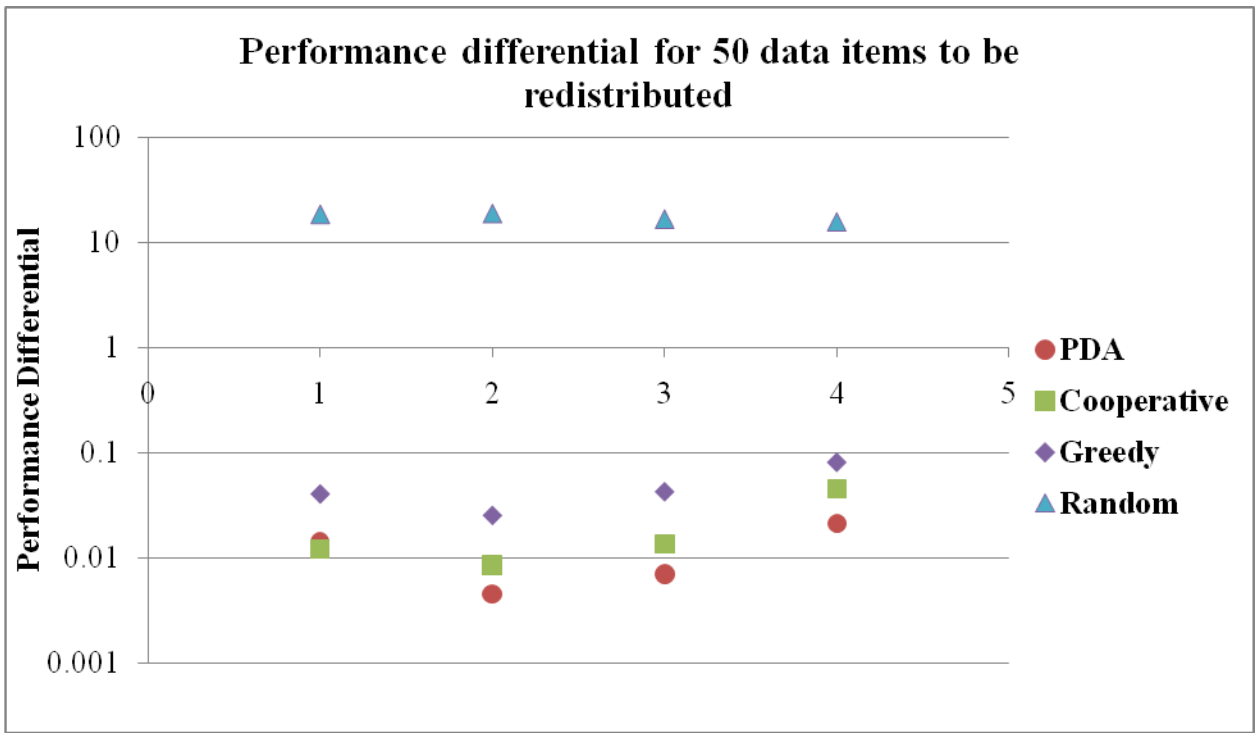


Fig. 29. Performance differential for 50 data to be redistributed.

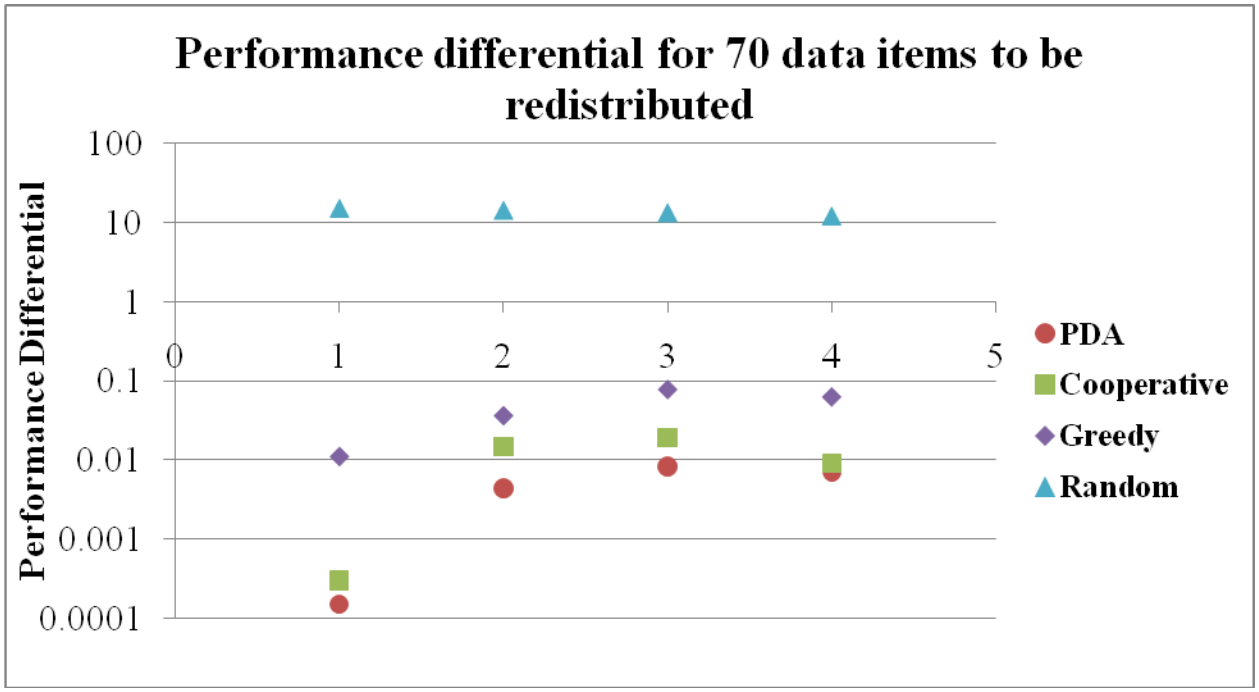


Fig. 30. Performance differential for 70 data to be redistributed.

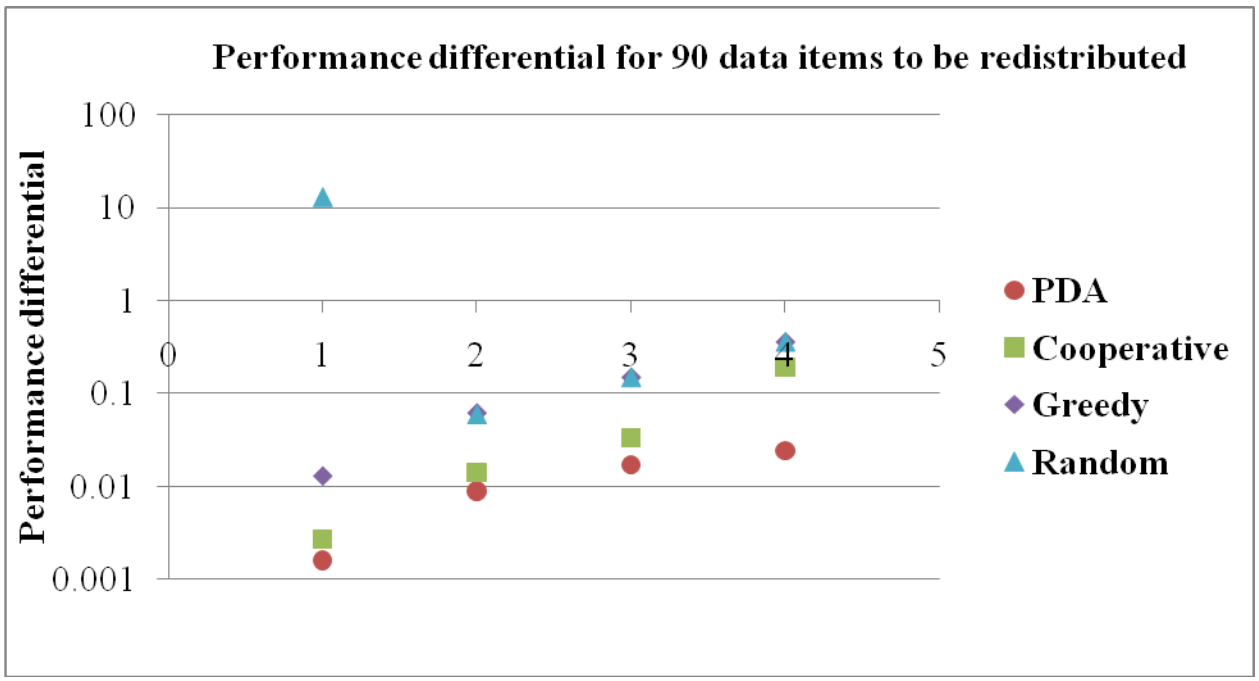


Fig. 31. Performance differential for 90 data to be redistributed.

Observation: We observe that RA also performs close to GA when the number of data items to be redistributed is 90. CA performs close to PDA in some cases as all the data generators are given equal priority to offload their data items. Thus, we state that our centralized heuristics algorithms and the distributed algorithm perform close to the optimal Hungarian algorithm.

Chapter 5

Conclusion and Future work

5.1 Conclusion

We study the data redistribution problem in sensor networks. Our results are two-fold. First, we show that the data redistribution problem is equivalent to the classic assignment problem, which can be solved optimally in a centralized manner. Second, for a distributed algorithm, we have applied the idea of electrostatic potential field to develop a distributed data redistribution mechanism. Through our own simulations, we show that our distributed algorithm is competitive to the centralized algorithm.

5.2 Future Work

As future work, we plan to pursue in the following directions. First, data collected in sensor networks usually display temporal and spatial correlation. Instead of offloading all the over flown data, such sensory data characteristic should be exploited so that redistributed energy consumption can be further reduced while preserving all the sensory information well. Second, our data redistribution algorithms so far are offline algorithms, meaning that the number of data to be offloaded by data generators is known at the beginning of the experiment, and dynamic node failures are not a concern. We would like to study more robust dynamic online algorithms wherein data is generated dynamically and network topology could change as the result of node failure, join, and departure and link failure. Last, we will explore the rational behavior of sensor nodes from game theoretical perspective. After all, the selfish sensor nodes, considering that they could become data generators later on, rather are cautious as to store data generated by others.

REFERENCES

LIST OF REFERENCES

[1] http://www.isi.edu/ilense/snuse/index.html.Example of sensor networks for underwater seismic [Date retrieved: 12 December 2009]

[2] G. Aathur, P. Desnoyers, D. Ganesan, and P. Shenoy. Ultra-low power data storage for sensor networks. In *Proc. of the 5th International Conference on Information Processing in Sensor Networks (IPSN '06)*, pages 374–381.

[3] Martin Erwig and Fernuniversitat Hagen. The graph voronoi diagram with applications. *Networks*, 36:156–163, 2000.

[4] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In Proc. of the 33rd Hawaii International Conference on System Sciences (HICSS '00), page 8020.

[5] K. R. Chowdhury I. F. Akyildiz, T. Melodia. A survey on wireless multimedia sensor networks. *Computer Networks (Elsevier)*, 51(4):921–960, 2007.

[6] Samir Khuller and Yoo ah Kim. Algorithms for data migration with cloning. In *SIAM Journal on Computing*, pages 27–36. ACM Press, 2003.

[7] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 1955.

[8] V. Lenders, M. May, and B. Plattner. Density-based anycast: A robust routing strategy for wireless ad hoc networks. *IEEE/ACM Transactions on Networking*, 16:852–863, 2008.

[9] S. Li, Y. Liu, and X. Li. Capacity of large scale wireless networks under gaussian channel model. In *Proc. of the ACM MobiCom 2008*.

[10] L. Luo, Q. Cao, C. Huang, L. Wang, T. Abdelzaher, and J. Stankovic. Design, implementation, and evaluation of enviromic: A storage-centric auio sensor network. *ACM Transactions on Sensor Networks*, 5(3):1–35, 2009.

[11] L. Luo, C. Huang, T. Abdelzaher, and J. Stankovic. Envirostore:A cooperative storage system for disconnected operation in sensor networks. In *Proc. of IEEE Infocom*, pages 1802 – 1810, 2007.

[12] K. Martinez, R. Ong, and J.K. Hart. Glacsweb: a sensor network for hostile environments. In *Proc. of IEEE 1st International Conference on Sensors and Ad-hoc networks (SECON 2004)*, pages 81–87.

[13] N. T. Nguyen, A.-I. A. Wang, P. Reiher, and G. Kuenning. Electricfield-based routing: A reliable framework for routing in manets. *ACM Mobile Computing and Communications Review*, 8(2):35–49, 2004.

[14] A. Pinar and B. Hendrickson. Interprocessor communication with limited memory. *IEEE Transactions on Parallel and Distributed Systems*, 15:606–616, 2004.

[15] Rahul C. Shah, Sumit Roy, Sushant Jain, and Waylon Brunette. Data mules: Modeling a three-tier architecture for sparse sensor networks. In *Proc. of IEEE International Workshop on Sensor Network Protocols and Applications (SNPA 2003)*, pages 30– 41.

[16] D.B. Shmoys and E. Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical Programming*, 62(3):461–474, 1993.

[17] Stephen F. Siegel and Andrew R. Siegel. Madre: The memoryaware data redistribution engine. In *Proc. of PVM/MPI (Recent Advances in Parallel Virtual Machine and Message Passing Interface)*, pages 218 – 226, 2008.

[18] S. Soro and W. Heinzelman. A survey of visual sensor networks. *Advances in Multimedia*, 2009.

[19] S. Toumpis. Mother nature knows best: A survey of recent results on wireless networks based on analogies with physics. *Computer Networks*, 52:360–383, 2008.

[20] I. Vasilescu, K. Kotay, D. Rus, M. Dunbabin, and P. Corke. Data collection, storage, and retrieval with an underwater sensor network. In *Proc. of the 3rd international conference on Embedded networked sensor systems (SenSys 2005)*, pages 154–165.

[21] Geoff Werner-Allen, Konrad Lorincz, Jeff Johnson, Jonathan Lees, and Matt Welsh. Fidelity and yield in a volcano monitoring sensor network. In *Proc. of 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2006.
1–474, 1993.