

We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800

Open access books available

122,000

International authors and editors

135M

Downloads

Our authors are among the

154

Countries delivered to

TOP 1%

most cited scientists

12.2%

Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



Parallel Processing for Range Assignment Problem in Wireless Sensor Networks

M. Prasanna Lakshmi and D. Pushparaj Shetty

Abstract

Wireless sensor network is a collection of autonomous devices called sensor nodes which sense the environmental factors such as temperature, pressure, humidity, moisture, etc. The nodes sense the data, process it and transmit to the other nodes within their transmission range through radio propagation. Energy minimization in wireless sensor networks is a significant problem since the nodes are powered by a small battery of limited capacity. In case of networks with several thousand nodes, the simulation of algorithms can be very slow. The parallel computing model provides significantly faster simulation time for larger networks. Parallel processing involves executing the program instructions by dividing them among multiple processors with the objective of reducing the running time. So, we propose algorithms for the range assignment problem in wireless sensor networks using the parallel processing techniques. We also discuss the complexity of the proposed algorithms and significance of the parallel processing techniques in detail. The proposed techniques will be useful for implementing the distributed algorithms in WSNs.

Keywords: range assignment, parallel processing, minimum spanning tree, wireless sensor networks, algorithm, complexity

1. Introduction

In this section, preliminaries of Wireless Sensor Networks (WSNs) and parallel processing are discussed in detail.

1.1 Wireless sensor networks

Wireless Sensor Network (WSN) is a group of spatially dispersed autonomous devices called sensor nodes equipped with a radio transceiver along with an antenna. The devices are responsible for sending and receiving radio signals. Transmission range is assigned to the nodes to facilitate the communication between the nodes of a network. The sensor nodes sense the physical conditions of the environment such as temperature, pressure, sound, heat, light and humidity, etc., at various locations, process the data and transmit to the other nodes that lie within their transmission range through radio propagation. WSN monitors a physical system in real world and has applications in several fields such as environmental

monitoring, biological detection, military and health care, etc. [1]. Because of the power constraints for the nodes using batteries with limited capacity, minimization of energy consumption is a significant problem in WSNs [2].

Constructing an efficient topology by assigning transmission range to the nodes of a network to minimize the total energy consumption has become a major challenge in WSNs. In general, the transmission range assigned to a node can be tuned so that the required connectivity constraint for the resultant network is satisfied and the total energy is minimized; this class of problems is categorized as topology control problems [3]. Some of the specified constraints include simple connectivity, bi-connectivity, k -connectivity, etc. [4]. By using an appropriate topology, the energy utilization of the network can be minimized which results in an increased lifetime of a WSN.

Transmission range of a node is the range within which the data sent by other node is received properly [5]. Two nodes in a WSN can communicate if and only if one node is in the transmission range of the other. Transmission range of a node u with the range r in three different dimensions is shown in the **Figure 1**. In general, the communication is multi hop in nature, and a node transmits the data to the destination node in its range using the relay nodes. In order to minimize the energy consumption, it is desirable to use short edges rather than the long energy-inefficient edges.

Let P_s be the range of source node u to transmit the signal to the destination node v . Then a signal transmitted by the source node u is attenuated by a factor: $P_r \propto \frac{P_s}{dist(u,v)^\alpha}$, where $dist(u,v)$ is the Euclidean distance between u and v , and α is the distance range gradient or path loss coefficient that depends on various environmental factors and generally lies between 2 and 6 [6]. In this chapter, the terms range and power are used interchangeably.

In reality, some problems use mathematical models that describe the problem in a systematic way and enable us to solve the problem efficiently. Graphs can be used to model many relations and represent many physical problems in the real world. A WSN is modeled as an undirected graph to solve the energy minimization problems. In this model, a vertex of the undirected graph represents a node and the edge joining two vertices represents the communication link between the nodes. The nodes are deployed on a Euclidean plane and each pair of two nodes is associated to the Euclidean distance between the nodes. Each node is located by its x and y coordinates and the distance function is used to find the Euclidean distance between any two nodes x and y , and is denoted by $w(xy)$ which is given by $w(xy) =$

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}.$$

Figure 2 illustrates the graph theoretic modeling of a WSN with four sensor nodes v_1, v_2, v_3 , and v_4 . In this, **Figure 2(a)** shows a network of four nodes with

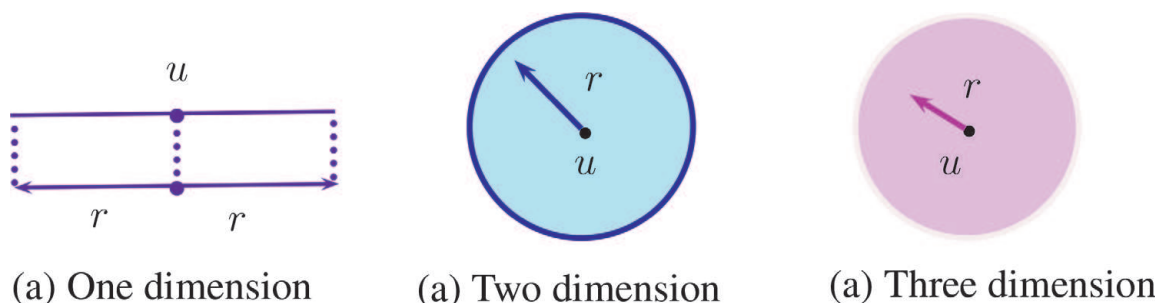


Figure 1.
Radio coverage in different dimensional networks.

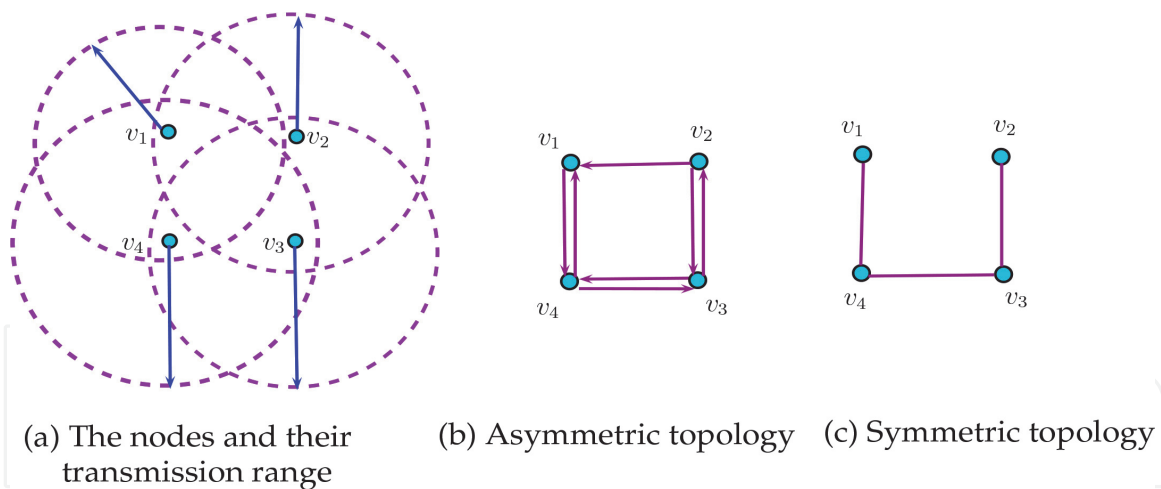


Figure 2.
 Illustration of graph theoretic modeling.

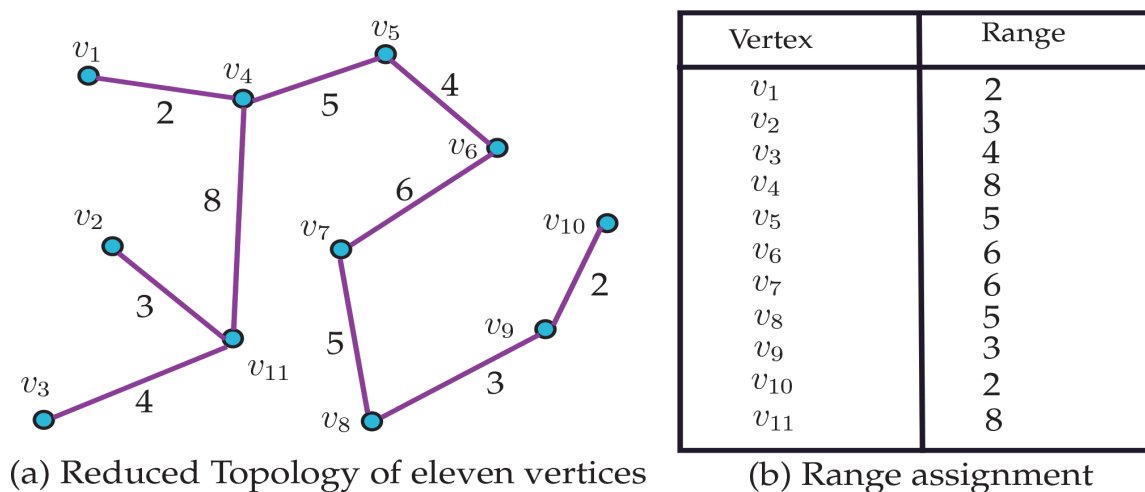


Figure 3.
 Illustration of range assignment.

their respective transmission ranges and its asymmetric and symmetric versions are shown in **Figure 2(b)** and **(c)**, respectively [7]. There is an edge between two vertices in symmetric graph if and only if it is bidirectional. A network is said to be bidirectional if and only if each edge is bidirectional and there exists a bidirectional path between every pair of two nodes of the network. Since the signal transmitted by a sensor node must be acknowledged by the receiver node, the bidirectional links are given preference.

Throughout this chapter, the considered topology is bidirectional. Therefore, we use an undirected graph along with the weight function w to represent a WSN. Once a WSN is formulated as an undirected graph $G = (V, E, w)$, the transmission range $R(v)$ assigned to each vertex $v \in G$ is the maximum of all its adjacent edge weights and is mathematically given as follows:

$$R(v) = \max \{w(vu) | uv \in E(G)\}. \quad (1)$$

The total range of the graph which is the sum of the ranges of all the vertices of G is given by $R(G) = \sum_{i=1}^n R(v_i)$. **Figure 3** is presented to illustrate the range assignment of a WSN. In this, **Figure 3(a)** shows a spanning tree of 11 vertices along with weights assigned to the edges and **Figure 3(b)** shows the range values assigned to the vertices as the maximum of all its adjacent edge weights in the spanning tree, as given in Eq. (1).

Minimum range assignment problem in a WSN is to assign transmission range to the nodes of the network such that the resultant subgraph H satisfies the specified constraints and the total range of the network, i.e., $R(H)$ is minimized [8]. Let H be a spanning subgraph of the given graph $G = (V, E, w)$, $R_H(v)$ be the range assigned to vertex $v \in V(H)$ as given in Eq. (1) and $R(H)$ be total range of the subgraph H . For various problems considered in this chapter, our interest is to find a spanning tree with minimum range.

Chen et al. [9] studied the problem of strong connectivity in multi hop packet radio networks and proposed a 2-approximation algorithm which initially considers an undirected Minimum Spanning Tree (MST) and later establishes the bidirectional connectivity. Strong Minimum Energy Topology problem (SMET) in a WSN is studied by Cheng et al. [10], in which the objective is to assign transmission range to the nodes of the network such that the total range of the network is minimized and the reduced topology is connected. The authors have proved that the SMET problem is NP-complete and proposed two algorithms namely: Minimum Spanning Tree (MST) based Heuristic which is of 2-approximation ratio and an Incremental Power Greedy Heuristic and performed the simulation to show that the Incremental Power Greedy Heuristic performs better than the MST Heuristic. Formulation of the SMET problem is as follows:

Problem: Strong Minimum Energy Topology problem (SMET).

Instance: A complete graph $G = (V, E, w)$, where $w : V \times V \rightarrow \mathbb{R}^+$ is a weight function.

Question: A range assignment such that the induced subgraph H is connected and the total range of H , i.e., $R(H)$ is minimized.

1.2 Parallel processing

Large scale WSNs such as environmental monitoring systems produce large amount of data in the order of Peta bytes. The challenges in this case are storage and computation as the sensors are constrained by their resources which are scarcely available [11]. If the large amount of data is processed centrally, all data needs to be transmitted to a central server using multi hop transmissions which leads to high computational cost. One of the best ways to avoid such problem is to exploit the advantages of the distributed storage and parallel processing. Instead of transmitting data to a central server, data is stored and processed in network which reduces the computational cost. In this process, the computational task is decomposed into many small tasks, and the computation is executed in parallel by the distributed sensors.

Sullivan et al. [12] proposed a set of new parallel algorithms for a tree decomposition-based approach to solve the maximum weighted independent set problem. Independent set problem for a given graph is to find a subset of vertices with maximum cardinality such that no two vertices in the graph are adjacent to each other. Maximum Weighted independent set problem in which, the vertices are assigned weights, is to find an independent set with maximum weight [13]. Zhu et al. [14] proposed parallel algorithm for the hierarchical-based protocol for WSNs based on the Low-Energy Adaptive Clustering Hierarchy (LEACH) algorithm in order to improve the routing efficiency of the networks. Authors have implemented the algorithms using the parallel processing technique and showed that this technique has increased the overall performance. Lounis et al. [15] presented a new model for parallel computing of energy consumption in WSNs, and implemented on GPU architecture. Through simulation, authors showed that the proposed model provides simulation times significantly faster than that of the sequential model for large networks and for long simulations.

Andoni et al. [16] gave algorithms for geometric graph problems in the modern parallel models. The authors found a $(1 + \epsilon)$ approximation algorithm for a

Geometric Minimum Spanning tree (MST) problem. In geometric MST, set of n points are plotted on a low dimension space such as R^d and other bounded metric spaces, in which weights of the edges are the distances between their endpoints. Authors also gave a general algorithmic framework that also applies to Earth-Mover Distance and the transportation cost problem. Katsigiannis et al. [17] presented a Helper Threading scheme for parallelizing the Kruskal's algorithm [18] which is used for finding a Minimum Spanning Forest (MSF). The implementation employs one main thread, which executes the serial algorithm, and several helper threads, that run in parallel and reduce the work of the main thread.

In this chapter, we discuss the parallel processing techniques for the range assignment problem for simple connectivity. We propose algorithms for this problem using parallel processing techniques and determine the complexity. The proposed algorithms initially find an MST and assign range to each vertex of the network based on its range in the Euclidean MST. We find the Euclidean MST using well known algorithms: Prim's and Kruskal's and later we assign the range using parallel processing technique for both the cases. We also discuss the complexity of the proposed algorithms in detail for both the algorithms.

Simulation is an act of imitating operation of real world process or system over time [19]. The simulation is performed on a model which represents the system. It is known that simulations closely reflect accurate behavior of the system. Simulations help in evaluating the algorithm before investing on the physical hardware.

2. Related work

Finding a Euclidean MST is an important problem in Graph Theory and there are several algorithms in the literature [20]. MST problems have significant applications in computer and communication network design, as well as indirect applications in fields such as computer vision and cluster analysis [21]. Parallelization of Prim's algorithm is studied by Deo and Yoo [22] and the proposed algorithms used shared memory computers. Gonina et al. [23] studied the parallel implementation of the Prim's algorithm for finding the MST of dense graphs. Authors presented the simulation results which demonstrate the advantage of proposed algorithms.

Vladimir et al. [24] studied serial variants of Prim's and Kruskal's algorithm and presented their parallelization, targeting message passing parallel machine with distributed memory. Authors have considered large graphs and presented experimental results which show that Prim's algorithm is a good choice for dense graphs while Kruskal's algorithm is better for sparse graphs. Authors have proposed algorithms based on the sequential algorithms of Prim and Kruskal, targeting message passing parallel machine with distributed memory. The proposed algorithm which uses the Prim's algorithm has a running time of $O\left(\frac{n^2}{k}\right) + O(n \log k)$ and the one which uses the Kruskal's algorithm has a running time of $O\left(\frac{n^2}{k}\right) + O(n^2 \log k)$ where n is the number of vertices and k is the number of partitioned subsets of V such that each subset has $\frac{n}{k}$ consecutive vertices and their edges. In this section, we discuss the parallel processing algorithms for computing Euclidean MST using both Prim's and Kruskal's algorithms separately. In both the algorithms, each processor gets a subset of vertices from the given set of vertices to process.

The considered input for the parallelization of both Prim's and Kruskal's algorithm by Vladimir et al. [24] is an undirected simple graph with weights assigned to the edges. For our research the input considered is a complete graph along with edge weights, where the edge weights are the Euclidean distance between the nodes.

2.1 Parallelization of Prim's algorithm by Loncar et al.

Prim's algorithm starts from an arbitrary vertex and then grows the subtree by choosing a new vertex from the set of unvisited vertices and adding it to the subtree in each iteration. The Prim's algorithm runs until the number of edges becomes $n - 1$ or all the vertices are added to the subtree. The complexity of the Prim's algorithm is $O(n^2)$ where n is the number of vertices of the given graph. In this algorithm, the input considered is an undirected graph with n vertices and m edges along with the edge weights in which weight of edge uv is denoted by $w(uv)$ [24].

The weights are represented using the adjacency matrix which is defined as follows:

$$a_{ij} = \begin{cases} w(v_i v_j) & \text{if } (v_i v_j \in E) \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

Prim's algorithm is implemented using the auxiliary array d of length n which stores the weights from each vertex to the MST. The lightest weight edge is chosen from d and is added to MST in each iteration and the array d is updated accordingly. The main loop of the Prim's algorithm cannot be parallelized since the minimum edge weights incident to MST change in each iteration. So, only two steps are parallelized: finding a minimum weight edge that connects a new vertex not in MST to a vertex in MST and updating the array d . The complexity of the proposed algorithm is $O\left(\frac{n^2}{k}\right) + O(n \log k)$ where n is the number of vertices and k is the number of processors.

2.2 Parallelization of Kruskal's algorithm by Loncar et al.

Kruskal's algorithm finds a minimum weighted edge which connects two components in the forest in each iteration, i.e., it grows multiple trees in parallel. Initially, Kruskal's algorithm creates a forest considering each vertex as a tree and next it sorts all the edges. Each time a minimum weighted edge is chosen and added to the forest if it connects two different trees else it is discarded. This process is repeated until the forest becomes a spanning tree. The complexity of the proposed algorithm is $O(m \log n)$ where m is the number of edges and n is the number of vertices [24].

Similar to the Prim's algorithm, we have adjacency matrix which stores the weights between the vertices, and each processor is assigned a subset of vertices. In parallelization of Kruskal's algorithm, each processor sorts the edges of its vertices according to their weights. At each process, a local MST is found using Kruskal's algorithm and finally, all the local MST's are merged. The complexity of the proposed algorithm is $O\left(\frac{n^2}{k}\right) + O(n^2 \log k)$ where n is the number of vertices and k is the number of processors.

3. Parallelization of range assignment problem

In this section, we present the proposed algorithms for range assignment problem using parallel processing techniques. The proposed algorithms adopt the algorithms for parallelization of both Prim's and Kruskal's algorithms by Loncar et al. [24]. In this environment, the nodes are deployed on a Euclidean plane and each pair is associated with the Euclidean distances between those nodes. Since a WSN is modeled as an undirected graph along with a weight function, the input for these algorithms is a complete graph $G = (V, E, w)$ where the weight function is as explained in Section 1. The range assigned to a vertex v is denoted by $R(v)$.

Objective of this problem is to find a spanning tree such that the total range of the spanning tree is minimized. In the proposed algorithms, we initially find the spanning tree of the given complete graph and assign the range to the vertices based on their range in the Euclidean MST using parallel processing. We propose two algorithms: one using the Prim's algorithm and the other using the Kruskal's algorithm. We also compute the complexity of the proposed algorithm in both the cases. The list of notations used in this chapter is given in **Table 1**.

3.1 Range assignment using Prim's algorithm

Let $G = (V, E, w)$ be the given complete graph. In this algorithm initially, we find a spanning tree by parallelizing the Prim's algorithm and range is assigned to the vertices based on their range in the MST by using parallel processing. The formulation of the problem is as follows:

Problem: Parallel Algorithm for Range assignment using Prim's algorithm (PARA-Prim's).

Input: A complete graph $G = (V, E, w)$, where w is a weight function and P the set of processors.

Objective: To find a spanning tree such that the total energy is minimized, using parallel processing.

In the proposed algorithm, we consider a complete graph and the number of processors as input and compute a spanning tree which minimizes the total range. In this algorithm, we partition the vertices into k groups and each processor gets $\frac{n}{k}$ consecutive vertices and their edges. Each processor also contains auxiliary array d of the vertices which maintains the distances of the vertices to the MST. Let V_i and d_i be the set of vertices and the auxiliary array of the processor p_i , respectively.

Each process finds a minimum weighted edge which connects the MST with vertices of the set V_i and sends to the leader processor using all-to-one reduction. The leader processor selects one with the minimum weight among all the received edges, adds it to MST and broadcasts to all the processors. Processors mark the vertices connected to MST and update their auxiliary array d and this process is repeated until all the vertices are added to the subtree.

Next, we assign range to the vertices in the following way: for a vertex, each process finds the farthest vertex to it in the MST and sends the distance between

Notation	Explanation
V	Set of vertices
E	Set of edges
n	Number of nodes
$w(uv)$	Euclidean distance between u and v
$R(v)$	Range of vertex v
d	Auxiliary array
P	The set of processors
p_i	i^{th} processor
k	Number of processors
T	Resultant spanning tree

Table 1.
List of notations.

them to the leader processor using all-to-one reduction. Later, leader processor selects one with maximum edge weight among all the received edge weights and that weight is assigned as range to the vertex v . This process of range assignment is repeated for all the vertices. The sequence of steps is presented in Algorithm 1.

Theorem 1.1 [24] The parallelization of Prim's algorithm takes $O\left(\frac{n^2}{k}\right) + O(n \log k)$ running time where n is the number of nodes and k is the number of processors.

Algorithm 1: PARA-Prim's	
1	begin
2	Let $ P = k, V = n, T = \phi, j = 1, Visited[i] = 0$ for $i = 1$ to n
3	for $i = 1$ to k do
4	$p_i = \{v_j, v_{j+1}, \dots, v_{j+\frac{n}{k}}\}$
5	$j = \frac{n}{k} + 1$
6	end
7	while $ V(T) < n - 1$ do
8	for $i = 1$ to k do
9	for $j = 1$ to $\frac{n}{k}$ do
10	$min = \infty$
11	for $k = 1$ to $\frac{n}{k}$ do
12	if $p_i[j] \neq p_i[k] \ \&\& \ visited[p_i[j]] \neq 0 \ \&\& \ w(p_i[j]p_i[k]) < min$
13	then
14	$e[i] = p_i[j]p_i[k]$
15	end
16	end
17	end
18	for $i = 1$ to k do
19	if $w(e[i]) < w(e_{min})$ then
20	$e_{min} = uv = e_i$
21	end
22	end
23	if $Visited[u] = 0 \ \ Visited[v] = 0$ then
24	$T = T \cup \{e_{min}\}$
25	$Visited[v] = 1$
26	update the corresponding $d[v]$
27	end
28	for $i = 1$ to n do
29	for $j = 1$ to k do
30	$max[i] = 0$
31	for $k = 1$ to $\frac{n}{k}$ do
32	if $v_i \neq p_j[k] \ \&\& \ v_i p_j[k] \in T \ \&\& \ w(v_i p_j[k]) > max$ then
33	$max[i] = w(v_i p_j[k])$
34	end
35	end
36	end
37	$max = max[1]$
38	for $i = 1$ to k do
39	if $max[i] > max$ then
40	$max = max[i]$
41	end
42	end
43	$R[v_i] = max$
44	end
45	end
46	return T, R
47	end

Theorem 1.2 The algorithm PARA-Prim's takes $O\left(\frac{n^2}{k}\right) + O(n \log k)$ running time, where n is the number of nodes and k is the number of processors.

Proof: From Theorem 1.1, it is clear that finding the MST by parallelizing the Prim's algorithm takes $O\left(\frac{n^2}{k}\right) + O(n \log k)$ running time for n number of nodes and k number of processors. In PARA-Prim's algorithm the range assignment is done based on the MST obtained by parallelizing the Prim's algorithm. For each vertex, we find the farthest vertex in each processor which takes the running time of $O\left(\frac{n}{k}\right)$ and for all such vertices it takes a running time of $O\left(\frac{n^2}{k}\right)$. Next each processor sends the farthest vertex to the leader process and leader processor finds the global farthest vertex which takes a running time of $O\left(\frac{n}{k}\right)$ and for all such n vertices it takes a running time of $O\left(\frac{n^2}{k}\right)$. So the running time of PARA-Prim's algorithm is $O\left(\frac{n^2}{k}\right) + O(n \log k)$.

3.2 Range assignment using Kruskal's algorithm

Similar to the Prim's algorithm in this algorithm, we have k number of processors assigned with $\frac{n}{k}$ set of consecutive vertices. Each processor sorts the edges of its vertices and finds a local MST using the Kruskal's algorithm. Finally all the local MSTs are merged in the following way: first processor sends its set of edges of the local MST to the second processor and forms a new local MST by using the set of edges from both the processors. Now, the first process remains no longer and terminates, and this process of merging continues until single process remains. Range assignment is done to the vertices in parallel way similar to the PARA-Prim's algorithm as explained in the previous subsection. The formulation of the problem is as follows:

Problem: Parallel Algorithm for Range assignment using Kruskal's algorithm (PARA-Kruskal's).

Input: A complete graph $G = (V, E, w)$, where w is a weight function and P the set of processors.

Objective: To find a spanning tree such that the total energy is minimized, using parallel processing.

The sequence of steps for the above explained procedure is presented in Algorithm 2 named PARA-Kruskal's.

Theorem 1.3 [24] The parallelization of Kruskal's algorithm takes $O\left(\frac{n^2}{k}\right) + O(n^2 \log k)$ where n is the number of vertices and k is the number of processors.

Theorem 1.4 The algorithm PARA-Kruskal's takes $O\left(\frac{n^2}{k}\right) + O(n^2 \log k)$ running time, where n is the number of nodes and k is the number of processors.

Proof: From Theorem 1.3, it is clear that, the running time of the parallelization of Kruskal's algorithm is $O\left(\frac{n^2}{k}\right) + O(n^2 \log k)$. Next, we do the range assignment to the vertices based on its range in the MST formed by parallelizing the Kruskal's algorithm. From Theorem 1.3, it is clear that the running time of the range assignment for all the vertices is $O\left(\frac{n^2}{k}\right)$. So, the running time of the PARA-Kruskal's algorithm is $O\left(\frac{n^2}{k}\right) + O(n^2 \log k)$.

Algorithm 2: PARA-Kruskal's

```

1 begin
2   Let  $|P| = k, |V| = n$ 
3   Follow the steps from 3 to 5 of Algorithm 1
4   for  $i = 1$  to  $k$  do
5      $T_i = \Phi$ 
6     Let  $E_i$  be the set of all edges of the vertices of  $p_i$ 
7     Sort the edges of  $E_i$  according to their edge weights
8     while  $|V(T_i)| < \frac{n}{k} - 1$  do
9       for  $j = 1$  to  $\frac{n}{k}$  do
10         $min = \infty$ 
11        for  $k = 1$  to  $\frac{n}{k}$  do
12          if  $w(p_i[j]p_i[k]) < min$  then
13             $min = w(p_i[j]p_i[k])$ 
14             $a = p_i[j]$ 
15             $b = p_i[k]$ 
16          end
17        end
18      end
19      if  $T_i \cup \{ab\}$  is acyclic then
20         $T_i = T_i \cup \{ab\}$ 
21      end
22    end
23  end
24  for  $i = 1$  to  $k - 1$  do
25    Let  $A$  be the set of all edges of  $T_i$ 
26    Let  $B$  be the set of all edges of  $T_{i+1}$ 
27     $p_i$  sends  $A$  to  $p_{i+1}$ 
28     $p_{i+1}$  finds  $T$  as a new local MST from  $A \cup B$ 
29  end
30  Repeat the steps from 28 to 44 of Algorithm 1
31  return  $T, R$ 
32 end

```

4. Comparison with the state of arts

Solving the range assignment problem by computing Euclidean MST using Prim's algorithm and Kruskal's algorithm takes running time of $O(n^2)$ and $O(n^2 \log n)$, respectively. For WSNs with large number of sensor nodes, this complexity is quite high which can be improved by using parallel processing techniques. Simulation can be performed on hardware that supports multiple cores in order to realize the improvements over serial programming environments. The complexities of the range assignment problem using Prim's algorithm and Kruskal's algorithm by employing parallel processing techniques are $O\left(\frac{n^2}{k}\right) + O(n \log k)$ and $O\left(\frac{n^2}{k}\right) + O(n^2 \log k)$, respectively. So, theoretically there is a significant improvement in the complexity of the range assignment problem by applying parallel processing techniques. Simulation can be performed to study the stability of the proposed algorithms and to compare the proposed algorithms with the existing algorithms for the range assignment algorithms.

5. Conclusions

In this research, we have studied the range assignment problem by employing parallel processing techniques of both Prim's and Kruskal's algorithms.

The complexity of the proposed two algorithms is discussed in detail and it is shown that complexity can be improved by using parallel processing techniques for larger networks. It is an interesting problem to study variations of range assignment problems using parallel processing techniques. The implementation of the proposed algorithms can be realized using CUDA programming which supports programming GPU with multiple cores.

Abbreviations

WSN	wireless sensor network
MST	minimum spanning tree
MSF	minimum spanning forest
PARA	parallel algorithm for range assignment

Author details

M. Prasanna Lakshmi^{1*†} and D. Pushparaj Shetty^{2†}


1 Indian Institute of Information Technology, Sri City, Chittoor, India

2 National Institute of Technology Karnataka, Mangalore, India

*Address all correspondence to: prasannasainitw@gmail.com

† These authors are contributed equally.

IntechOpen

© 2019 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/3.0>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. 

References

- [1] Akyildiz IF, Su W, Sankarasubramaniam Y, Cayirci E. Wireless sensor networks: A survey. *Computer Networks*. 2002;**38**(4):393-422
- [2] Clementi, AE, Penna P, Silvestri R. The power range assignment problem in radio networks on the plane. In: *Annual Symposium on Theoretical Aspects of Computer Science*. Berlin, Heidelberg: Springer; February 2000. pp. 651-660
- [3] Lloyd EL, Liu R, Marathe MV, Ramanathan R, Ravi SS. Algorithmic aspects of topology control problems for ad hoc networks. *Mobile Networks and Applications*. 2005;**10**(1-2):19-34
- [4] West DB. *Introduction to Graph Theory*. Vol. 2. Upper Saddle River, NJ: Prentice hall; 1996
- [5] Santi P. Topology control in wireless ad hoc and sensor networks. *ACM Computing Surveys (CSUR)*. 2005; **37**(2):164-194
- [6] Carmi P, Chaitman-Yerushalmi L. On the minimum cost range assignment problem. In: *International Symposium on Algorithms and Computation*. Berlin, Heidelberg: Springer; 2015. pp. 95-105
- [7] Calinescu G, Wan PJ. Range assignment for high connectivity in wireless ad hoc networks. In: *International Conference on Ad-Hoc Networks and Wireless*. Berlin, Heidelberg: Springer; 2003. pp. 235-246
- [8] Fuchs B. On the hardness of range assignment problems. *Networks: An International Journal*. 2008;**52**(4):183-195
- [9] Chen WT, Huang NF. The strongly connecting problem on multihop packet radio networks. *IEEE Transactions on Communications*. 1989;**37**(3):293-295
- [10] Cheng X, Narahari B, Simha R, Cheng MX, Liu D. Strong minimum energy topology in wireless sensor networks: NP-completeness and heuristics. *IEEE Transactions on Mobile Computing*. 2003;**2**(3):248-256
- [11] Wang Y, Wang Y. Distributed storage and parallel processing in large-scale wireless sensor networks. In: *High Performance Computing Workshop*. Cetraro, Italy: IOS Press; 2010. pp. 288-305
- [12] Sullivan BD, Weerapurage D, GroÅnr C. Parallel algorithms for graph optimization using tree decompositions. In: *2013 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum*. IEEE; 2013. pp. 1838-1847
- [13] Minty GJ. On maximal independent sets of vertices in claw-free graphs. *Journal of Combinatorial Theory, Series B*. 1980;**28**(3):284-304
- [14] Zhu Y, Yao Q, George G, Wu S, Zhang C. Parallel LEACH algorithm for wireless sensor networks. In: *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp)*; 2012. p. 1
- [15] Lounis M, Bounceur A, Laga A, Pottier B. GPU-based parallel computing of energy consumption in wireless sensor networks. In: *2015 European Conference on Networks and Communications (EuCNC)*. IEEE; 2015, June. pp. 290-295
- [16] Andoni A, Nikolov A, Onak K, Yaroslavlsev G. Parallel algorithms for geometric graph problems. In: *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*. ACM; 2014. pp. 574-583

[17] Katsigiannis A, Anastopoulos N, Nikas K, Koziris N. An approach to parallelize Kruskal's algorithm using helper threads. In: 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum. IEEE; 2012. pp. 1601-1610

[18] Cormen TH, Leiserson CE, Rivest RL, Stein C. Introduction to Algorithms. Cambridge, Massachusetts, London, England: McGraw-Hill Book Company; 2009

[19] Sobeih A, Hou JC, Kung LC, Li N, Zhang H, Chen WP, et al. J-Sim: A simulation and emulation environment for wireless sensor networks. IEEE Wireless Communications. 2006;**13**(4): 104-119

[20] Rosen KH. Handbook of Discrete and Combinatorial Mathematics. Boca Raton, London, New York, Washington, DC: CRC Press; 2017

[21] Graham RL, Hell P. On the history of the minimum spanning tree problem. Annals of the History of Computing. 1985;**7**(1):43-57

[22] Deo N, Yoo YB. Parallel Algorithms for the Minimum Spanning Tree Problem. Computer Science Department: Washington State University; 1981

[23] Gonina E, Kalál' LV. Parallel Prim's Algorithm on Dense Graphs with a Novel Extension. 2007

[24] Lončar V, Åkrbic S. Parallel implementation of minimum spanning tree algorithms using MPI. In: 2012 IEEE 13th International Symposium on Computational Intelligence and Informatics (CINTI). IEEE; 2012. pp. 35-38