VOT 75083

SIMULATED ANNEALING TECHNIQUE FOR REDUCING COMMUNICATION IN MESH NETWORKS

ASSOC. PROF. DR. SHAHARUDDIN SALLEH

JABATAN MATEMATIK

FAKULTI SAINS

UNIVERSITI TEKNOLOGI MALAYSIA

2006

UNIVERSITI TEKNOLOGI MALAYSIA

BORANG PENGESAHAN
LAPORAN AKHIR PENYELIDIKAN

TAJUK PROJEK : **SIMULATED ANNEALING TECHNIQUE FOR REDUCING COMMUNICATION IN MESH NETWORKS.**

Saya PROF. MADYA. DR. SHAHARUDDIN SALLEH

Mengaku membenarkan Laporan Akhir Penyelidikan **ini disimpan di Perpustakaan Universiti Teknologi Malaysia dengan syarat-syarat kegunaan seperti berikut :**

1. Laporan Akhir Penyelidikan ini adalah hakmilik Universiti Teknologi Malaysia.
2. Perpustakaan Universiti Teknologi Malaysia dibenarkan membuat salinan untuk tujuan rujukan sahaja.
3. Perpustakaan dibenarkan membuat penjualan salinan Laporan Akhir Penyelidikan ini bagi kategori TIDAK TERHAD.
4. * Sila tandakan (/)

☐	SULIT	(Mengandungi maklumat yang berdarjah keselamatan atau Kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972).
☐	TERHAD	(Mengandungi maklumat TERHAD yang telah ditentukan oleh Organisasi/badan di mana penyelidikan dijalankan).
☐	TIDAK TERHAD	

TANDATANGAN KETUA PENYELIDIK

_____
Nama & Cop Ketua Penyelidik

Tarikh : _____

CATATAN : ***** *Jika Laporan Akhir Penyelidikan ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali sebab dan tempoh laporan ini perlu dikelaskan*

DEDICATION AND ACKNOWLEDGEMENT

This research has been conducted mostly at the Department of Mathematics, Faculty of Science, Universiti Teknologi Malaysia. The authors would like to thank the University for providing the moral and financial support in the form of the Fundamental Research Grant no. 75083 from the Ministry of Education.

ABSTRACT

This report is the compilation of our research work under Vot no. 75083 (Fundamental) at the Department of Mathematics, Universiti Teknologi Malaysia. The work consists of four main problems of study. First, we look at the single-row transformation of complete graphs into single rows. This problem involves the application of single-row routing into pair-matching problems such as channel assignments in the cellular telephone network systems. The problem is generalized and extended by involving the cliques of connected graphs for mapping and transforming these nodes into single-row nodes. We then discuss the wireless ad hoc network application on a mesh network for single-casting, multicasting and broadcasting using a model called FRECAST. Finally, the problem extends to a computational model called SPLAI using finite-element method for the wireless sensor networks. This segment of the work is capable of locating the sensor nodes in a network using a dynamic coordinate system based on the coronas and wedges. The output from the research project consists of one book, six journal papers and two proceedings papers (all at the international level), and four softwares.

ABSTRAK

Laporan ini adalah hasil daripada penyelidikan di bawah Vot 75083 (Fundamental) di Jabatan Matematik, Universiti Teknologi Malaysia. Kerja meliputi empat bahagian utma. Pertama, kita mengkaji masalah transformasi graf lengkap kepada baris tunggal. Masalah ini melibatkan aplikasi masalah baris tunggal kepada masalah-masalah umum padanan pasangan seperti pengagihan saluran dalam rangkaian telefon selular. Masalah ini diperluas kepada pemetaan dan transformasi graf-graf berhubung kepada baris tunggal. Seterusnya, perbincangan bersambung kepada model FRECAST yang menggunakan model perkomputeran bergerak tanpa talian ad hoc bagi masalah-masalah komunikasi satu-satu, satu-banyak dan satu-semua. Akhir sekali, kajian bertumpu kepada pembentukan model rangkaian sensor tanpa talian melalui model SPLAI bagi perkomputeran unsure-terhingga. Model akhir ini melibatkan penentuan kedudukan sensor dalam rangkaian melalui system kordinat dinamik menggunakan korona dan sector. Output daripada projek penyelidikan ini mengandungi sebuah buku, enam kertas kerja bagi jurnal dan dua kertas kerja prosiding (semuanya peringkat antarabangsa), dan empat perisian.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

SYMBOLS AND NOTATIONS

G	a graph
C_m	a complete graph with m vertices
S_m	single-row representation of C_m
Q	congestion of the nets in S_m
D	number of interstreet crossings (doglegs) in S_m
E	total energy in S_m
L	partial order of nets arranged from top to bottom in S_m
v_j	vertex j in the graph
d_j	degree of vertex j in the graph
m	number of vertices in the graph
t_i	terminal i
b_k	left terminal of net k
e_k	right terminal of net k
n_k	net k, given as $n_k = (b_k, e_k)$
$n_{y,i,m}$	the ith net in level y in S_m
$b_{y,i,m}$	beginning (left) terminal of the ith net in level y in S_m
$e_{y,i,m}$	end (right) terminal of the ith net in level y in S_m
$w_{y,m}$	width of every net in level y in S_m
$r_{y,m}$	number of nets in level y in S_m
z_j	zone j in S_m
G'	Set of all computed cliques in G
C_i	Clique i in G'
G_L	List of nodes in G that are not in the computed cliques
G_M	G with an additional node v_0
M	Number of nodes in G
H	Energy in the Hopfield network
R	Transmission range for connectivity in G
d_i	Degree of node i in G
q_{ij}	Element in the adjacency matrix of G
u_{xi}	Input of neuron i in group x
v_{xi}	Output of neuron i in group x
w_{ij}	Weight of link between v_i and v_j

Chapter 1

RESEARCH FRAMEWORK

1.1 Introduction

A mesh network is a specific representation of a connected graph that has many applications in science, engineering, IT and other infrastructural structures. In its simplest form, a rectangular mesh is a regular structure whose interior nodes have a degree of four, while those in the boundaries have a degree of two or three. A torus is a special case of the rectangular mesh whose nodes have a degree of four, irrespective whether they are interior or exterior nodes.

In this research, we study four components of mesh communication. First, we study the complete graph, and its transformation to the single-row model. A complete graph is a fully-connected graph where every node is adjacent to all other nodes in the graph. Very often, many applications in science and engineering are reducible to this type of graph. Hence, a simplified form of a complete graph contributes in providing the solutions to these problems. In this paper, we present a technique for transforming a complete graph into a single-row routing problem. Single-row routing is a classical technique in the VLSI design that is known to be NP-complete. We solved this problem earlier using a method called ESSR, and, the same technique is applied to the present work to transform a complete graph into its single-row routing representation. A parallel computing model is proposed which contributes in making the problem modular and

scalable. We also discuss the application of this work on the channel assignment problem in the wireless cellular telephone networks.

The second work is the generalization of the first. In a dimensional problem, the transformation of a graph into its linear network can be viewed as a transition involving demand and supply. A connected graph represents the demand flows between the components in the graph while the network supporting it is the resource or capacity links for supporting the demand volumes. The transformation involves a mapping from the graph to its network to satisfy certain performance metrics. We propose a model for transforming a connected graph to its linear network model in the form of a single-row routing network. The main objective is to provide an optimum routing network that minimizes the congestion. In this technique, the given graph is first partitioned into several disjoint cliques using the Hopfield neural network using our model called AdCliq. From the cliques, a set of intervals derived from the zones are obtained through the matching nodes in the single-row axis. The intervals are then mapped into a network of non-crossing nets using our previously developed tool called ESSR. The network is optimal in terms of minimum street congestion and number of doglegs, and this provides a reasonably good step towards the overall solution to the demand-supply problem.

The third work is an application of mesh communication to the problems of single-casting, multi-casting and broadcasting in ad hoc wireless networks. *Ad hoc network* is a self-adapting wireless network formed from special function nodes. Each node in the network is “intelligent” in the sense that it has a transceiver for receiving and transmitting messages. The nodes act as routers and are capable of forming a network in the absence of communication infrastructures such as base stations. We discuss the problem of message transmission in highly-dense ad hoc networks. Both, the single-casting and multicasting problems from one or more nodes in the graph require finding the shortest paths in order to make the message transmission optimum. The multicasting and broadcasting methods in our model is based on the Prim’s algorithm for finding the minimum spanning tree (MST) in a graph. A *spanning tree* is subgraph of a connected graph which forms a tree that includes all the nodes in the graph. The

minimum spanning tree of the graph is a spanning tree of the graph whose total weights is a minimum. We present a visualization model called FRECAST developed using C++ on the Windows environment for computing the shortest path and the minimum spanning tree, and apply them in single-casting, multicasting and broadcasting. FRECAST has been developed based on the routing protocol on the ad hoc networks called WRP. The model supports up to 30 nodes but it can easily be scaled upward to support a denser graph.

The last work is a computational application to the wireless sensor networks. Wireless sensor network refers to a group of sensors, linked by a wireless medium to perform distributed sensing task. The primary interest is their capability in monitoring the physical environment through the deployment of numerous tiny, intelligent, wireless networked sensor nodes. Our interest consists of a sensor network, which includes a few specialized nodes called processing elements that can perform some limited computational capabilities. We propose a model called SPLAI that allows the network to compute a finite element problem where the processing elements are modeled as the nodes in the linear triangular approximation problem. Our model also considers the case of some failures of the sensors. A simulation model to visualize this network has been developed using C++ on the Windows environment.

1.2 Problem Statement

There are four problems for study in this research, as follows:

1. How are the nodes from a complete graph transformed into the nodes in a single-row axis for pair-matchings in such a way that the network consists of non-crossing horizontal and vertical lines?
2. How does the problem in (1) above be generalized into a general connected graph?
3. How is the application to the communication in ad hoc networks?
4. How does the problem relate to the computational model in a wireless sensor network?

1.3 Objectives of the Study

Several objectives are outlined, as follows:

1. To study the properties of connected and complete graphs, and how they relate to the single-row network through linear transformation.
2. To study the applications of the transformation from the complete and connected graphs in wireless cellular networks, wireless ad hoc networks and wireless sensor networks.
3. To study computational model in the wireless sensor networks.

1.4 Report Organization

The report is organized into five chapters. The first chapter is the research framework. Chapter 2 discusses the transformation of a connected graph into a single-row network which has several notable potential applications, such as in channel assignments of the cellular telephone networks. Chapter 3 is the generalization of the first work on connected graphs. Chapter 4 discusses the single-casting, multicasting and broadcasting problems in wireless ad hoc networks. Chapter 5 is another application in the form of a computational model using finite-element method on the wireless sensor networks.

Chapter 2

SINGLE-ROW TRANSFORMATION OF COMPLETE GRAPHS

2.1 Introduction

In many cases, problems in engineering and other technical problems can be represented as problems in graph theory. A problem of this nature is said to be reducible to the form of vertices and links of a graph, and the solution to the problem can be obtained by solving the graph problem. Furthermore, several solutions to the problems in graph theory have found their roots in some well-known prototype problems, such as the traveling salesman problem, the shortest path problem and the minimum spanning tree problem. Solutions to these problems are provided in the form of dynamic programming techniques, mathematical programming and heuristics. Most of these prototype problems have been proven to be NP-complete and, therefore, no absolute solutions to the problems are established. However, their reduction to the form of graphs have, in some ways, simplified their complexity and pave way for further improvement to their solutions.

In this chapter, we study the relationship between a complete graph and its single-row representation. A complete graph is a graph where every vertex in the graph is adjacent to all other vertices. Single-row routing is a classical problem about finding an optimum routing from a set of *terminals*, or nodes, arranged in a single-row in the printed circuit boards (PCB) design. In the *Very Large Scale Integration* (VLSI)

technology, the terminals are the pins and vias, and the routes consist of non-intersecting horizontal and vertical tracks called *nets*. The main goal in single-row routing is to find a realization that reduces the congestion in the network.

In this chapter, we propose a model for transforming a complete graph as nets in a single-row axis. The motivation for this proposal comes from the fact that some problems in engineering are reducible to the form of a complete graph. A complete graph shows the working relationship between all pairs of nodes in the graph. The relationship, in this case, may represent parameters such as the precedence in a directed flow, the communication cost for transferring data and the matching between the nodes.

This idea may also apply to the subgraphs of the graph in the form of one or more cliques. A clique is a complete subgraph of a graph. Our suggestion in this case is to solve the original problem using the divide-and-conquer approach. The problem may be broken into several components where each component is represented as a clique in the graph. This suggests the formation of a parallel computing model with the cliques forming separate and independent modules. A clique may form its own computing base in a distributed computing system. A group of cliques may form a cluster in a parallel computing system.

In this work, we study the mapping properties of a complete graph into its single-axis representation, in the form of the single-row routing problem. We devise a strategy for mapping this graph, and then apply the method for solving a graph-reducible problem, namely, the channel assignment problem in the wireless cellular telephone networks. Channel assignment problem is a NP-hard problem which has its root in the graph coloring problem. The application of the complete graph transformation in the channel assignment problem suggests our method is applicable to the real world applications.

This chapter is organized into eight sections. Section 2.1 is the introduction. Section 2.2 describes the problem in the chapter, while Section 2.3 presents the elementary symbols and terminologies used in the chapter. In Section 2.4, we describe the single-row routing problem, while its solution using the simulated annealing method

is discussed in Section 2.5. We also discuss our earlier model called the *Enhanced Simulated annealing for Single-row Routing* (ESSR) technique in this Section. In Section 2.6, we outline the details of the mapping strategy for converting the complete graph into its single-row axis representation. For the applications, we propose two parallel computing model for this problem in Section 2.7 involving a single-row multiprocessor network and a cellular network model for the channel assignment problem. We conclude the chapter with the summary and conclusion in Section 2.8.

2.2 Problem Formulation

The problem can be stated as follows: given a complete graph C_m, how can the edges in this graph be drawn so that they don't cross each other? The problem here translates into finding a planar graph. Our approach for solving this problem is to transform the complete graph into a single-row routing problem [10], as the single-row representation is a form of planar graph. It is easy to verify that a complete graph, C_m, with m vertices has $m(m-1)$ links (or edges). This is because each vertex in the graph has a degree of $(m-1)$.

Figure 2.1 shows a complete graph C_4 (left) and its single-row representation (right). In this transformation, each node in the graph forms a zone having the number of terminals equals its degree, which is three in this example. The communication links between the nodes are still preserved in this transformation as each zone has exactly one link with all other zones in the network. Therefore, communication between the nodes in the original graph is maintained in this new representation.

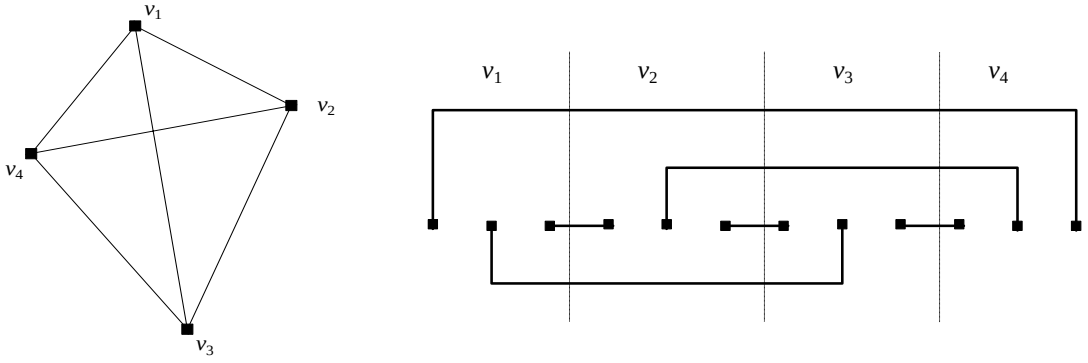


Figure 2.1: A complete graph C_4 (left) and its single-row representation (right).

The problem begins with the mapping of the links in this graph as terminals in a single-row axis. Single-row routing problem is an important component in finding an optimum routing in VLSI design [1,2]. The single-row representation, S_m, of the graph, C_m, consists of m zones and $m(m-1)$ terminals, all aligned in a single-row axis. The terminals are to be formed in equally-spaced intervals along the single-row axis. In VLSI, each terminal represents a pin or via. In the single-row routing problem, nets joining pairs of terminals are to be formed to allow communication between the terminals. A net is made up of non-intersecting horizontal and vertical lines that is drawn in the order from left to right.

In order to produce a practical area-compact design, the nets have to be drawn according to the routes that will minimize the wiring requirements of the network. The main objective in the single-row routing problem is to determine the most optimum routing between pairs of the terminals so as to reduce the congestion in the whole network. It is also important that the routing is made in such a way that the interstreet crossings (doglegs) between the upper and lower sections of the single-row axis be minimized. This is necessary as the terminals in the single-row axis are very close to each other, and a high number of interstreet crossings will generate an intolerable level of heat that may cause problems to the network. In optimization, the problem of minimizing the congestion in the network reduces to a search for the right orderings of the nets, based on a suitable energy function.

2.3 Symbols and Notations

Symbols are used based on two categories, namely, a graph and its single-row representation. A graph G consists of vertices, v_j, for $j = 1,2,...,m$, and a set of edges, or links, joining these vertices. To avoid confusion, the nodes in the single-row axis are referred to as *terminals* in this chapter.

The following notations are being used:

G	a graph
C_m	a complete graph with m vertices
S_m	single-row representation of C_m
Q	congestion of the nets in S_m
D	number of interstreet crossings (doglegs) in S_m
E	total energy in S_m
L	partial order of nets arranged from top to bottom in S_m
v_j	vertex j in the graph
d_j	degree of vertex j in the graph
m	number of vertices in the graph
t_i	terminal i
b_k	left terminal of net k
e_k	right terminal of net k
n_k	net k, given as $n_k = (b_k, e_k)$
$n_{y,i,m}$	the ith net in level y in S_m
$b_{y,i,m}$	beginning (left) terminal of the ith net in level y in S_m
$e_{y,i,m}$	end (right) terminal of the ith net in level y in S_m
$w_{y,m}$	width of every net in level y in S_m
$r_{y,m}$	number of nets in level y in S_m
z_j	zone j in S_m

2.4 Single-Row Routing Problem

Single-row routing is a combinatorial optimization problem that has been proven to be NP-complete [1,2]. Traditionally, single-row routing is one of the techniques employed

for designing the routes between the electronic components of a printed-circuit board. Each path joining the terminals is called a *net*. In the single-row routing problem, we are given a set of $2m$ evenly-spaced terminals (pins or vias), t_i, for $i = 1,2,...,2m$, arranged horizontally from left to right in a single horizontal row called the *single-row axis*. The problem is to construct m nets from the list $L = \{n_k\}$, for $k = 1,2,...,m$, formed from horizontal intervals, (b_k, e_k), in the node axis, where b_k and e_k are the beginning and end terminals of the intervals, respectively. Each horizontal interval is formed from a pair of two (or more) terminals through non-intersecting vertical and horizontal lines. The nets are to be drawn from left to right, while the reverse direction is not allowed.

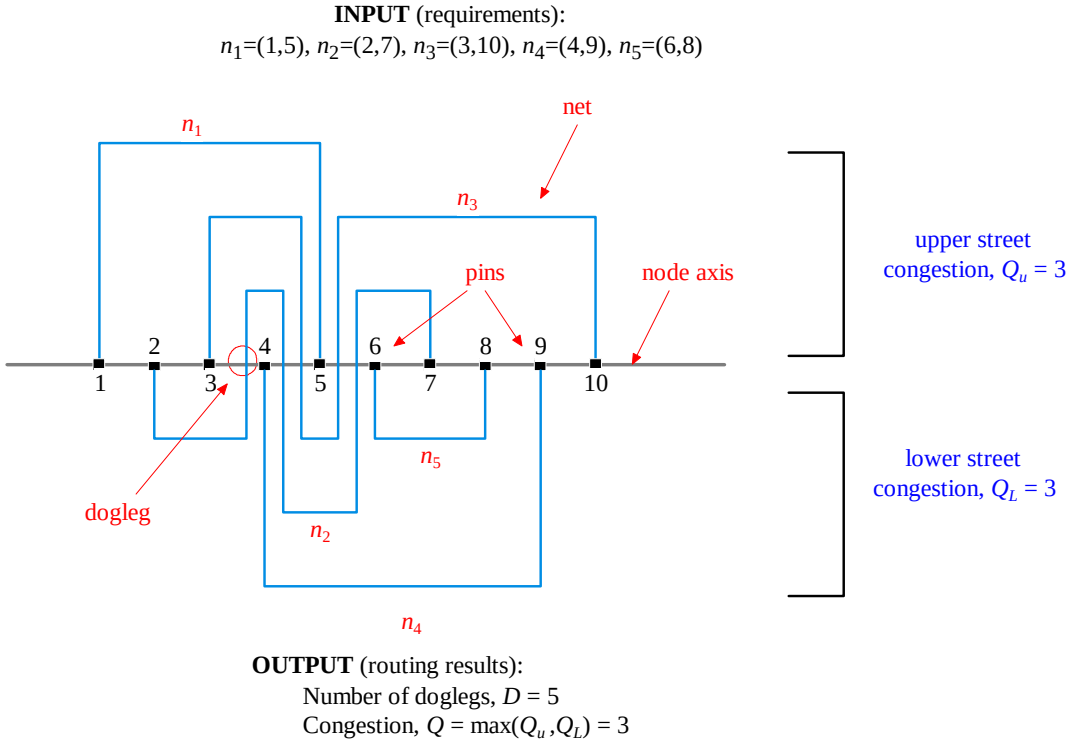


Figure 2.2: Terminologies in the single-row routing problem

Figure 2.2 shows a realization in a single-row routing. Physically, each net in the single row represents a conductor path for its terminals to communicate. The area above the single-row axis is called the *upper street*, while that below is the *lower street*. The number of horizontal tracks in the upper and lower streets are called the *upper street congestion*, C_u, and *lower street congestion*, C_l, respectively. The overall street

congestion Q of a realization is defined as the maximum of its upper and lower street congestions, that is, $Q = \max(C_u, C_l)$.

Various methods based on graph theory, mathematical programming and heuristics have been proposed to solve the single-row routing problem. In [3], Kuh *et al.* proposed some necessary and sufficient conditions for determining the most minimum congestion. In [4], a graph decomposition technique was proposed to obtain an optimum routing based on the overlapping intervals of a graph. In [5], Du and Liu proposed a heuristic for finding an optimum routing based on a method that sorts the nets according to their classes, internal cut numbers and residual cut numbers.

2.5 Simulated Annealing Technique

In this section, we describe our previous method in [6] for solving the single-row routing problem using simulated annealing. Simulated annealing [7] is a heuristic method for solving combinatorial optimization problems using the atomic properties of particles undergoing thermal activities. As demonstrated by Rutenbar [8], it is possible to apply simulated annealing in VLSI design. In his work, simulated annealing has been applied to design an optimum layout for the chip layout and floor-planning.

Simulated annealing is a stochastic and hill-climbing heuristical method based on the gradient-descent optimization technique. Simulated annealing often produces good solutions that are comparable to other techniques. The simulated annealing method based on the Metropolis Algorithm [7] implements the Boltzmann distribution function as its energy minimization network. This probability function is known to have the mechanism to escape from getting trapped in a local minimum. Therefore, convergence is guaranteed although at the expense of a long computation time. In addition, simulated annealing is easier to implement as the objective function does not have to be in an explicit functional representation.

Simulated annealing involves a massive iterative improvement process of local search for the global minimum of a given energy function. This method is based on the

simple and natural technique of trial and error. Initially, the process in simulated annealing requires the definition of a solution space, a cost function and a set of moves that can be used to modify the solution. Through the iterative improvement method, one starts with an initial solution, x_0, and compares this solution with its neighbors. A solution, x', is said to be a *neighbor* of a solution , x, if x' can be obtained from x via one of the moves. The neighbors of x_0 are examined until a neighborhood solution, x, with a lower cost is discovered. In this case, x becomes the new solution and the process is continued to examine the neighbors of the new solution. The algorithm terminates when it arrives at a solution which has no neighboring solution with a lower cost.

In our problem, a pertubation is performed to examine the neighbors by moving a net at random to a new position. The resulting change in energy ΔE is then evaluated. If the energy is reduced, that is $\Delta E<0$, the new configuration is accepted as the starting point for the next move. However, if the energy is increased, $\Delta E>0$, the move generates the probability of acceptance, given by $\Pr[\text{acceptance}] = e^{-\Delta E/T}$. The move is accepted if this probability is greater than a threshold probability of acceptance, ε, and rejected otherwise. The value of ε is proportional to the rate of acceptance or rejection. With a higher value, the number of moves accepted for $\Delta E>0$ are reduced and the same rule applies vice versa.

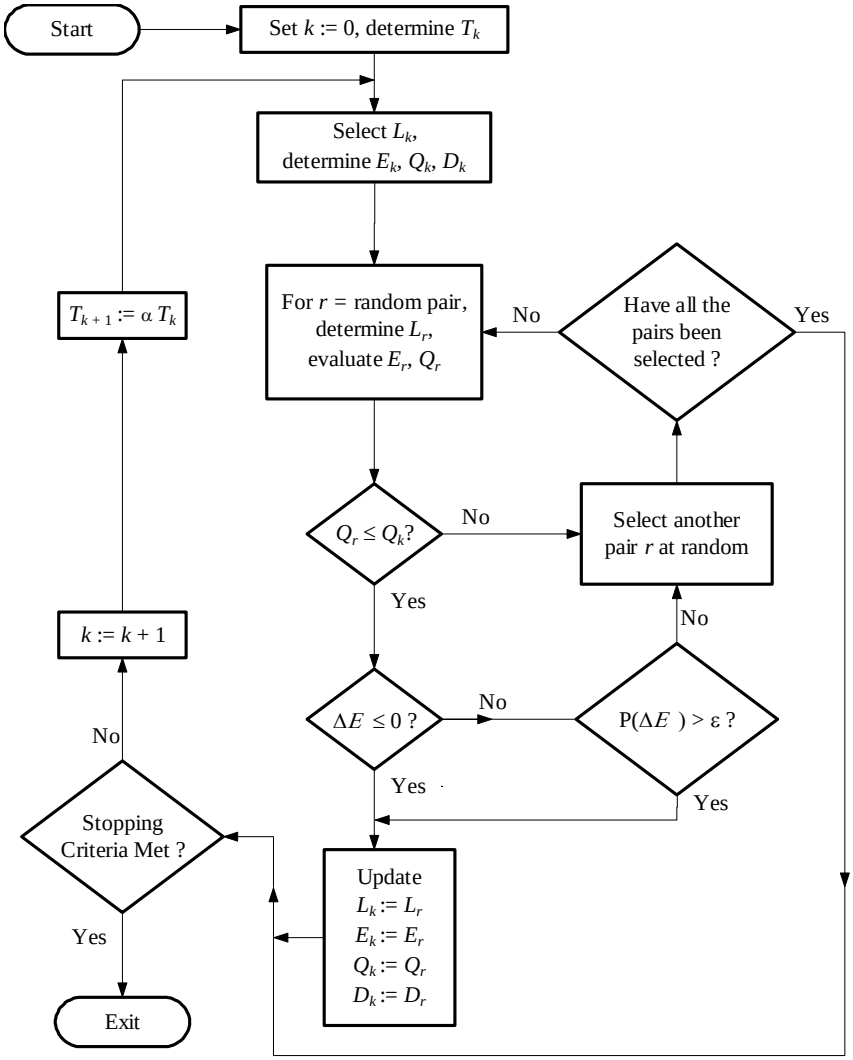


Figure 2.3: ESSR method for the single-row routing problem

Our objective in this problem is to obtain a realization that minimizes both the street congestion, *Q*, and the number of doglegs, *D*. However, this objective is very difficult to achieve as the two components are separate but dependent entities. While having one component minimized, the other tends to show some resistance to its minimization. In [6], we proposed the Enhanced Simulated annealing for Single-row Routing (ESSR) method based on simulated annealing that produces a routing that minimizes both the congestion and the number of interstreet crossings. Figure 3.3 illustrates how ESSR is implemented to solve the single-row routing problem.

To express the above requirement, the energy in a given net ordering is expressed as the total length of all the tracks, according to the energy function, E, derived from our earlier work, as follows:

$$E = \sum_{k=1}^{M} \sum_{j=1}^{M_k} \left| h_{k,j} \right| . \tag{2.1}$$

In the above equation, $h_{k,j}$ is the energy of segment j in net k, while M is the number of nets in the problem and M_k is the number of segments in net k.

2.6 Complete Graph Partitioning Strategy

A graph G consists of a set of vertices and edges, connecting some of these vertices. A graph where a path exists between any pair of vertices in the graph is called a *connected graph*, otherwise it is a *disconnected graph*. Node j in the graph having d links with its neighbors is said to have a degree of d_j. A graph with m nodes where every node is a neighbor of every other nodes in the graph is a complete graph, C_m. In C_m, every vertex has the same degree of $m-1$.

2.6.1 Formation of Zones and Terminals from a Complete Graph

In C_m, every link between a pair of vertices in the graph is mapped as a terminal in S_m. Therefore, a C_m graph having m vertices and $m(m-1)$ links is mapped into m zones with a total of $m(m-1)$ terminals in S_m. A vertex with degree j in the graph occupies a zone in S_m with d_j terminals.

We outline the overall strategy for mapping a complete graph. In general, the transformation of a complete graph, C_m, into its single row representation, S_m, consists of two main steps. First, the vertices, v_j, are mapped into the zones, z_j, that are numbered according to their vertex number, j, for $j = 1,2,...,m$. The next task is to

determine the number of terminals in each zone, z_j, in S_m, which is simply the degree, d_j, of its corresponding vertex, v_j, in C_m. Finally, we obtain the complete layout of S_m by combining all the terminals from each zone and number them successively beginning from the first zone to the last.

Our method for creating the zones and their terminals in S_m from a complete graph, C_m, is outlined in `Algorithm 1`, as follows:

/* **Algorithm 1**: Formation of zones and terminals in S_m from C_m */

Given a complete graph C_m;

Draw the zones, z_j, in S_m, which corresponds to v_j in C_m, for $j = 1,2,...,m$;

for $j = 1$ to m

Determine the degree, d_j, of every vertex, v_j, in C_m;

Set $i = 1$;

for $k = 1$ to d_j

Set the terminal number, $t_i = i$;

Update $i \leftarrow i + 1$;

2.6.2 Construction of Nets from a Complete Graph

In the previous section, we described a plan to form the zones and nets in S_m from C_m using `Algorithm 1`. We illustrate the idea on the problem of forming a single-row representation of C_5, a complete graph with $m = 5$ vertices. In this problem, each vertex in the graph has a degree of 4. There are $m = 5$ zones, z_i, for $i = 1,2,...,5$ and the number of terminals on the single-row axis is $m(m-1) = 20$. Hence, the number of nets

formed is $r_m = \frac{m(m-1)}{2} = 10$. Figure 2.4 shows the zones and terminals in S_5 formed from C_5 when `Algorithm 1` is applied.

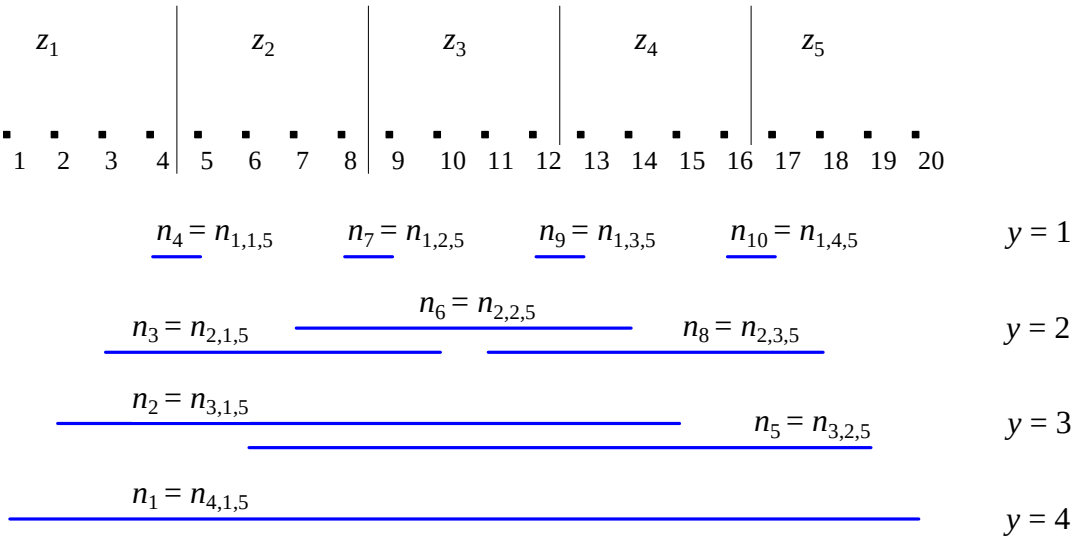


Figure 2.4: Formation of nets based on the zones and levels in S_5 from C_5

We now present a technique for forming the nets in the network that contributes in minimizing the total energy in S_m. The technique calls for the formation of the nets by grouping them first into several levels based on their width. The *width* of net k, denoted as w_k, is defined as the difference between its beginning and end terminals, given as $w_k = e_k - b_k$. A *level*, y, in S_m consists of a set of equal-width nets grouped in ascending order from the lowest width to the highest. Our strategy begins with Preposition 1 which consists of first forming levels where the nets with equal width are grouped. In this proposal, the nets in S_m are created by defining their end-points. Once the nets have been formed, the next step consists of sorting and renumbering the nets based on their beginning terminals, in ascending order from the lowest to highest. These two steps are summarized in `Algorithm 2`.

Preposition: The ith net in level y in S_m, denoted as $n_{y,i,m} = (b_{y,i,m}, e_{y,i,m})$, formed from the complete graph, C_m, is grouped into levels based on its width, $w_{y,m}$, according to the following relationships:

$$b_{y,i,m} = (m - y) + (m - 1)(i - 1), \tag{2.2a}$$

$$e_{y,i,m} = b_{y,i} + w_{y,m} \tag{2.2b}$$

for $y = 1,2,..., m-1$, and $i = 1,2,..., m-1$.

From Preposition, we obtain the width of the nets in level y of S_m, given as follows:

$$w_{y,m} = 1 + (m + 1)(y - 1), \tag{2.3}$$

and the number of nets in each level as follows:

$$r_{y,m} = (m - y). \tag{2.4}$$

The strategy for grouping the nets into levels based on their width is to minimize the total network energy, given in Equation (2.1). This goal can be achieved by forming nets starting from the shortest width, continue with the next shortest and so on. Starting with level 1, that is, $y = 1$, the nets are formed from two consecutive terminals from two different zones. This level has the most minimum width possible, given by $w_{1,m} = 1$. This minimum width has the advantage of producing the net energy equals 0, as the net can be drawn directly on the node axis. The ith net is formed from the last terminal in z_i and the first terminal in zone $(i+1)$th, to make sure that the width remains the same. Using Equations (2.2a) and (2.2b) from Preposition 1, we then obtain the ith net in this level, $n_{1,i,m} = (b_{1,i,m}, e_{1,i,m})$, given as $b_{1,i,m} = (m-1) + (m-1)(i-1)$ and $e_{1,i,m} = b_{1,i} + 1$.

In level 2, the first net is obtained by having the second last terminal in z_1 as its beginning terminal, and the second terminal of z_2 as the ending terminal. This gives the width as $w_{2,m} = 1 + (m+1) = m+2$. In general, the ith terminal in this level, $n_{2,i,m} = (b_{2,i,m}, e_{2,i,m})$, is given by $b_{2,i,m} = (m-2) + (m-1)(i-1)$ and $e_{2,i,m} = b_{2,i} + w_{2,m}$.

`Algorithm 2` summarizes our method for constructing the nets based on the levels of the nets with equal width. In this algorithm, the nets are formed based on Equations (2.2a) and (2.2b). The number of nets and their width in each level are determined from Equations (2.4) and (2.3), respectively. Once the nets have been formed, the algorithm then sorts and renumbers the nets based on their beginning terminals, in ascending order from the lowest to highest. `Algorithm 2` prepares the nets before the next important step, which is their execution in ESSR to determine their optimum routing.

/* **Algorithm 2:** Construction of nets in S_m */
Given a complete graph C_m with m vertices;
Let the number of nets in level 1, $r_1 = r$;
The initial width of nets in level 1 is 1, that is, $w_{1,m} = 1$;
for $y = 1$ **to** r
  if $y > 1$
    Update the width of the nets in level y, $w_{y,m} \leftarrow 1 + (m+1)(y-1)$;
    Update the number of nets in level y, $r_{y,m} \leftarrow (m-y)$;
  for $i = 1$ to r_y
    Form the ith net in level y as follows:
      Update the left terminal of net y, $b_{y,i,m} \leftarrow (m-y) + (m-1)(i-1)$;
      Update the right terminal of net y, $e_{y,i,m} \leftarrow b_{y,i} + w_{y,m}$;
for $y = 1,2,\ldots, r$
  for $i = 1,2,\ldots, r_y$
    Sort $(b_{s,i,m}, e_{s,i,m})$ in ascending order with $b_{s,i,m}$ as the primary key;
for $k = 1$ to $\dfrac{m(m-1)}{2}$
  Assign $n_k = (b_k, e_k)$ from the sorted $(b_{s,i,m}, e_{s,i,m})$;

We illustrate the idea of constructing the nets using Preposition through the example in Figure 2.4. In this figure, a complete graph with 5 vertices, C_5, maps as S_5. The zones and terminals are obtained by applying `Algorithm 1`. By applying Equations (2.2a) and (2.2b) from Preposition 1, we obtain the nets grouped into 4 levels, as shown in Figure 2.4. `Algorithm 2` transforms the C_5 into S_5. Table 2.1 shows the nets obtained from this construction. We then apply ESSR to the nets to obtain the results in the form of an ordering with minimum energy, $E = 11$, as shown in Figure 2.5. The final realization of the network with $Q = 3$ and $D = 1$ is shown in Figure 2.6.

Table 2.1: Formation of nets in S_5 from C_5

Level, y	Width, $w_{y,5}$	#nets, r_y	Nets
1	1	4	(4,5), (8,9), (12,13), (16,17)
2	7	3	(3,10), (7,14), (11,18)
3	13	2	(2,15), (6,19)
4	19	1	(1,20)

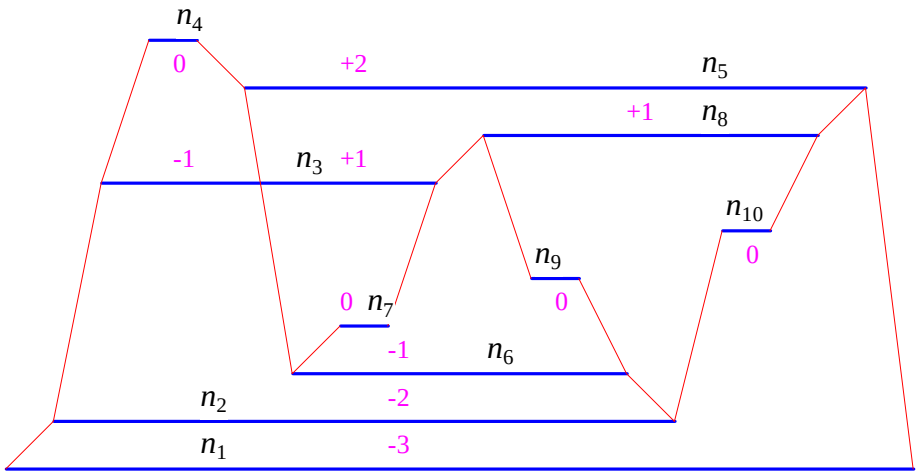


Figure 2.5: Nets ordering with minimum energy, $E = 11$, of C_5 using Algorithm ESSR

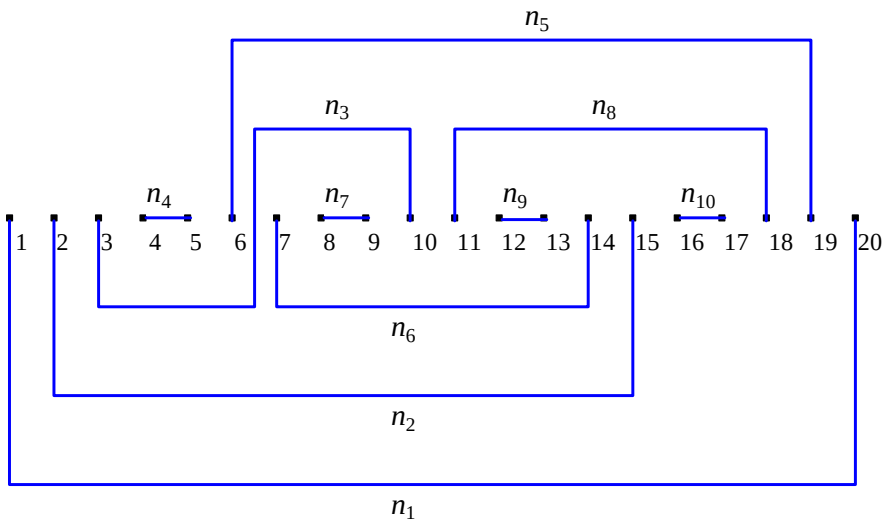


Figure 2.6: Final realization of C_5 with $E = 11$, $Q = 3$ and $D = 1$

We also apply the method to several other models of complete graphs. Table 2.2 summarizes the results of these graphs with m vertices, C_m, in their single-row representations. Figure 2.6 shows the final realization of the routing obtained using ESSR from C_{10}.

Table 2.2: Summary of results for some complete graphs, C_m

C_m	#nets	E	Q	D
$m = 5$	10	11	3	1
$m = 6$	15	28	4	40
$m = 8$	28	128	9	21
$m = 10$	45	403	16	53

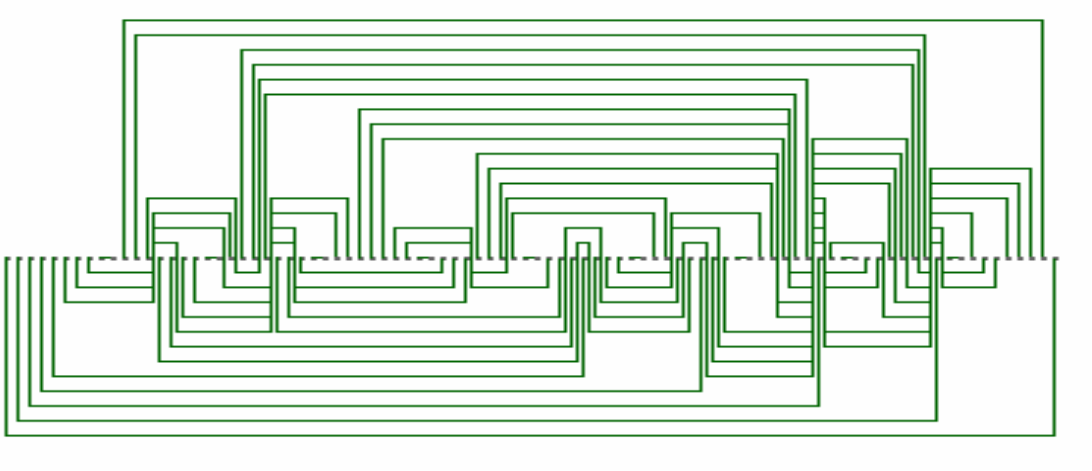


Figure 2.7: Realization of an optimal assignment of 45 nets from C_{10} in Table 2.2.

2.7 Parallel Processing Model

Single-row routing technique is not restricted to the design of the printed-circuit boards only. We explore other potential benefits and suggest two of them in this section. The models suggested are parallel computing networks involving the designs of the single-row multiprocessor network and the channel assignment problem in a wireless cellular telephone network. We describe the two models in a brief note according to their relationship to the single-row transformation problem.

2.7.1 Single-Row Multiprocessor System

The transformation results in the form of non-crossing nets in the single-row model suggest the nets are independent of each other. In a single-row routing without doglegs, communication between the pairs of terminals in the single row can be established without passing through any intermediate node. A model called a single-row multiprocessor system is proposed, as shown in Figure 2.8. In this diagram, the processors are the shaded rectangles arranged in the single-row axis. The circles in the upper and lower streets of the network are the switches which can route the communicating lines into the north, south, east and west directions. Each communicating line in this network model is called a bus.

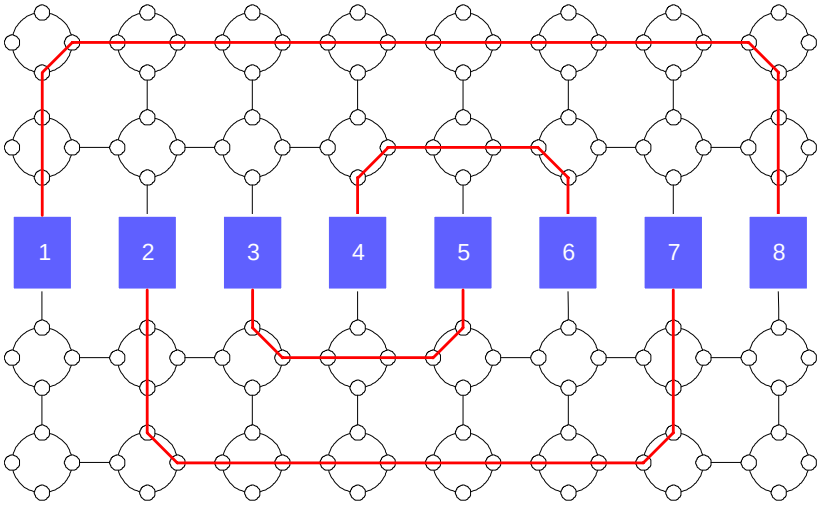


Figure 2.8: Single-row multiprocessor model having no doglegs

Figure 2.9 shows another single-row multiprocessor model which involves doglegs. This model is derived from the transformation results of the complete graph C_5 into the single-row model S_5, which produces five zones. The routing results are obtained from ESSR with an optimum congestion of three and one dogleg. In this model, each zone formed in the single row axis is represented as a processor while the terminals form the ports in the processors. The processors in this diagram are labeled as p_i, for $i = 1,2,...,5$. A dogleg in the network means the net between terminals 3 and 11 crosses the single row axis through p_2. In this case, an extra port is created in this processor to enable the crossing to take place. It follows that doglegs can be represented as extra terminals in the host processors at the crossings.

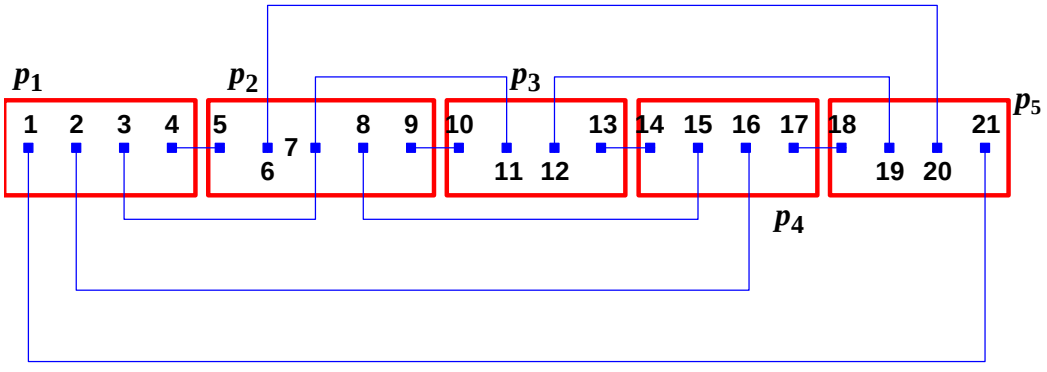


Figure 2.9: Single-row multiprocessor network with a dogleg.

2.7.2 Application to the Channel Assignment Problem

The single-row mapping strategy can also be applied to the problem of assigning radio channels in a wireless cellular telephone network. In the wireless cellular telephone network [9], the assignment of radio frequencies to the mobile users within the network can be modeled as the problem of mapping a complete graph into non-intersecting single-row nets. This network consists of a geographical region partitioned into several cells where each cell is allocated with a *base station* (BS). A base station has a transmitter and a receiver for receiving and transmitting messages from/to mobile users within its cell. The base stations in the network are linked with high-speed cables to the *mobile switching center* (MSC), which functions as a controller to allow communication between any two mobile users in the network by assigning a channel each to each of them.

Figure 2.10 shows two cells in a cellular network where communication is established using the single row routing technique. The model is produced from the transformation of C_6 into S_6. There are 42 channels and there are represented as terminals in the node axis. When a call is made from a cell, a request is received by the base station in the cell. The base station relays this request to the mobile switching center (MSC). Assuming the call is made and received within the network, a channel each needs to be assigned to the caller and the receiver. In this network, MSC plays an important role in assigning a pair of different channels to both the caller and the receiver, to allow immediate circuit switching.

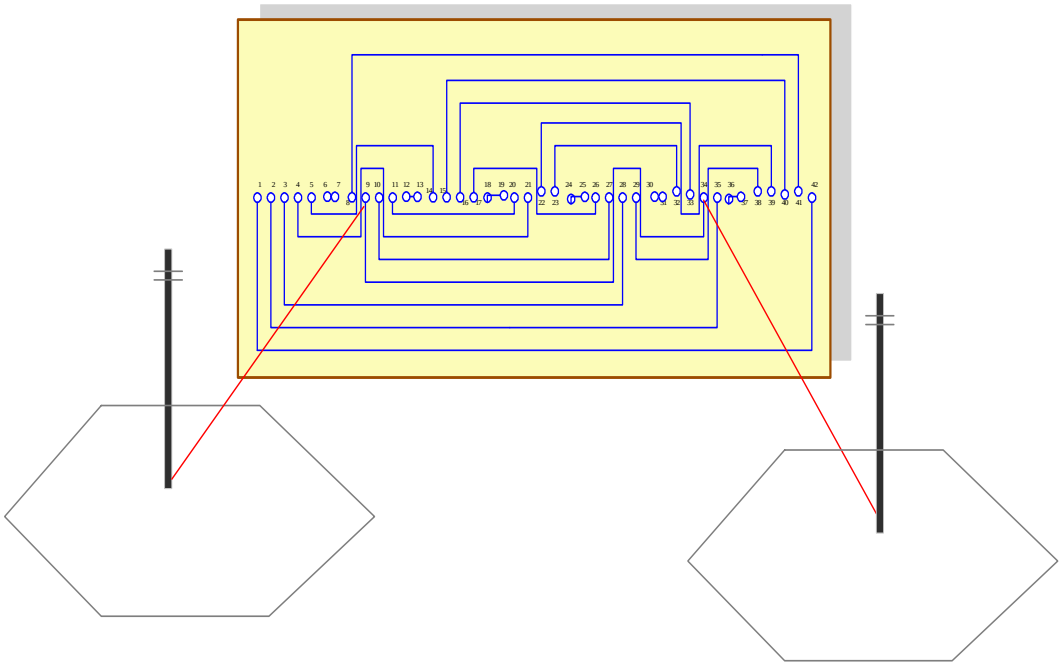


Figure 2.10: Single-row routing model for the cellular network

In the channel assignment problem, we model the channels as the edges of a complete graph. The cells in the network are then represented as nodes in the graph. In the single-row axis, each of these cells is a zone and the channels allocated to a cell are terminals in the zone. Communication between two mobile users from two different cells is established through a net linking their two terminals. We model the single-row communication to be handled by the mobile switching center. This is because MSC has a control on all channel assignments in the network, and this important task must be done immediately without delay when requests for calls are received. In addition, MSC must also be able to provide services associated with problems in channel assignments, such as location finding of mobile users, and channel handovers as a mobile user moves from one cell to another.

We illustrate our model using an example with a network of 5 cells. The problem reduces to the complete graph, C_5, which is represented as the zones, z_j, for $j = 1,2,...,5$ in S_5, as shown in Figure 2.5. Hence, 20 channels are available for assignments and each of these channels is represented as a terminal in the single-row axis. The channels are formed using the same technique discussed in Section 2.6.2, to

produce the results as shown in Table 2.1 (the channels are numbered by assuming there are no electromagnetic interferences on the channels) . Figure 2.6 is then the final realization of the optimum routing of the nets obtained using ESSR.

2.8 Summary and Conclusion

In this chapter, we propose a method for transforming a complete graph, C_m, into its single-row representation, S_m. We first describe the single-row routing problem, which is a classical technique in VLSI design. We relate the problem to our earlier work which solved the problem using a method based on simulated annealing, called ESSR. The transformation from C_m to S_m involves the formation of nets based on Preposition. The proposal groups the nets with equal width which contributes in reducing the overall energy of the network. The whole process is implemented using `Algorithms 1` and `2`. We then apply ESSR to the network to obtain some reasonably good results for optimum routing. Finally, we also describe briefly the application of this transformation technique in solving the channel assignment problem in the wireless cellular telephone networks.

Chapter 3

SINGLE-ROW MAPPING OF CONNECTED GRAPHS

3.1 Introduction

In most cases, a connected graph represents a network of inter-dependent nodes where their links provide the communication paths between the nodes. In a connected graph, a node can communicate with any other node by hopping through one or more intermediate nodes. In a multi-commodity network problem, the nodes in a connected graph represent the demand and they are competing against each other for the available resources, or link capacities, in the form of network equipment (or hardware). The objective is to minimize an objective function subject to some constraints. A linear programming model is obtained if the equations in the objective function and its constraints form a linear system.

In this chapter, two important components are discussed and linked to each other. First is the connected graph which is an abstraction representing a demand in the demand-supply problem. Second is single-row routing which represents the supply in the problem. Traditionally, single-row routing is a classical problem that contributes in the printed-circuit board (PCB) designs [13]. We study the demand-supply relationship between a connected graph with the single-row routing problem, and show how the latter can be applied for solving the problem in the former.

In [13], we discussed a technique for transforming a complete graph into its single-row representation. A complete graph is a fully-connected graph where every node is adjacent to all other nodes in the graph. Very often, many applications in science and engineering are reducible to this type of graph. Hence, a simplified form of a complete graph contributes in providing the solutions to these problems. In this chapter, we present a technique for transforming a complete graph into a single-row routing problem. Single-row routing is a classical technique in the VLSI design that is known to be NP-complete. We solved this problem earlier using a method called ESSR, and, the same technique is applied to the present work to transform a complete graph into its single-row routing representation.

Our work in this chapter is motivated from the need to expand the scope of the single-row routing problem. In [13], we showed that single-row routing can also be applied in two non-PCB applications, namely, the channel assignment problem in the wireless cellular network systems and the theoretical single-row multiprocessor network system. In the first application, each node in the given graph maps into a zone consisting of several channels for serving the demand from the mobile users. In the second application, the nodes in the single-row axis are modeled as processors in a theoretical multiprocessor system. Therefore, it is safe to say that single-row routing technique can also be applied to cover many other problems, particularly those that require matchings between the pairs of nodes in the graph.

It is obvious that two NP-complete problems are encountered here, namely, the maximum clique of a graph and single-row routing. In supporting our work, two tools have been developed, both using C++. First is AdCliq which is a visual simulator for partitioning a connected graph into sets of cliques using the Hopfield neural network method. Second is ESSR [6] which generates a realization in the form of single-row representation of the nodes grouped in cliques in AdCliq. ESSR performs a search on the minimum energy function of the nets based on the stochastic simulated annealing method. As a team AdCliq and ESSR form a strong combination for transforming the nodes in a connected graph into their single-row realization.

In this chapter, we propose a technique for transforming a connected graph into its single-row representation. The method starts by partitioning the graph into several disjoint cliques using the Hopfield neural network. Once the cliques have been established, the step continues by forming an interval graph with matching nodes in the single-row axis. Finally, we apply ESSR to obtain its least congested single-row routing.

The chapter is organized into eight sections. Section 3.2 is the problem formulation while the terminologies and symbols used are outlined in Section 3.3. Section 3.4 discusses some brief introduction to the single-row routing problem, and its solution using ESSR. The maximum clique problem and its solution using the Hopfield neural network is discussed in Section 3.5. This is followed by a description of the transformation technique using both AdCliq and ESSR in Section 3.6. We then display some experimental results from the simulations using AdCliq and ESSR in Section 3.7. Finally, Section 3.8 is the summary and conclusion.

3.2 Problem Formulation

The problem can be stated as follows: given a connected graph G with an arbitrary number of nodes and edges, how can this graph be transformed into a network of non-crossing nets with their proper matching nodes? It is also our main objective in the problem to design a routing in the network with minimum street congestion and number of doglegs.

The above problem may be expressed as a *dimensional problem* in the form of demand-supply problem [14]. In this case, the graph represents the demand while the single-row network is the supply or equipment (hardware) for supporting the demand. Therefore, the problem translates into mapping the demand to the equipment in such away to meet certain performance metrics. In many cases, the demand-supply problem can be expressed as a constrained or unconstrained linear programming problem.

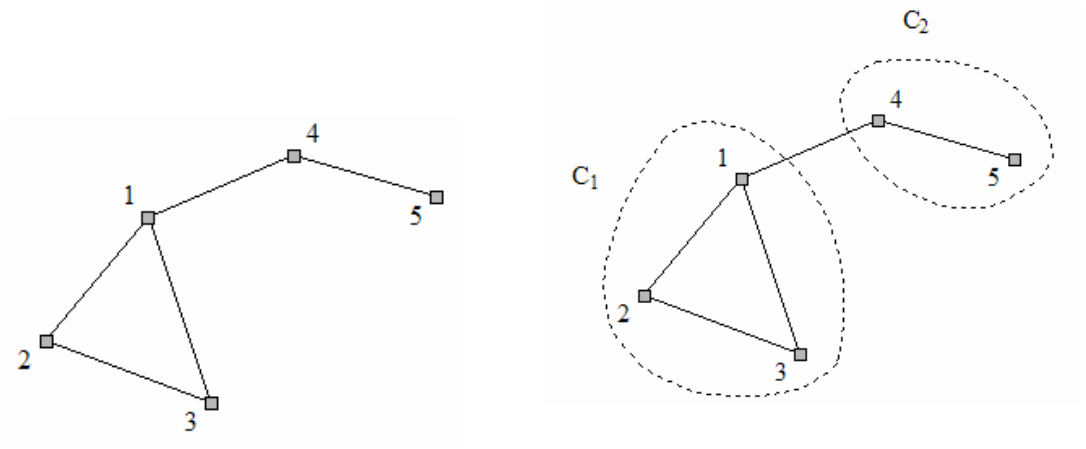


Figure 3.1: A connected graph G_5 with five nodes (left) And its partitions.

Figure 3.1 shows an example of a connected graph G_5 with five nodes. Our strategy in solving the problem requires transforming G_5 into two cliques, C_1 and C_2, before mapping the nodes into a single-row network S with 10 pins. The transformation requires a series of steps including partitioning the graph into several cliques before getting the linear representation.

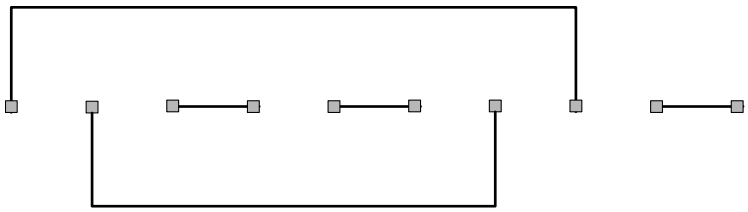


Figure 3.2 Single-row representation S from G.

Figure 3.2 show the result of the transformation into its single-row representation. The figure shows the final realization of the single-row representation with a minimum street congestion value of 1 and having no dogleg. This realization is optimal with its energy at the minimum value of 2.

3.3 Symbols and Terminologies

Several symbols and terminologies related to a connected graph and its transformation to the single-row model are explained as follows:

Graph and Hopfield network models

G	Graph
G'	Set of all computed cliques in G
C_i	Clique i in G'
G_L	List of nodes in G that are not in the computed cliques
G_M	G with an additional node v_0
M	Number of nodes in G
H	Energy in the Hopfield network
R	Transmission range for connectivity in G
d_i	Degree of node i in G
q_{ij}	Element in the adjacency matrix of G
u_{xi}	Input of neuron i in group x
v_i	Node i in G
v_{xi}	Output of neuron i in group x
w_{ij}	Weight of link between v_i and v_j

3.4 Hopfield Network Approach to the Maximum Clique Problem

A clique C of a graph G is a subgraph of G where a node in C is adjacent to every other node in the subgraph. The number of nodes in a clique is called its cardinality. A clique with the maximum cardinality in a graph is also called the maximum clique. The maximum clique problem is one of the most fundamental problems in graph theory which has many applications such as in scheduling, clustering and resource allocation. The problem is known to be NP-complete as it has interacting variables with many degrees of freedom.

The maximum clique problem is one of the most fundamental problems in graph theory. Several methods have been proposed to solve this problem. One such method is the Hopfield neural network [12] where an advantage can be seen from the fact that the graph maps directly to the network. The network itself is a dynamic system which allows the neurons to update their values asynchronously on new inputs, and maintain these values as inputs to other neurons. This process leads the network to enter a new state which depends on some iterative parameters such as time and the thermostatic temperature. The network is said to stabilize when changes on the input values for some neurons bring no effect to the network as a whole.

Hopfield neural network is a fully-connected, recurrent network, which has strong potential for hardware implementation, due to the collective dynamical properties of its interacting neurons [12]. The network itself is a complete graph where a node is connected to every other node in the network. Updates on the nodes are executed asynchronously where only a few nodes are selected at random at each time step. Unlike feed-forward networks, the Hopfield network involves cyclic connections that make them behave as a nonlinear dynamic system. Essentially, the system has very rich temporal and spatial behaviors, such as stable and unstable fixed points, limit cycles, and chaotic behavior which can be used to model associative memory. The network dynamics are dominated by locally stable states called *attractors*, from where their state equations converge after a long temporal evolution.

We implement the method which formulates the maximum clique problem as an unconstrained quadratic zero-one problem, with the energy function $f(\mathbf{x})$ given as follows:

$$f(\mathbf{x}) = \mathbf{b}^{\mathbf{T}}\mathbf{x} + \mathbf{x}^{\mathbf{T}}\mathbf{Q}\mathbf{x} = \sum_{i=1}^{M} b_i x_i + \sum_{i=1}^{M}\sum_{j=1}^{M} q_{ij} x_i x_j, \tag{3.2}$$

for the vector $\mathbf{x} = [x_1, x_2, ..., x_M]^T$ with $x_i \in \{0,1\}$, $\mathbf{b} = [b_1, b_2, ..., b_M]^T$ and $\mathbf{Q} = [q_{ij}]$ is an $M \times M$ symmetrical rational matrix with zero diagonal elements. The corresponding objective here is to minimize $f(\mathbf{x})$.

A new matrix G_M is obtained by adding v_0 into G and this makes the unconstrained quadratic zero-one programming problem equivalent to the problem of minimizing the weight summation over the same partition in G_M. From this concept, the maximum clique problem becomes the global minimization of the following energy function:

$$H = \sum_{x=0}^{M} \sum_{y=0}^{M} \sum_{i=1}^{2} w_{xy} v_{xi} v_{yi} \tag{3.3}$$

where $\sum_{i=1}^{2} v_{xi} = 1$ for $x = 0,1,...,M$ where $v_{xi} \in \{0,1\}$ and $v_{01} = 1$. In this case, the weight w_{ij} of an edge between nodes i, j in G_M is defined by

$$\begin{aligned} w_{0i} &= w_{i0} = \frac{1}{4}\left(\sum_{j=1}^{N} q_{ij} - 1 \right), \\ w_{ii} &= 0; \quad w_{ij} = w_{ji} = \frac{1}{4} q_{ij} \end{aligned} \tag{3.4}$$

for all $i \neq j$ and $i, j = 1,...,M$.

In general, the M-node maximum clique problem can be mapped onto the Hopfield network with M neurons. Comparing the Hopfield energy function (3.2) with the energy function of Equation (3.3) defined for the maximum clique problem, we obtain all the self-connection weights, $w_{xi,xi} = 0$, the interconnection between neurons in the same group x, $w_{xi,xj} = 0$ and the bias for every neuron $\theta_{xi} = 0$. Substituting these values in Hopfield's updating rule (3.3), we obtain the input of neuron i in the xth group given by

$$u_{xi} = -2 \sum_{y=0}^{M} w_{xy} v_{yi}. \tag{3.5}$$

The dynamics of the network are reduced to

$$v_{xi}(t+1) = \begin{cases} 1 & \text{if} \quad u_{xi}(t) = \max_{j=1,\ldots,m} \{u_{xj}(t)\}, \\ 0 & \text{otherwise.} \end{cases} \tag{3.6}$$

We chose the group of neurons, *x* with the smallest energy difference to be updated at time *t*, which is $\Delta H_x(t) = \min_{y=1,\ldots,n} \{\Delta H_y(t)\}$. This selection enables the network to converge to a stable state in just a few iterations, and the global minimum leads to the maximum clique of the graph.

3.5 Single-Row Transformation

Single-row transformation of a graph involves four steps, as shown in Figure 3.3. In the first step, the given connected graph *G* is partitioned into several disjoint cliques $G' = \{C_i\}$. The given graph represents the demand in the demand-supply problem. Our approach for finding the cliques is based on the Hopfield neural network method through AdCliq, as described in Section 3.4. The second step involves the formation of nodes on a single-row axis and their classification into zones. This leads way to the third step which is the formation of intervals for matching the nodes on the single-row axis. The last step produces the optimal realization for the minimum congestion in the network using ESSR. The solution is expressed in the form of a network realization which represents the equipment in the demand-supply problem.

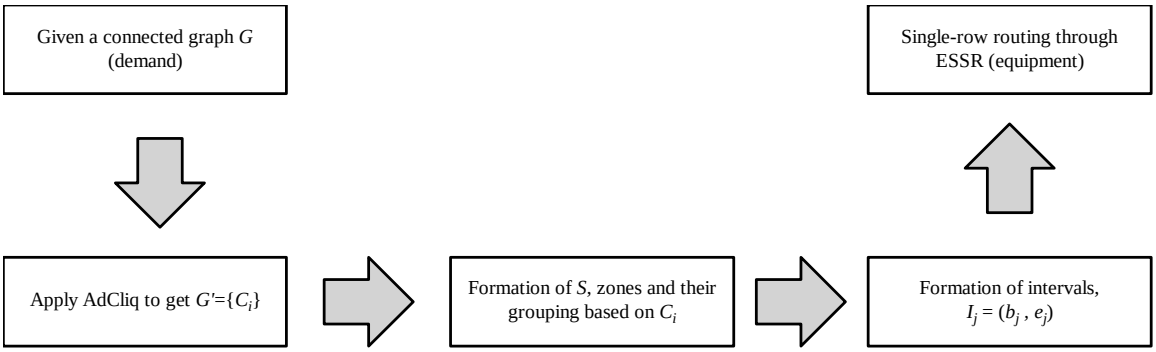


Figure 3.3: The steps in the transformation.

3.5.1 AdCliq: Our Tool for Partitioning a Graph into Cliques

In our work, the maximum clique problem is an important component in transforming a given graph G to its linear form for the demand-supply problem. An important step in the work is to partition G into q disjoint cliques $G = \{C_1, C_2, ..., C_q\}$ in such a way that $|C_1| \geq |C_2| \geq ... \geq |C_i| \geq ... \geq |C_q|$, where $|C_i|$ is the cardinality of C_i. In this notation, C_1 is the maximum clique of G, C_2 is the next maximum clique with the nodes in C_1 excluded from G, and continues until the minimum clique C_q.

Our model is called AdCliq, and it is based on the Hopfield neural network. AdCliq has been developed using the C++ programming language on the Windows environment. The simulation model generates a graph G with two to 100 nodes at the randomly determined positions in the window. A link or edge between a pair of nodes is determined through their Euclidean distance: a link exists if their Euclidean distance is less than a preset threshold value R. This approach provides links to the nodes of the graph that are close to each other, and ignore those that are far apart from each other. Therefore, an area that is congested with nodes is said to have a high density, and this prompts for a higher number of links to meet the higher demand of communication and data sharing.

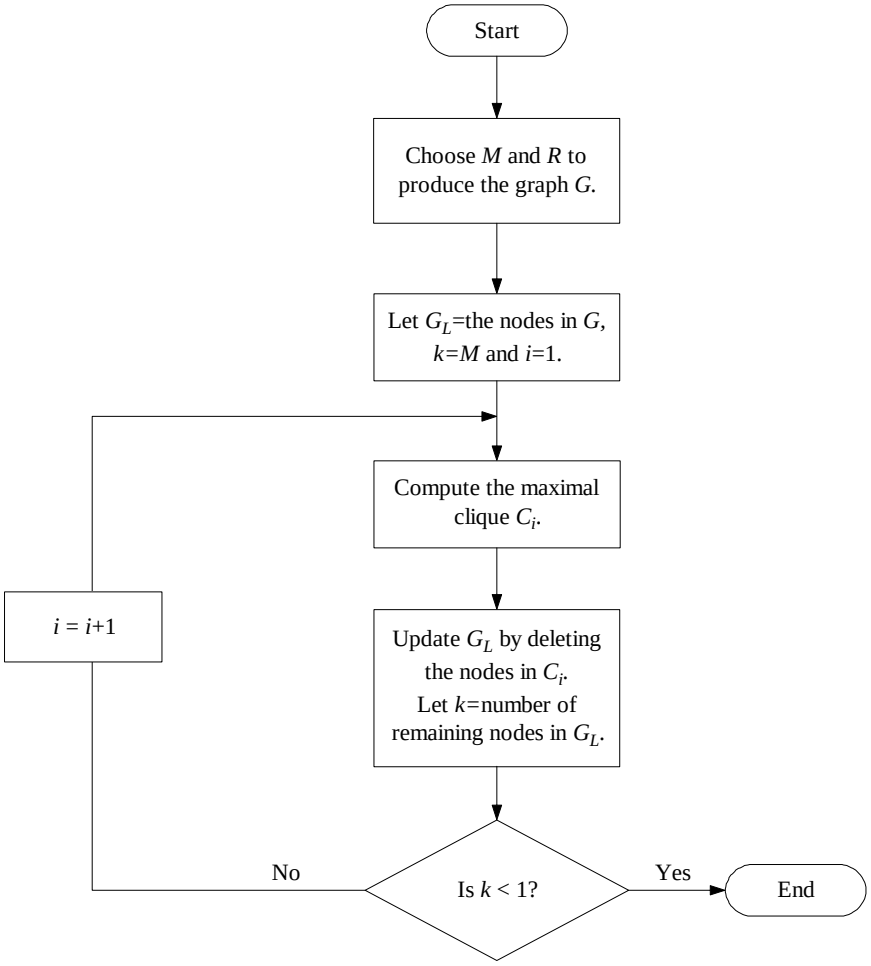


Figure 3.4: Execution steps in AdCliq

Figure 3.4 shows the flowchart of the execution steps in AdCliq. The model starts with input in the form of M and R, for the number of nodes in G and the transmission range, respectively. From the input, a list G_L of all the nodes in G is created, with n as its number. Initially, $k = M$ but this value will decrease as G_L excludes all the nodes from the computed cliques.

The next step in AdCliq is to compute the maximum clique based on the list of nodes in G_L. The clique is denoted C_1, and once it has been determined all the nodes in C_1 are removed from G_L. The number of nodes k in G_L is updated, and a check is made to determine whether the iteration continues or to be stopped. If $k < 1$ the operation is stopped as it signals all the cliques in G have been computed. Otherwise,

the above step is repeated for finding the second maximum clique, or C_2, and so on until the stopping criteria $k < 1$ is reached.

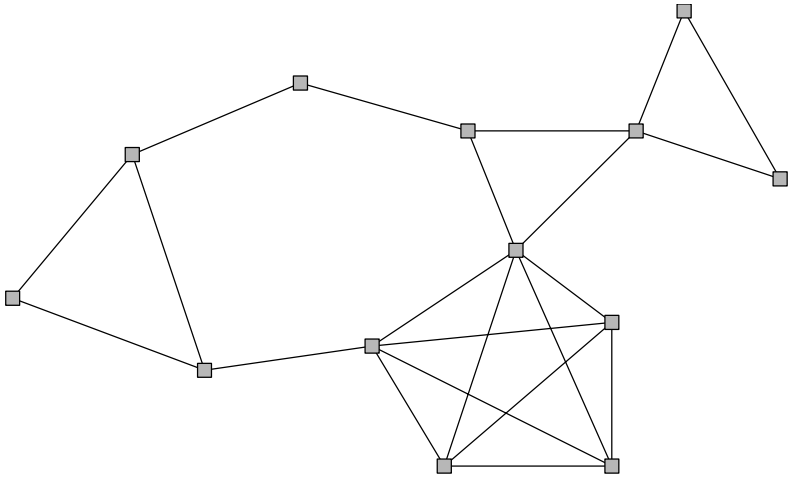


Figure 3.5: A graph with 13 nodes.

Adcliq is illustrated using an example on the graph in Figure 3.5. The connected graph G consists of 13 nodes, labeled as v_i where $i = 1,2,...,13$. Applying AdCliq, we obtain $G' = \{C_1, C_2, C_3, C_4\}$, with $C_1 = \{v_1, v_2, v_3, v_4, v_5\}$, $C_2 = \{v_6, v_7, v_8\}$, $C_3 = \{v_9, v_{10}, v_{11}\}$ and $C_4 = \{v_{12}, v_{13}\}$, as shown in Figure 3.6.

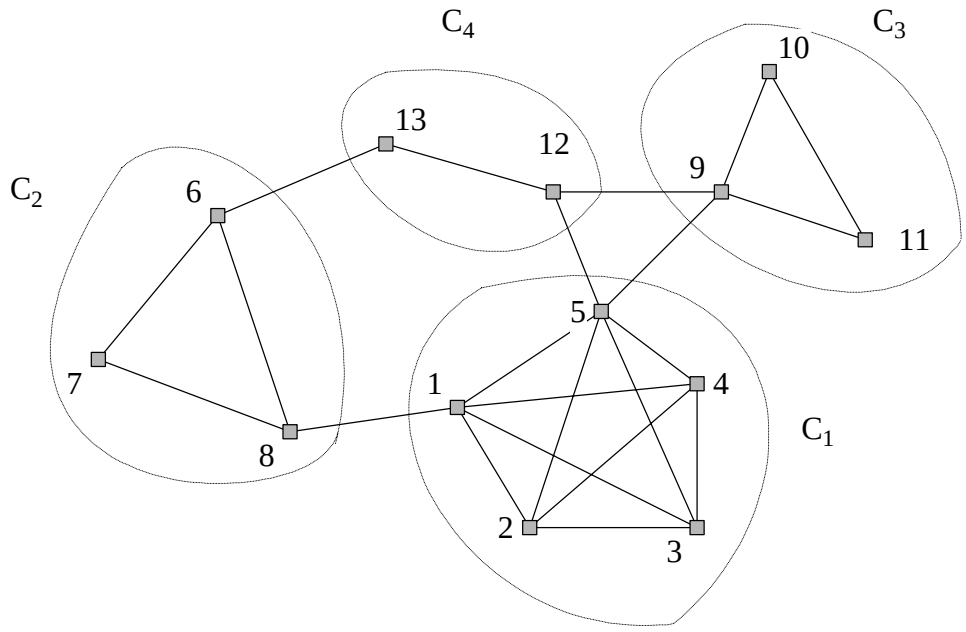


Figure 3.6: Cliques of the graph in Figure 3.5.

3.5.2 Formation of Zones in the Single-Row Node Axis

The second step involves the formation of the single-row node axis S from G. In the transformation, the graph G with M nodes generates N nets in S, where each net represents a pair of nodes in S. The node v_i in G forms zone z_i consisting of d_i nodes in S, where d_i is the degree of v_i. It can easily be shown that there are $2N = \frac{1}{2}\sum_{i=1}^{m} d_i$ nodes in S. The zones are further classified as cliques C_j according to the graph G'.

The next step is to form a set of linear nodes S from G'. Each node v_i in G' expands into d_i nodes in S, where d_i is the degree of v_i. Figure 3.7 shows the linear nodes from G' obtained from Figure 3.6. In the figure, $d_1 = 5$ since the degree of v_1 is 5. There are $\sum_{i=1}^{13} d_i = 44$ nodes in S, and they are placed into 13 zones, z_i, for $i = 1,2,...,13$, according to their origin in G'. The nodes in S are labeled s_k, for $k = 1,2,...,44$.

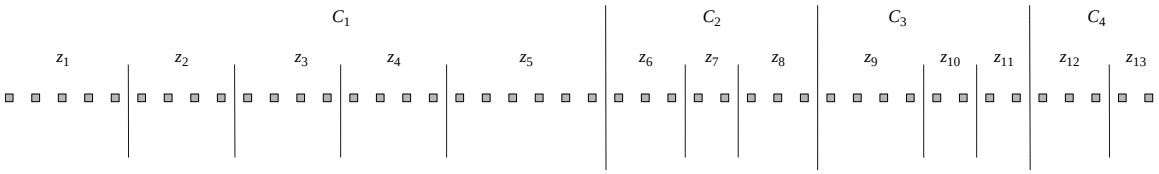


Figure 3.7: Transformation of G' into S

It is obvious from the transformation that a node in G becomes a zone in S. The zones z_i in Figure 3.7 are arranged according to their grouping into cliques, in the order from C_1 to C_4. This is necessary to make sure the neighboring nodes are close to each other in the new arrangement. It is also necessary to minimize the length of each interval to be formed in the subsequent steps as the length of an interval is also the horizontal length of the net.

3.5.3 Formation of Intervals

The third step involves matching the pins $S = \{s_k\}$ into pairs of two to form the intervals I_k, for $j = 1,2,...,N$. Each interval $I_k = (b_k, e_k)$ is made up of a left (beginning) pin b_k and a right (ending) pin e_k. The formation of intervals is based on the width and level of the intervals, similar to the method discussed in our earlier work in [13].

The matching between the nodes in G' can be either intra-clique or inter-clique. A matching is said to be *intra-clique* if the two nodes belong to the same clique. Otherwise, it is *inter-clique*. In both cases, the matchings represent the nets to be drawn from left to right in S.

Intra-clique matching involves a technique as described in Algorithms 1 and 2 in [13]. In this technique, the nets are formed by grouping the intervals into several layers based on their width. The *width* of interval k, denoted by w_k, is defined as the difference between its beginning and end pins, given as $w_k = e_k - b_k$. A *level*, y, in S_m consists of a set of equal-width nets grouped in ascending order from the smallest width

to the largest. Our strategy begins by forming levels where the nets with equal width are grouped. The nets in S_q are created by defining their end-points. Once the nets have been formed, the next step consists of sorting and renumbering the nets based on their beginning pins, in ascending order from the lowest to highest.

The ith net in level y in S_q, denoted as $n_{y,i,q} = (b_{y,i,q}, e_{y,i,q})$, formed from the clique, C_q, is grouped into levels based on its width, $w_{y,q}$, according to the following relationships:

$$b_{y,i,q} = (q - y) + (q - 1)(i - 1), \tag{3.9a}$$

$$e_{y,i,q} = b_{y,i} + w_{y,q} \tag{3.9b}$$

for $y = 1,2,..., q-1$, and $i = 1,2,..., q-1$.

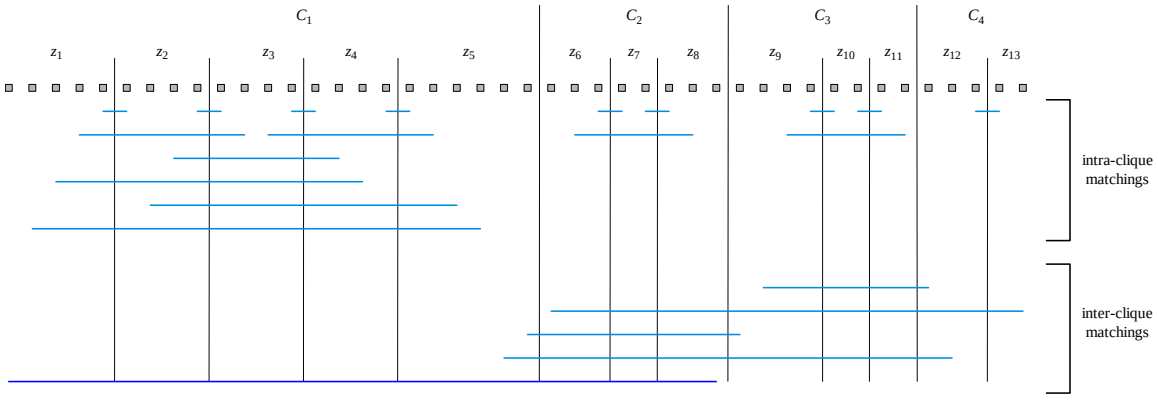


Figure 3.8: Interval graph formation from the matching nodes.

Inter-clique matchings involve the creation of interval $I_k = (b_k, e_k)$ formed from a pin in z_i and another in z_j from the graph relationship (v_i, v_j) where v_i and v_j are not in the same clique in G'. Figure 3.8 shows the intervals formed from $G' = \{C_1, C_2, C_3, C_4\}$ in the inter-clique matching. The results from the inter-clique matchings are tabulated in Table 3.1.

Table 3.1 Matching pairs obtained from the transformation.

net	pins
n_1	(1,31)
n_2	(2,21)
n_3	(3,16)
n_4	(4,11)
n_5	(5,6)
n_6	(7,20)
n_7	(8,15)
n_8	(9,10)

net	pins
n_9	(12,19)
n_{10}	(13,14)
n_{11}	(17,18)
n_{12}	(22,41)
n_{13}	(23,32)
n_{14}	(24,44)
n_{15}	(25,30)
n_{16}	(26,27)

net	pins
n_{17}	(28,29)
n_{18}	(33,40)
n_{19}	(34,39)
n_{20}	(35,36)
n_{21}	(37,38)
n_{22}	(42,43)

3.5.4 Transformation of G into a Single Row S

In the last step, each interval I_k in S is mapped and realized into net n_k using ESSR. The net is drawn from left to right in such a way that it is made up of horizontal and vertical line segments only, and it does not cross another net. We apply ESSR with $I = \{I_k\}$ as the input, to produce an optimal routing with minimum congestion based on the minimum energy E, as described in our earlier work in [1]. Figure 3.9 shows the routing results from the nets in Table 3.1. The results are optimal with $E = 26$, $Q = 3$ and $D = 3$ from the ordering list $L = \{1,14,19,5,20,6,9,21,10,7,16,8,4,17,11,18,22,15,13,3,12,2\}$.

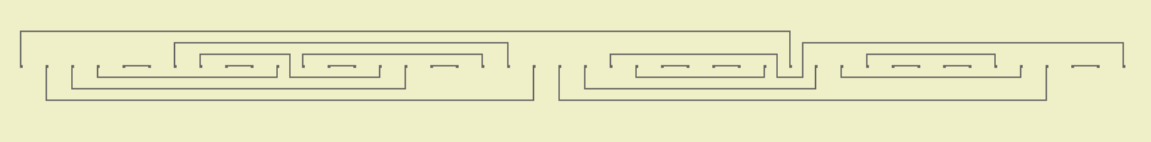


Figure 3.9: Final realization with $E = 26$, $Q = 3$ and $D = 3$, using ESSR.

3.6 Experimental Results and Analysis

Our simulations are performed using AdCliq and ESSR. Both AdCliq and ESSR are user-friendly tools that were developed using C++ on the Windows environment. AdCliq supports a maximum of 100 nodes per graph which are generated randomly in the rectangular area of the window. The connected graph is produced from the random placement of nodes in the plane. The adjacency of the nodes in the graph G is controlled by the preset transmission range R. In this case, a pair of nodes are adjacent (an edge exists between the nodes) if its distance is less than or equal to R. This factor contributes to the total number of links in G, which is the same as the total degrees of all its nodes, GD. R is directly proportional to GD: a high value of R contributes to a high GD, which means a high adjacency rate between the pairs of nodes in G. AdCliq computes the number of cliques of the graph, its maximum clique and the total degrees, and display the results in the window.

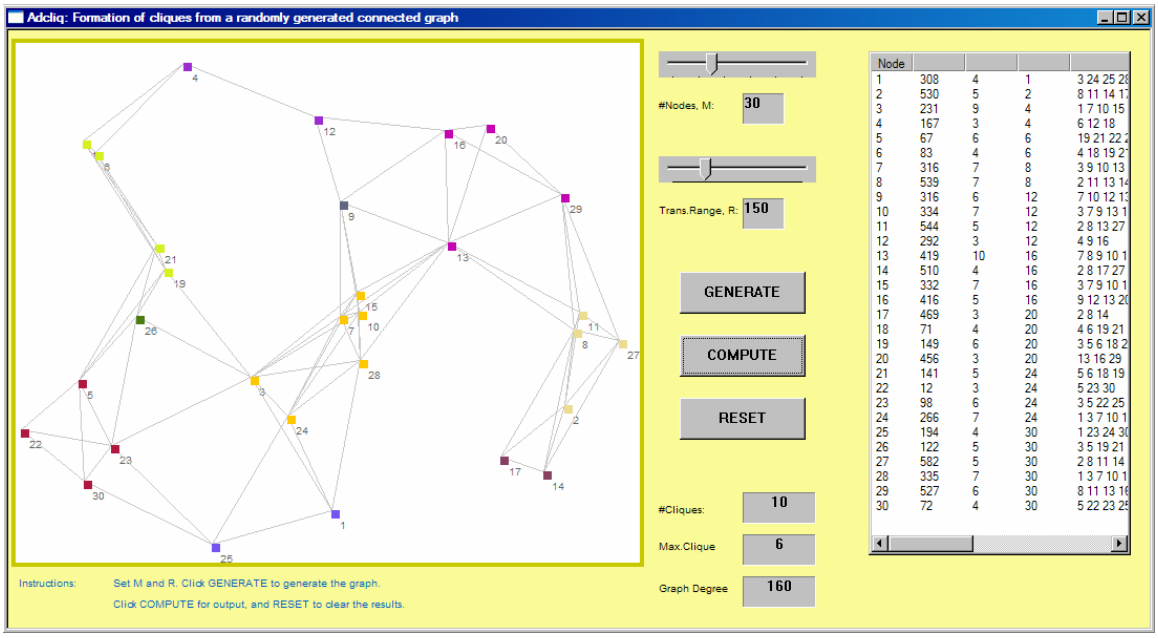


Figure 3.10: AdCliq interface showing the graph in Set 8.

We illustrate the interface in AdCliq using an example of a graph with 30 nodes. Figure 3.10 shows the visual interface of AdCliq with G having 30 nodes scattered

randomly in the rectangular area on the left of the window. A sample run with $R = 150$ produces ten cliques in G, with the maximum clique of six and total node degrees of 160. The maximum clique produced is $C_1 = \{3,7,10,15,24,28\}$. Other cliques are $C_2 = \{6,18,19,21\}$, $C_3 = \{2,8,11,27\}$, $C_4 = \{13,16,20,29\}$, $C_5 = \{5,22,23,30\}$, $C_6 = \{4,12\}$, $C_7 = \{14,17\}$, $C_8 = \{1,25\}$, $C_9 = \{9\}$ and $C_{10} = \{26\}$.

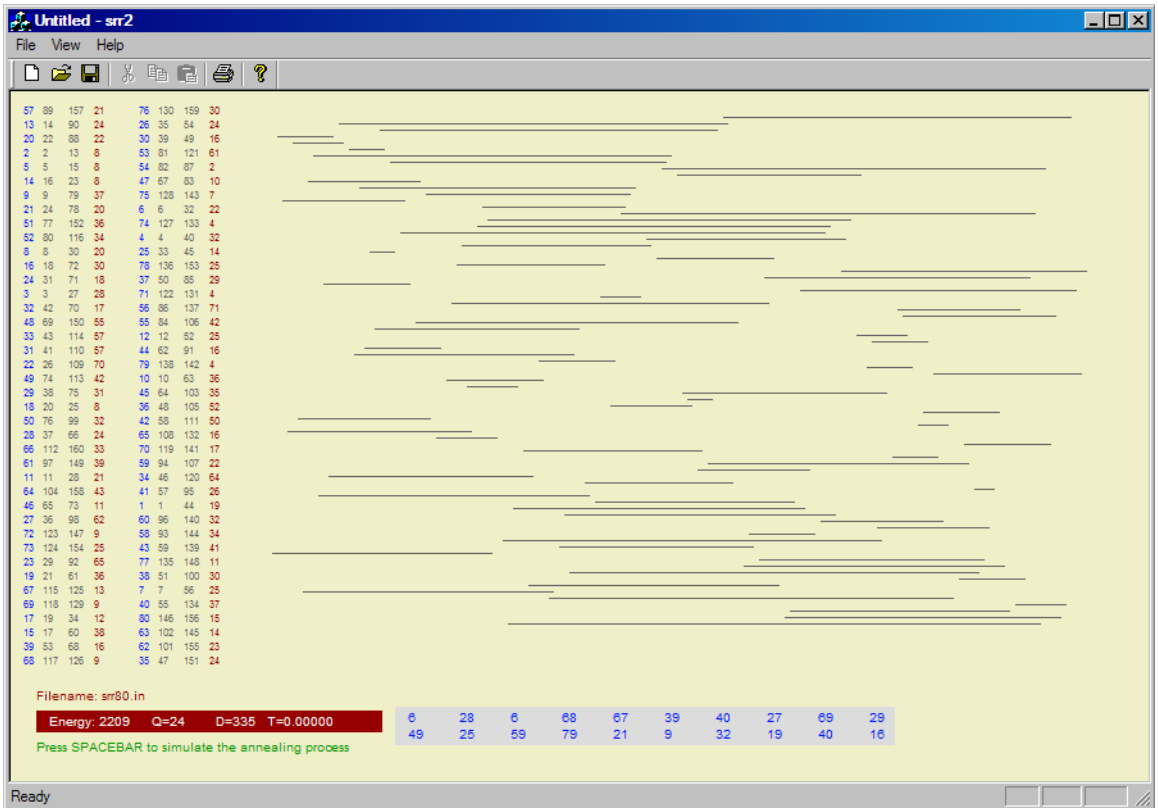


Figure 3.11: Intervals S formed from the graph in Figure 3.10.

Figures 3.12 shows ESSR in forming the intervals between the pins in S using the information from the graph G in AdCliq. The iterations become stable after 2270 iterations with the minimal energy value $E = 2209$, and this results in $Q = 24$ and $D = 335$. The energy E is the optimal value obtained after some massive annealing steps in ESSR which correspond to minimum Q and D based on Equation (3.1). The details on the energy minimization procedures in ESSR can be referred to our earlier work in [4]. This output is shown as a realization in Figure 3.12.

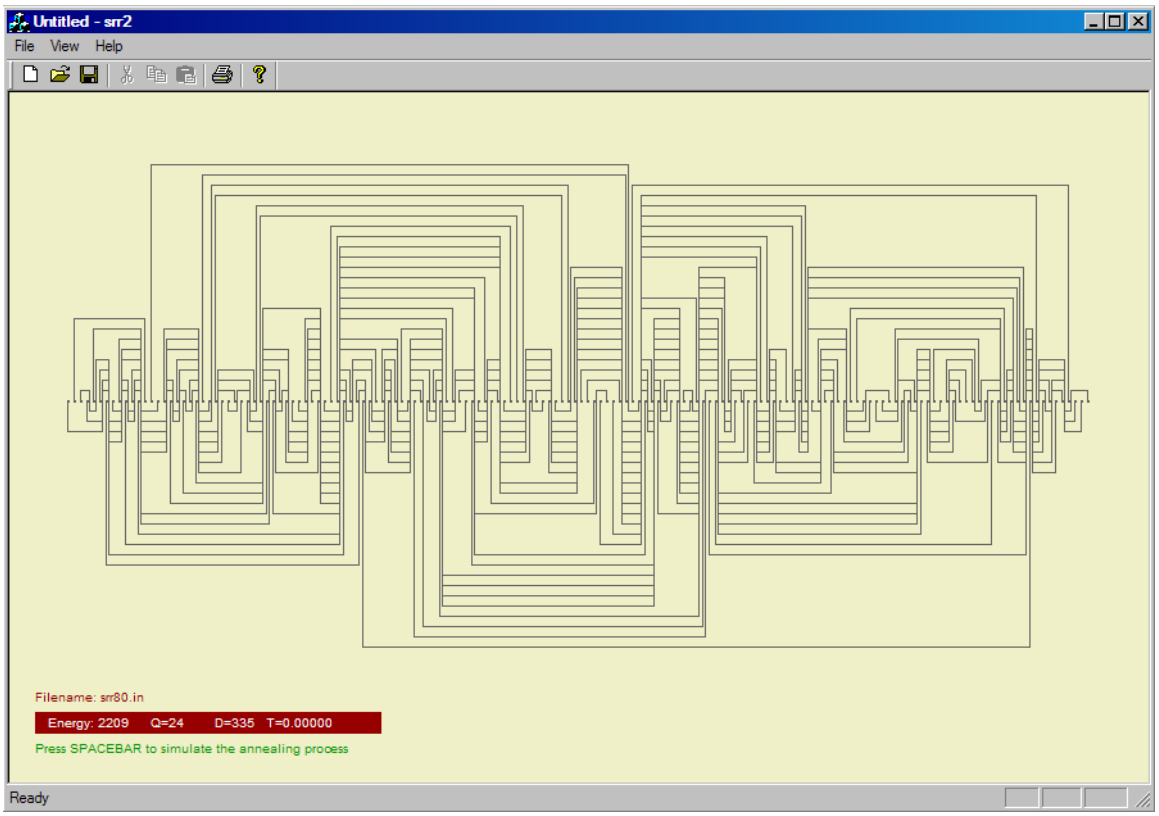


Figure 3.12: Final realization in S with 160 pins.

We perform simulations using AdCliq and ESSR on 14 data sets where the above illustrations make up Set 8. The graphs with between six to 60 nodes are generated using AdCliq, while their transformation into the single rows are performed using ESSR. Table 3.2 shows the results from the simulation on the data sets. The graphs information using AdCliq is shown in columns 2 through 6 in the table. AdCliq computes the number of cliques, the maximum clique MC and the total number of degrees GD of each graph. It then follows with the transformation of each graph G into a row of pins using ESSR, with the information on the transformation shown in columns 7 to 11 in the table. In the transformation process, the number of nodes in G maps directly as the number of zones in S. In addition, each node degree in G maps to a pin in S. The output in the form of energy E, congestion Q and number of doglegs D from the simulated annealing processes are shown in columns 9, 10 and 11, respectively.

Table 3.2: Sample results from the simulations.

Set	G Partitioning using AdCliq					S transformation using ESSR				
	M	R	#Cliques	MC	GD	#Zones	#Pins	E	Q	D
1	6	250	3	3	12	6	12	12	2	3
2	8	250	4	3	22	8	22	20	3	4
3	10	250	4	4	28	10	28	24	4	2
4	13	200	4	5	44	13	44	26	3	3
5	15	200	5	5	68	15	68	151	10	30
6	20	200	8	5	80	20	80	343	12	36
7	25	200	9	6	134	25	134	994	19	157
8	30	150	10	6	160	30	160	2209	24	335
9	35	150	12	6	204	35	204	3872	37	425
10	40	150	14	8	232	40	232	4817	48	663
11	45	150	13	9	310	45	310	6534	57	847
12	50	100	13	7	414	50	414	9590	70	994
13	55	100	14	8	556	55	556	12954	88	1134
14	60	100	15	10	634	60	634	15337	102	1529

Figure 3.13 shows the relationships between the number of nodes M in the graph with the energy (left), and congestion and number of doglegs (right) on the sets of data in Table 3.2. It is obvious from the graphs that the complexity of the problem increases exponentially as the number of nodes increases.

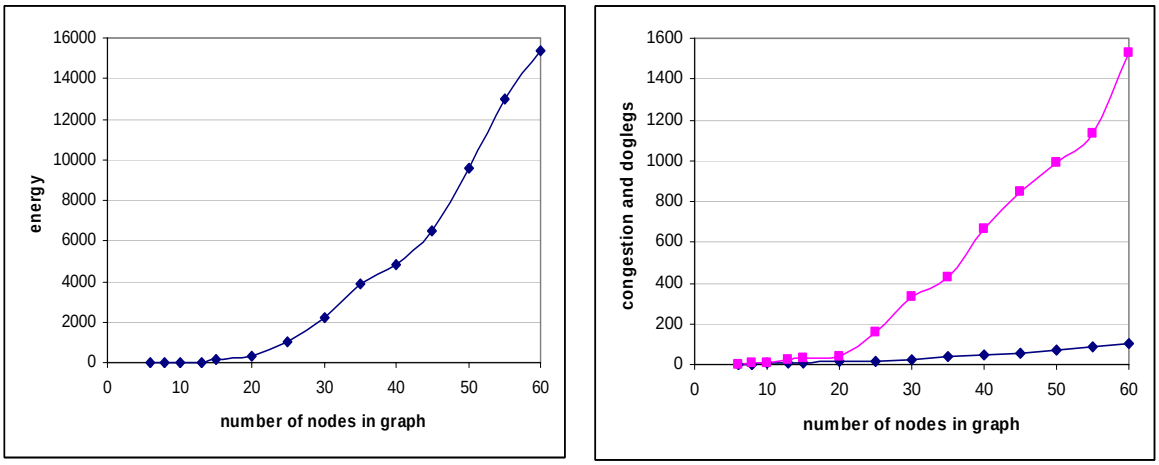


Figure 3.13: Relationships between the number of nodes M with E, Q and D.

The transformation problem has a number of open problems in the form of theoretical foundation and applications. The transformation model from a connected graph to the single row network has a huge potential for a wide range of applications. In [13], we discussed two potential applications, namely the channel assignment problem and the multiprocessor routings. Both problems may be explored further as there are many open problems relevant to the transformation model discussed in this chapter.

We discuss briefly the potential application of the transformation model to the channel assignment problem. The case mentioned in [13] is specific to a complete graph where all nodes in the graph have equal degrees as they are connected to each other. Therefore, the transformation is made in such a way that every zone has an equal number of pins (channels). A zone in this model may represent a cell in the network. Practically, the model may not apply well as the demand for channels may differ between the cellular cells in the network. The present model is more flexible as it caters to the non-uniform channel requirements as the degree of each node in the graph varies. As shown in Figure 3.8, a clique instead of a zone may represent a cell in this case. In turn, a zone may represent the mobile subscriber who may want to communicate with more than one person in the network as specified by its node degree.

3.7 Summary and Conclusion

In this chapter, the single-row transformation model of a connected graph has been presented. The transformation model is proposed as a multi-commodity problem in the form of supply-demand mapping. In the demand-supply dimensional problem, the connected graph may represent the demand flows between the components in the graph while the network supporting it is the resource or capacity links for supporting the demand volumes. The main idea behind this transformation model is to provide a reasonably good optimal single-row routing network that minimizes the congestion in the network. We propose AdCliq which is a technique for partitioning a given graph into several disjoint cliques using the Hopfield neural network. From the cliques, a set of intervals derived from the zones are obtained through the matching nodes in the single-row axis. The intervals are then mapped into a network of non-crossing nets using our previously developed tool called ESSR. The network is optimal in terms of minimal street congestion and number of doglegs, and this provides a reasonably good step towards the overall solution to the demand-supply problem.

Chapter 4

FRECAST: MESSAGE TRANSMISSION MODEL FOR AD HOC NETWORKS

4.1 Introduction

Since 1970s, wireless networks have become increasingly popular in the computing industry. There are currently two variations of mobile wireless networks, networks with infrastructure and those without infrastructure. A network with infrastructure has fixed and wired gateways for providing communicating services to the nodes in the network. An infrastructureless network has no fixed equipments for transmitting and receiving messages. The latter is commonly referred to as the *ad hoc network*.

Ad hoc network is a self-organized wireless computing network made up of nodes that are capable of communicating with each other in the absence of infrastructures such as base stations and mobile switching centers. Each node in the network has a transceiver for both transmitting and receiving messages within a limited transmission range. Each node is also limited in its activity due to its short lifespan in the form of a small battery supply. The absence of infrastructures means the nodes have to organize themselves to form the network without the help from external sources. The nodes in the network can be mobile and are connected dynamically in an arbitrary manner. Nodes of

these networks function as routers which discover and maintain routes to other nodes in the network. These networks typically consist of equal nodes that communicate over wireless links without central control.

Ad hoc networks are becoming increasingly important for various applications. Example applications of ad hoc networks are emergency search-and-rescue operations, meetings or conventions in which persons wish to quickly share information, and data acquisition operations in inhospitable terrain. There is a lot of interest in the military, where research has been conducted to apply ad hoc networks to critical operations such as in search-and-rescue missions, and combat operations in the presence of inhospitable environment. In science, a variant of ad hoc networks called the sensor networks are being deployed to detect certain types of chemicals in various biochemical applications. There are applications to enable several cars in a geographical area to communicate with each other in order to exchange information. It is expected that ad hoc networks will make an impact in the way people communicate in years to come.

Ad hoc network is not fully stable yet as there are several issues in the design and development that need to be tackled. The problems encountered in ad hoc networks include topology control, message transmission and service access. In this chapter, we discuss the message transmission issues in ad hoc involving single and multicasting. We relate this problem to several fundamental graph theory and optimization problems, such as the shortest path and minimum spanning tree. Our work solves the single-casting and multicasting problems from one or more nodes in a dense graph require finding the shortest paths in order to make the message transmission optimum. We also explore the broadcasting problem from a single source. A model called FRECAST is proposed for visualizing these problems. FRECAST has its origin in a routing protocol called WRP [19].

The chapter is organized into six sections. Section 4.1 is the introduction. This is followed by the description of the message transmission problems in ad hoc networks in Section 4.2. Section 4.3 discusses several wireless routing protocols for ad hoc networks. This is followed by some discussion on FRECAST in Section 4.4. We also

describe the development of FRECAST involving the single-casting model based on the Floyd-Warshall's shortest path algorithm and the broadcasting method using the Prim's algorithm. Section 4.5 describes our visual tool, FRECAST, developed using C++ for simulating the ad hoc networks. Section 4.6 is the summary and conclusion.

4.2 Problem Formulation

Our problem involves finding a practical method for single-casting and multicasting in an ad hoc network. We also explore the problem of broadcasting from a single source which is a problem of finding the minimum spanning tree. *Single casting* involves the transmission of a message between a pair of nodes, while *multicasting* is the simultaneous transmission of a message to a group of nodes in the network. Another term, *broadcasting* is the transmission of a message to all nodes in the network. Our strategy consists of studying a protocol for routing and improves its performances through the development of a model called FRECAST.

Consider a dynamic programming technique for solving discrete optimization problems which are shortest path and minimum spanning tree problems. In this study, the single-casting and multicasting problems from one or more nodes in the graph require finding the shortest paths in order to make the message transmission optimum. The multicasting and broadcasting methods in our model is based on the minimum spanning tree (MST) with the same purpose to achieve the optimal message transmission. The graph contains n nodes numbered $0,1,\ldots,n-1$, and an edge connecting node i to node j has cost $d(i,j)$. Let $f(x)$ denote the cost of the least cost path from node 0 to node x. Thus, $f(0)$ is zero, and finding $f(n-1)$ solves the problem. The dynamic programming formulation for this problem yields the following recursive equations for $f(x)$:

$$f(x) = \begin{cases} 0 \\ \min\limits_{0 \le j \le n-1} \{ f(j) + d(j,x) \} \end{cases} \tag{4.1}$$

Communication between two nodes in an ad hoc network is not always direct; it can proceed in a multihop fashion so that every node is also a router. Because power consumption is directly proportional to the distance between nodes, direct single-hop transmissions between two nodes can require significant power, causing interference with other such transmissions. To avoid this routing problem, two nodes can use multihop transmission to communicate via other nodes in the network.

Two nodes that do not communicate directly can transmit messages simultaneously to a common neighbor on the same frequency. In addition to maintaining an ongoing routing task or facilitating route establishment, mobile networks also must support location management by keeping track of the node's location. Topology control problem include discovering neighbors, identifying position, determining transmission radius, establishing links to neighbors, scheduling node sleep and active periods, clustering, constructing the dominating set and maintaining the selected structure. Message transmission problems include routing, broadcasting, single-casting, multicasting, geo-casting and location updating. While service access problems consist of Internet access, cellular network access, data or service replication upon detection or expectation of network partition and unique IP addressing in merge network scenarios.

4.3 Review of Ad Hoc Networks

4.4.1 Ad Hoc Network Routing Protocols

Since the advent of Defense Advanced Research Projects Agency (DARPA) packet radio networks in the early 1970s, numerous protocols have been developed for ad hoc mobile networks. However, the routing protocols meant for wired networks cannot be used for mobile ad hoc networks because of the mobility of the ad hoc networks. The nodes in a ad hoc network are mobile and this causes the network to change its topology most of the time. The constantly changing topology makes it difficult to implement a specific technique with regard to routing, message transmission and message safety.

There are two types of routing in ad hoc networks, namely, *table-driven routing* and *on-demand routing*. In a table-driven routing protocol, each node maintains one or more tables containing routing information to every other node in the network. All nodes update these tables periodically so as to maintain a consistent and up-to-date view of the network. Some established table-driven routing protocols are *Destination-Sequenced Distance-Vector Routing* (DSDV), *Clusterhead Gateway Switch Routing* (CSR) and *Wireless Routing Protocol* (WRP) [17]. In DSDV, every node maintains a routing table that lists all the available destinations, number of hops to reach the destination and sequence number assigned by the destination node. CSR protocol is a clustered multihop mobile wireless network with several heuristic routing schemes. By having a cluster head controlling a group of ad hoc nodes, a framework for code separation (among clusters), channel access, routing and bandwidth allocation can be achieved. While in WRP, the goal is to maintain the routing information among all nodes in the network.

A different approach from table-driven routing is the on demand-driven routing where routes are created only when desired by the source node. In contrast to table-driven routing protocols, all up-to-date routes are not maintained at every node. Instead, the routes are created as and when required. When a source wants to send a message to a destination, it invokes the route discovery mechanisms to find the path to the destination. The route remains valid till the destination is reachable or until the route is no longer needed. The on-demand routing protocols include *Ad Hoc On-Demand Distance Vector Routing* (AODV), *Dynamic Source Routing* (DSR), *Temporally Ordered Routing Algorithm* (TORA), *Associativity Based Routing* (ABR) and *Signal Stability Routing* (SSR) [17].

AODV is an improvement to DSDV where it minimizes the number of broadcasts by creating routes on-demand, as opposed to DSDV that maintains the list of all the routes. In DSR, a node maintains route caches containing the source routes that it is aware of. The node updates entries in the route cache as and when it learns about new routes. TORA is a highly adaptive, efficient and scalable distributed routing algorithm based on the concept of link reversal where the main feature is that the control messages

are localized to a very small set of nodes near the occurrence of a topological change. The ABR protocol is free from loops, deadlock, and a packet duplicates, and defines a new routing metric for ad hoc mobile networks. In ABR, a route is selected based on the degree of association stability of mobile nodes. While SSR selects routes based on the signal strength between nodes and a node's location stability. This route selection criteria has the effect of choosing routes that have stronger connectivity.

4.4.2 Wireless Routing Protocol

The *wireless routing protocol* (WRP) is an excellent alternative for routing in wireless networks [19]. WRP is a table-based distance-vector routing protocol where each node in the network maintains a *distance table* (DT), a *routing table* (RT), a *link-cost table* (LT) and a *Message Retransmission List table* (MRL). This table is updated at every time slot in order to provide the correct information about each node's connectivity.

The distance table of a node *x* contains the distance of each destination node *y* via each neighbor *z* of *x*. It also contains the downstream neighbor of *z* through which this path is realized. The routing table of node *x* contains the distance of each destination node *y* from node *x*, the predecessor and the successor of node *x* on this path. It also contains a tag to identify if the entry is a simple path, a loop or invalid. Storing predecessor and successor in the table is beneficial in detecting loops and avoiding counting-to-infinity problems. The link-cost table contains cost of link to each neighbor of the node and the number of timeouts since an error-free message was received from that neighbor. The MRL contains information to let a node know which of its neighbor has not acknowledged its update message and to retransmit update message to that neighbor.

To describe WRP, we model a network as an undirected graph represented as *G(V, E)*. Each node represents a router and is a computing unit involving a processor, local memory and input and output queues with unlimited capacity. For routing table updating, every node can consider another node to be adjacent if there is radio connectivity between the two nodes. For the neighboring nodes *x* and *y*, node *x* may receive an update message from node *y*. Accordingly, we map a physical broadcast link

connecting multiple nodes into multiple point-to-point functional links defined for these node paths that consider being neighbors of each other. Each connected nodes is assigned a positive weight.

The communication links in the network are such that all update messages transmitted over an operational link are received in the order in which they were transmitted within a finite time. A link is assumed to exist between two nodes only if they are connected and they can exchange update messages reliably with a certain probability of success [19]. Every node informs its neighbors about changes through this update. For example, for the link $x \to z \to y$, z is adjacent to both x and y. Since x and y are not connected directly, then one or more neighboring nodes are needed to transmit a message from x to y. WRP requires the update message to be sent between node x and z. The update message contains a list of table updates on the destination, the distance to the destination and the predecessor of the destination. The update message also includes a list of responses indicating which nodes should acknowledge (ACK) the update [17].

After receiving an update message, a node is required to send positive ACK indicating that it has a good connectivity and has processed the update messages. If the update messages are lost or corrupted due to changes in a link to a neighbor or jamming, the node send update messages to their neighbors. The neighbors then update their distance table entries and check for new possible paths through other nodes. Any new paths are relayed back to the original nodes so that they can update their tables accordingly.

A node failure occurs when all links incident on that node fail simultaneously. Nodes learn of the existence of their neighbors from the receipt of ACKs and other messages. If a neighboring node is not sending messages, a simple flag like a *hello* message is expected from this node within a specified time period. Otherwise, the node assumes connectivity with this neighbor has been lost. When a link fails, the corresponding distance entries in node's distance and routing tables are marked as infinity. WRP has been successful in achieving its loop freedom this way.

4.4 FRECAST: Single-casting, Multicasting and Broadcasting

Ad hoc network can be modeled as an undirected graph represented as $G(V, E)$, where V is the set of nodes and E is the set of links (edges) connecting the nodes. Our model is called FRECAST, or *frequency casting*, which is based on WRP. FRECAST provides optimum routing based on the shortest path and minimum spanning tree to support both the single-casting and multicasting modes of message transmission. In addition, the simulation model also supports message broadcasting from a single node. FRECAST was developed using the C++ programming language on the Windows environment.

4.4.1 Single-casting Method

Single-casting involves the transmission of a message from a source node to a destination node in the network. In transmitting the message, the shortest path linking the source node and its destination node is desired in order for the network to achieve its maximum performance. This optimal message transmission is needed in the ad hoc network to save energy by reducing the total links cost that participate in the routes. The shortest path between any two nodes in a graph is determined using methods such as Dijkstra's algorithm and Floyd's algorithm. We adopt the Floyd-Warshall's algorithm in FRECAST as it solves the all-pairs shortest paths in a graph, which can be applied to both single and multicasting.

A wireless link among two nodes *i* and *j* is established when the distance $D(i, j)$ is less than or equal to the transmission radius *R* (a typical value for *R* is 250 m). Due to the node mobility, new wireless links are continuously created and existing ones deleted, leading to a so-called random unit graph.

A path $P(S, D)$ from *S* to *D* is specified as a sequence of nodes

$$P(S, D) \equiv \langle v_0, v_1, \ldots, v_k \rangle \tag{4.2}$$

where

$$v_0 = S \quad , \quad v_k = D$$
$$(v_i, v_j) \in E(\text{for } i \neq j, 0 \leq i \leq k-1)$$

The length of the path, $|P(S,D)|$, is the number of links (hops) a message performs before reaching the destination node, thus $|\langle v_0, v_1, \ldots, v_k \rangle| = K$. The term route is also used as a synonym of path. The task of routing can be divided into two sub-tasks:

1. The computation of $P(S,D)$
2. Data (message) packet forwarding along the route

Paths are calculated by a distributed algorithm running at network level, which includes some form of path maintenance to deal with topological changes. Maintenance can be provided by "sampling" the current network topology with periodic route updates and optionally sending updates upon a change in network topology. In this way, the algorithm should track topological changes and consistently update paths. Also, in the very likely case that many paths exist between *S* and *D*, the routing protocol has to calculate the best path *P**, in this case *P** is the shortest-path:

$$P^*(S,D) = \min_P \{|P(S,D)|\} \tag{4.3}$$

We illustrate FRECAST through an example based on Figure 1. The figure shows a connected graph $G(V,E)$ where $V = \{v_1, v_2, v_3, v_4, v_5\}$ and E is the set of links. The blue line presents the selected route for message transmission between v_1 and v_5. v_1 sends all the four tables, namely, Distance Table (DT), Routing Table (RT), Link-Cost Table (LCT) and Message Retransmission Link Table (MRL) to all neighbors. It can be shown from Figure 1 that DT consists of the distance of each destination v_5 via each neighbor of v_1. The RT in v_1 contains the distance of each destination v_5 from v_1,

the predecessor and the successor of v_1 on this path. While the LCT contains cost of link to each neighbor of node 1. The MRL table consists of the information to let node 1 know which of its neighbor has not acknowledged its update message and to retransmit update message to that neighbor. In the table, 1 represents the neighbor that acknowledged the update messages, otherwise represents as 0.

In our model, updated messages are sent between a node and its neighbors, for this example, an updates are sent between v_1, and its neighbors v_2, v_3 and v_4. v_2 is required to send positive ACK indicating that it has a good connectivity and has transmitted the update messages to its neighbors. Furthermore, after receiving updates from its neighbors, the information becomes available to the node. In order to achieve an optimum message transmission, v_2 is selected. v_2 then modifies its distance table entries and check for new possible paths through other nodes. Any new paths are relayed back to the original nodes so that they can update their tables accordingly.

The same procedures are repeated until the path reaches v_5. The shortest path is shown to be $v_1 \rightarrow v_2 \rightarrow v_5$ with a total cost of 5. This cost is minimum and it is an optimal message transmission in this network.

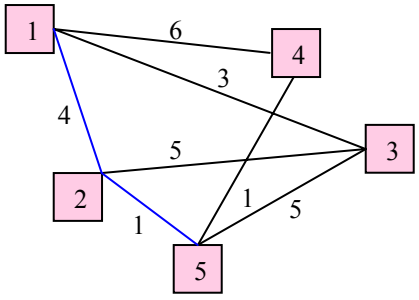


Figure 4.1: A graph with five nodes.

4.4.2 Broadcasting Method

The network is modeled as a graph $G = (V, E)$, where V represents the set of nodes (hosts) in the network, and E is the links. If $u, v \in V$, then an edge $e = (u, v) \in E$ exists if and only if u is in the transmission range of v and vice versa. The length of the broadcast packet is fixed. While broadcasting, no carrier sense and collision avoidance procedure is executed before transmission. The network is assumed to be connected. If it is partitioned, each component is treated as an independent network.

Assume that S is a source node and D is a set of destination nodes, the minimum broadcasting problem is a special case of the minimum multicast tree problem where

$$D = V - \{S\} \tag{4.4}$$

Given a wireless ad hoc network G and a source node S, to broadcast a message from S to all the other nodes such that the sum of transmission distances at all nodes is minimized. In other words, the task is to construct a broadcast spanning tree rooted at the source node and spans all the other network nodes such that the sum of transmission distances at non-leaf nodes is minimized for every new broadcast session.

Consider a connected graph $G = (V, E)$, which contains a tree $T = (V', E')$. The edges E', of T are called *branches* and the edges of G which are not in T are called *chords* (both relative to T). If $V' = V$ then T is said to be a spanning tree of the graph [16], if G has a unique spanning tree then G is itself a tree and that spanning tree is G itself. Further, each component of an arbitrary graph G will contain at least one spanning tree. A collection of such spanning trees, one for each component of G, is termed a spanning forest.

A graph G is called a tree if it is connected and has no circuits. The following properties are equivalent for characterizing a tree [15]:

(i) G is connected and contains no circuits.

(ii) *G* is connected and every edge of *G* is a bridge.

(iii) Every pair of nodes of *G* is joined by precisely one chain.

(iv) *G* contains no circuits, but the addition of any new edge creates a circuit.

The minimum spanning tree of the graph is a spanning tree of the graph whose total weights is a minimum. We define the minimum spanning tree problem as follows: Given an undirected graph *G* with real-valued edge weights assigned to each of its edges, find a spanning tree *T* of *G* that has a minimum total edge weight. The problem has a simple interpretation. Consider the complete graph *K(n),* where the weight of an edge corresponds to the cost of establishing a direct communication link between the pair of nodes corresponding to the endpoints of the edge. A minimum spanning tree on *K (n)* corresponds to a least cost communication network connecting all the nodes.

The optimal solution to the minimum spanning tree problem is surprisingly simple. We merely construct a tree, starting with an empty tree, iteratively adding edges to the tree in increasing order of edge weight, excluding an edge only if it forms a cycle with the previously added edges. The algorithm is greedy because it optimizes the possible improvement in the objectives function, here the weight of the spanning sub graph, at each successive step of the algorithm, subject only to the constraint that no cycles are formed.

A minimum spanning tree can be grown from an arbitrarily chosen node $x^{(0)}$, one edge at a time, by constructing a sequence of graphs $T^{(k)} = \left(X^{(k)}, F^{(k)}\right), (k = 0,1,\ldots,n-1)$ as follows [18].

Step 1

Let $x^{(0)}$ be any node of *G*, and let $X^{(0)} = \left\{x^{(0)}\right\}, F^{(0)} = \phi.$

Step 2

For $k = 1,2,\ldots,n-1,$ form $X^{(k)}$ and $F^{(k)}$ as follows:

Let $C^{(k)}$ be the cut set of edges which have exactly one endpoint in $X^{(k-1)}$, let $e^{(k)}$ be a minimum edge of $C^{(k)}$, and let $x^{(k)}$ be the endpoint of $e^{(k)}$ which does not belong to $X^{(k-1)}$. Then set $X^{(k)} = X^{(k-1)} \cup \{x^{(k)}\}$, $F^{(k)} = F^{(k-1)} \cup \{e^{(k)}\}$.

On termination, the graph $T^{(n-1)} = (X^{(n-1)}, F^{(n-1)})$ is a shortest spanning tree of *G*.

4.4.3 Multicasting Method

This section discusses the multicasting method in ad hoc networks. Multicasting involves the transmission of the same message from a source node to a set of destination nodes. The minimum paths are required in order to make the message transmission optimum. In this method, minimum spanning tree problem can be applied using Prim's algorithm. A *spanning tree* is subgraph of a connected graph which forms a tree that includes all the nodes in the graph. The *minimum spanning tree* of a graph is a spanning tree of the graph whose total weights is the most minimum.

The multicasting problem is defined as follows. Given a wireless ad hoc network $G = (V, E)$, a source node and a terminal set $D(\subset V)$, to broadcast a message from a source to the nodes *D* such that the sum of transmission distances consumed at all involved nodes (some nodes are relay nodes) is minimized. The task is to construct a multicast tree rooted at the source node including all the nodes in *D* such that the sum of transmission distances at non-leaf nodes in the tree is minimized, for every new multicast session.

We discuss the Prim Algorithm for finding a minimum spanning tree in a connected weighted graph. This algorithm employs a different method from Kruskal's algorithm for selecting the edges to produce the tree. It is normally attributed to Prim, who described it in 1957. However, Jarnik had essentially discovered it in 1930. The strategy in this algorithm is to replace a current tree *T* in a connected weighted graph *G* by a new tree formed by adding an edge of minimum weight that joins a node of *T* to a node not in *T*.

To determine a minimum spanning tree in a nontrivial connected weighted *(p, q)* graph *G* [16]:

Step 1:

Initialize a tree T. Let u be an arbitrary node of G and T← u

Step 2:

The tree T is updated. Let e be an edge of minimum weight joining a node of T and a node not in T, and T←T+e.

Step 3:

This step determines whether a minimum spanning tree has been constructed. If $|E(T)| = p-1$, then output E (T); otherwise, return to Step 2.

In Step 2, if *T* has *k* nodes, then there are $p-k$ nodes not in *T*. Thus each node *v* of *T* is adjacent to at most $p-k$ nodes of *G* that are not in *T*. Therefore, we need to find the minimum weight of at most $k(p-k)$ edges. Since $k(p-k) < (p-1)^2$, the complexity of Step 2 is $O(p^2)$. Since Step 2 is performed $p-1$ times, the complexity of Prim algorithm is $O(p^3)$.

4.4.4 Message Passing in the Ad Hoc Networks

The message passing paradigm is based on just two primitives: `SEND` and `RECEIVE`. `SEND` transmits a message from one node to another, and `RECEIVE` reads a message from another node [20]. The general form of the `SEND` primitive is

```
SEND (message,target)
```

`Message` contains the data to be sent, and `target` is the label of the destination node. Sometimes, `target` can also specify a set of nodes as the recipient of the message.

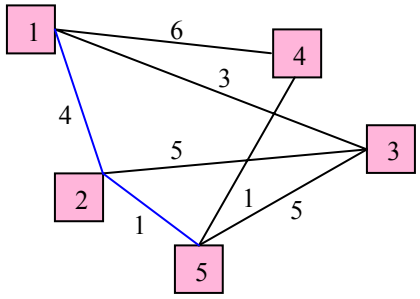
When `SEND` operation is executed, the operating system performs the following steps. It copies the data stored in `message` to a separate area in the memory, called the

communication buffer. Then, it sends the message. When the message arrives at the destination node, it is copied into this node's communication buffer and a system variable is set indicating that a message has arrived. However, the actual transfer data does not occur until the receiving node executes the corresponding `RECEIVE` operation (Figure 4.2).

The `RECEIVE` operation reads a message from the communication buffer into user memory. The general form of the `RECEIVE` primitive is

```
RECEIVE (message,source)
```

There is a great deal of similarity between the `RECEIVE` and `SEND` operations because they perform complementary operations. The `message` parameter specifies the location at which the data will be stored. At any time, more than one message may be stored in the communication buffer. These messages may be from the same node or different nodes. The `source` parameter specifies the label of the node whose message is to be read.




```
Node1
{
  int u;
  SEND(u,Node2);
}
```

```
Node2
{
  int w;
  RECEIVE(w,Node1);
  SEND(w,Node5);
}
```

```
Node5
{
  int z;
  RECEIVE(z,Node5);
}
```

Figure 4.2: Message Passing in the Ad Hoc Network.

4.4.5 IEEE 802.11 MAC Protocol

The basic access method in the IEEE 802.11 MAC protocol is the *Distributed Coordination Function* (DCF), which is a *Carrier Sense Multiple Access with Collision Avoidance* (CSMA/CA) MAC protocol. Besides the DCF, the IEEE 802.11 also incorporates an alternative access method known as the *Point Coordination Function* (PCF). The PCF operates similarly to a polling system; a point coordinator provides the transmission rights at a single station at a time. As the PCF access method cannot be adopted in ad hoc networks, we will concentrate on the DCF access method only.

When using the DCF, before a node initiates a transmission, it senses the channel to determine whether another node is transmitting. If the medium is found to be idle for an interval that exceeds the *Distributed InterFrame Space* (DIFS), the node continues with its transmission. Each active node stores this information in a local variable named *Network Allocation Vector* (NAV). Therefore, the NAV contains the period of time the channel will remain busy.

The CSMA/CA protocol does not rely on the capability of the nodes to detect a collision by hearing their own transmissions. Hence, immediate positive acknowledgements are employed to ascertain the successful reception of each packet transmission. Specifically, the receiver after the reception of the data frame, the receiver has to:

1. Waits for a time interval, called *Short InterFrame Space* (SIFS), which is less than the DIFS.
2. Initiates the transmission of an *acknowledgement* (ACK) frame.

The ACK is not transmitted if the packet is corrupted or lost due to collisions. A *Cyclic Redundancy Check* (CRC) algorithm is adopted to discover transmission errors. Collisions among nodes occur when two or more nodes start transmitting at the same time. If an acknowledgement is not received, the data frame is presumed to have been lost, and retransmission is scheduled. The DCF access method is summarized in Figure 4.3.

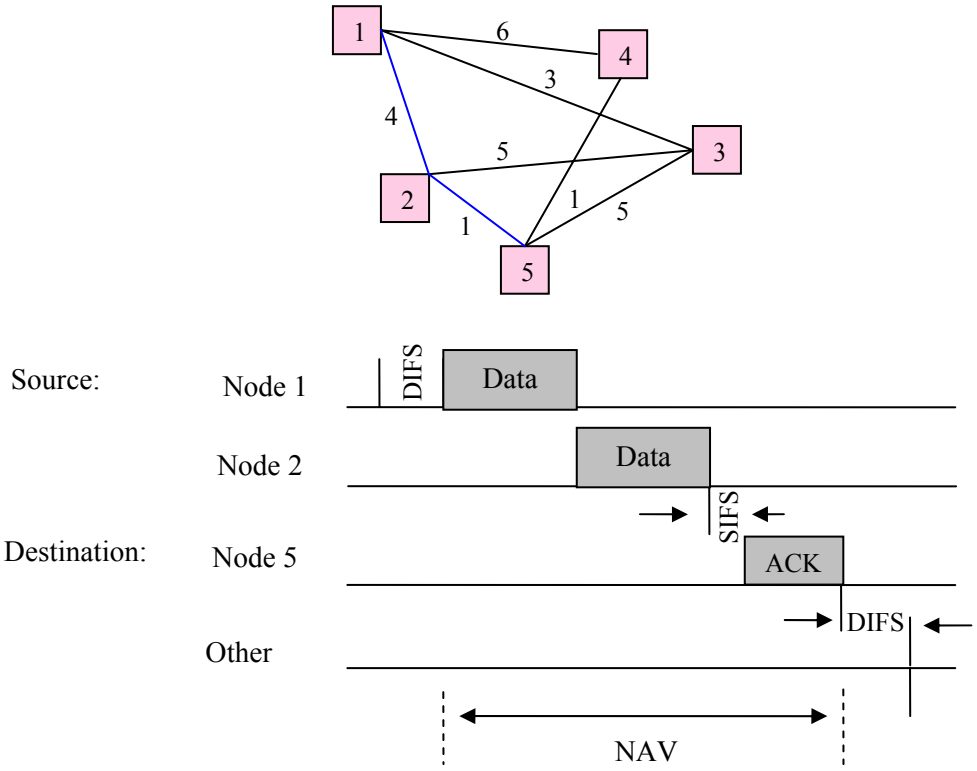


Figure 4.3: The DCF Access Method.

4.5 FRECAST Simulations

Our simulation model was developed using the C++ programming language, running on the windows environment. Figure 4.4 shows a FRECAST single-casting simulation using the randomly generated graph with 20 nodes. Figure 4.5 shows the spanning tree that shows the path for broadcasting from a single node on the graph as in Figure 4.4. While Figure 4.6 shows the Gantt chart of minimum spanning tree for broadcasting on the graph as in Figure 4.4.

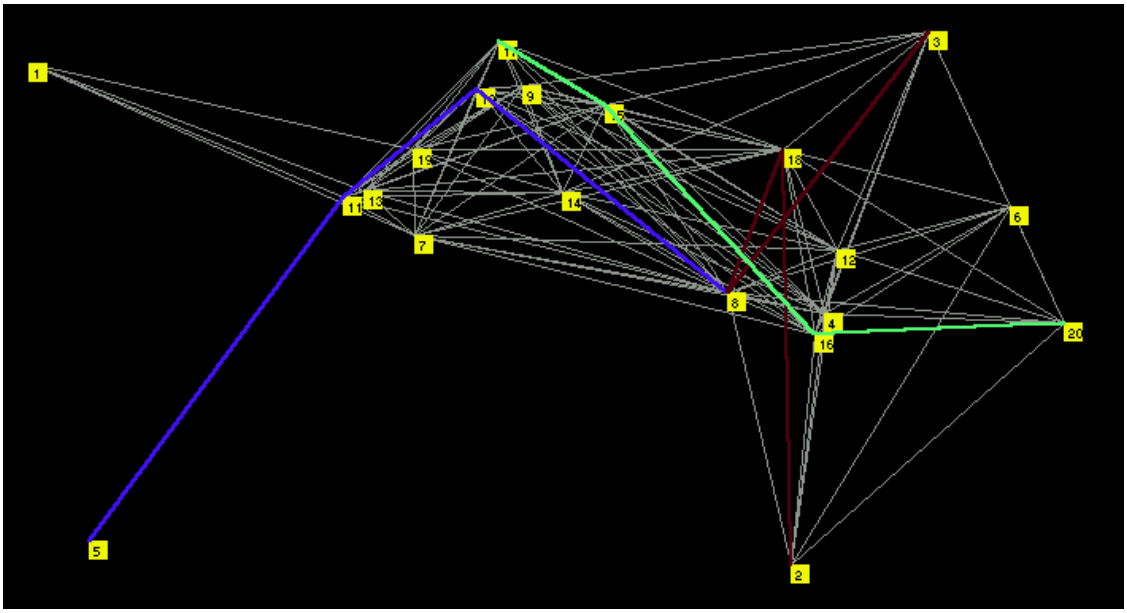


Figure 4.4: Single-casting using FRECAST.

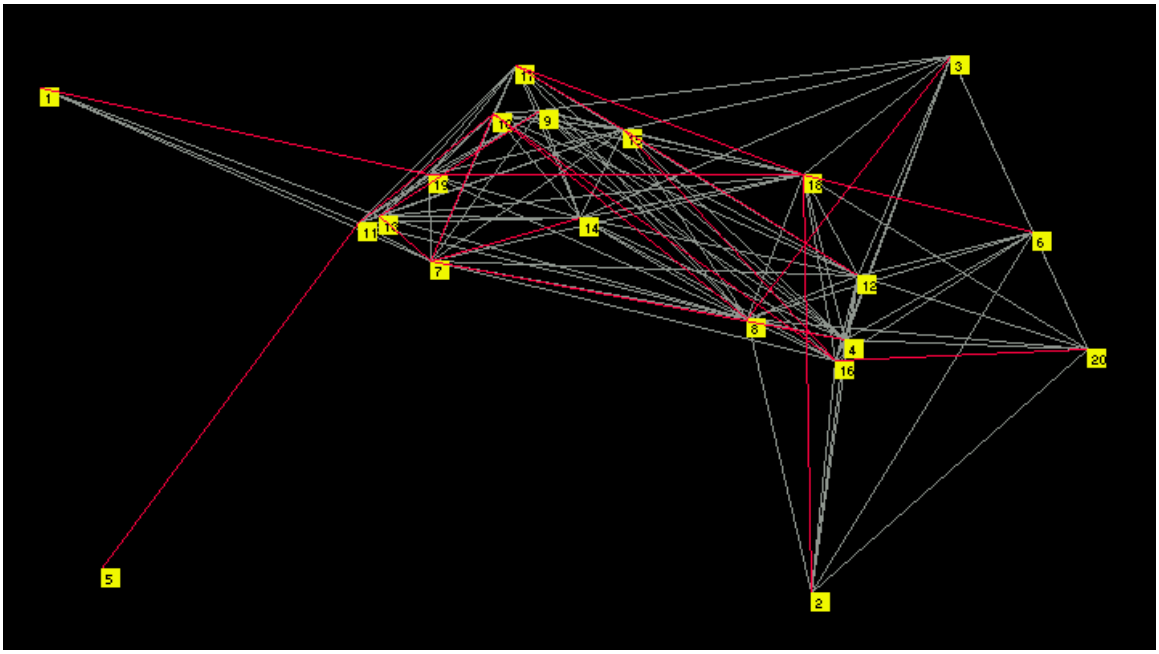


Figure 4.5: Broadcasting using FRECAST.

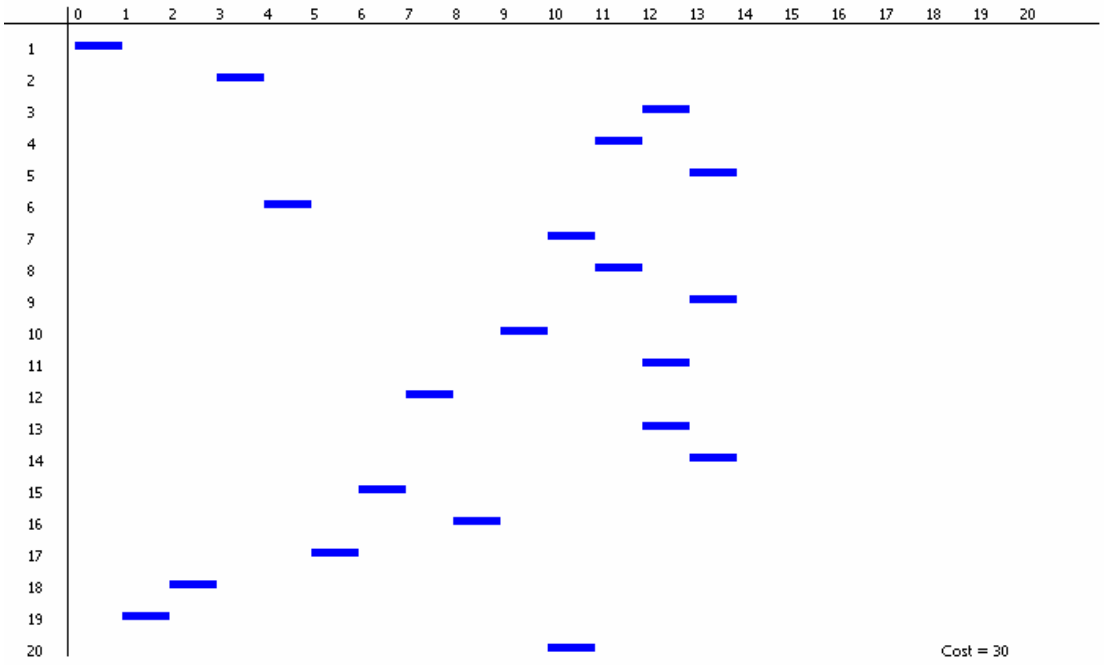


Figure 4.6: Gantt Chart of FRECAST Broadcasting.

4.6 Summary and Conclusion

In this chapter, we propose FRECAST as a tool for single-casting, multicasting and broadcasting in an ad hoc network. The single-casting problem has been modeled after the Floyd-Warshall's all pairs shortest path algorithm, while multicasting and broadcasting are based on the Prim's algorithm for the minimum spanning tree. One advantage here is FRECAST provides a visualization solution for these problems based on the table-driven WRP algorithm.

Chapter 5

SPLAI: FINITE ELEMENT MODEL FOR SENSOR NETWORKS

5.1 Introduction

Recent advances in micro-electromechanical systems (MEMS) have led to the rapid development of micro-sensors. Such sensors are generally equipped with data processing and communication capabilities. Wireless sensor networks are a new class of ad hoc networks that are expected to find increasing deployment in coming years. A sensor network is a collection of tiny disposable and low-power devices. A sensor node is a device that transforms a sensed attribute into something that provides information about its vicinity. Each sensor in the network sends such collected data, usually via radio transmitter, to a command center as a sink either directly or through a data concentration center as a gateway.

Sensor networks have been proposed for a wide variety of application areas. Some examples are such as intrusion detection and tracking, where sensors are deployed along the border of a battlefield to detect classify and track intruding personnel and vehicles. There also a lot of interest in the environmental monitoring, where specialized sensor nodes that are able to detect temperature changes and smoke can be deployed in high-risk of a forest, to give early warning of forest fires. In traffic analysis, traffic sensor network can monitor vehicle traffic on a highway or a congested part of a city. In security, surveillance sensor networks can be used to provide security in an art gallery or shopping mall. At the network layer of a sensor

network, the main aim is to find ways for energy-efficient route setup and reliable relaying of data from the sensor nodes to the sink. So, a special multihop wireless routing protocol between the sensor nodes and the sink are needed to maximize the lifetime of the network. Problems encountered in effective implementation of the sensor networks include coverage area, data routing and power efficient.

In this chapter, we discuss the data routing issues by considering the limited energy supply in sensor networks. The issues involve three phases: Interest Request, Message Transmission and Data Computation. We propose SPLAI, or *Sensor Protocols for 2D-Linear Approximation Information*, which is a dissemination algorithm based on the data-centric approach for modeling the finite element solution to sensor networks. Our routing technique in SPLAI requires finding the spanning tree and transmitting messages to the processing element by finding the shortest paths to optimize the routing. We also consider the self-organizing features of the network in the case when some nodes fail. The network adapt to the changes by updating the current information to the rest of the nodes. The computational model discussed in SPLAI consists of the two-dimensional (2D) linear triangular approximation of the finite element method.

The chapter is organized into four sections. In Section 5.1, we review the application and implication of routing issues in sensor networks. Section 5.2 describes the routing protocol problem that considers the limited energy supply constraint. Several data-centric routing approaches are discussed in Section 5.3. We describe a method for locating the dynamic coordinates of the sensor nodes in the network in Section 5.4. This is followed by some discussion on SPLAI in Section 5.5. We further describe the development of SPLAI involving the dissemination method using Prim's Algorithm and data computation using the linear triangular approximation of the finite element method in Section 5.6. Section 5.7 is the summary and conclusion.

5.2 Problem Formulation

The problem is stated as follows: Given a set of sensors and a few specialized nodes termed as processing elements (PEs) in a sensor network, how do the PEs react to the data provided by the sensors? Two sub-problems are associated here: first is locating the position of all the sensor nodes relative to the PEs in a process called *training*, and second is computing the data supplied by the sensors once training has been completed. The PEs in this context are assumed to have unlimited communication capability for transmitting and receiving messages to/from the sensors, and enough memory and processing capability to perform tasks such as mathematical calculations.

The sensor network in our discussion is modeled as an undirected graph. In this model, a node in the graph represents a sensor while an edge between two nodes represents the communication link between the sensors. The absence of a link between a pair of nodes means the two nodes are not within their transmission range.

Our case study in this chapter involves the development of a finite element model for computing the temperature in a convex hull that is bounded by sensors in a sensor network. This problem involves finding a practical method for disseminating and transmitting messages in sensor networks. We also explore the problem of approximating a value at new location within the network. Disseminating involves the transmission of messages to a group of nodes in the network, which is a problem of broadcast through a spanning tree. Transmitting involves the transmission between a pair of nodes, which is from sensor node to the processing element. A term, approximating is the computation of a message to predict messages at new location within the network. Our contribution consists of a protocol for routing and improves its performances through the development of our model, SPLAI.

5.3 Overview of Sensor Network Routing Protocols

Advances in wireless communications and electronics have fostered the development of relatively in-expensive and low-power wireless sensor nodes. The sensor nodes are extremely small in size and communicate un-tethered in short distances. These small devices are incorporated with sensing, data processing and communicating components, to collect data, monitor equipment, and transmit information, which leverages the idea of sensor networks. Each sensor node in these networks operates autonomously with no central point of control in the network and communicates using infrared devices or radios.

The main goal in conventional wireless networks such as mobile ad-hoc network or MANET is to provide high quality of service and high bandwidth efficiency when mobility exists. In contrast, in a sensor network conservation of energy is considered to be important than the performance of the network. Therefore, the current routing protocols designed for traditional networks cannot be used directly in a sensor network due to the following reasons [21]:

a. Sensor networks are data-centric, where data is requested based on certain attributes.
b. Application specific, where the requirements of the network will change with the application.
c. Aggregated data in the case that adjacent nodes may have similar data.
d. Since the model is data-centric, each sensor is assumed to have non-unique id. Additionally, large number of nodes in the network implies large ids, which might be substantially larger than the actual data being transmitted.

To comply with the above requirements, new routing schemes have been proposed. Routing protocols must select the best path to minimize the total power and to maximize the lifetime of all nodes. Due to such differences, many new algorithms have been proposed in the problem of routing data in sensor networks. Most of these routing mechanisms consider the characteristics of sensor nodes along with the application and architecture requirements [22].

Routing protocols in sensor networks can be classified as data-centric, hierarchical or location-based. Data-centric protocols are query-based and depend on the naming of desired data, which helps in eliminating many redundant transmissions. The sink sends queries to certain regions and waits for data from the sensors located in the selected regions. Since data is being requested through queries, attribute-based naming is necessary to specify the properties of data.

Some established data-centric routing protocols are *Flooding and Gossiping, Sensor protocols for information via negotiation* (SPIN) *and Directed Diffusion*. Flooding and Gossiping are two classical mechanisms to relay data in sensor networks without the need for any routing algorithms and topology maintenance. In flooding, a node wishing to disseminate a piece of data across the network starts by sending a copy of this data to all of its neighbors. Each sensor receiving a data packet broadcasts it to all of its neighbors except the node from which it just received the data. This process continues until the packet arrives at the destination or the maximum number of hops for the packet is reached. Gossiping is a slightly enhanced version of flooding where the receiving node sends the packet to a randomly selected neighbor, which picks another random neighbor to forward the packet to and so on [22]. SPIN considers data negotiation between nodes in order to eliminate redundant data and save energy [25]. Directed Diffusion has been developed and has become a breakthrough in data-centric routing [[26].

A different approach from the hierarchical protocols is achieved by clustering the nodes so that cluster heads can do some aggregation and reduction of data in order to save energy. As mentioned, cluster formation is typically based on the energy reserve of sensors and sensor's proximity to the cluster head. The hierarchical routing approaches for sensor networks include LEACH (Low-Energy Adaptive Clustering Hierarchy), PEGASIS (Power-Efficient GAthering in Sensor Information Systems) and TEEN (Threshold sensitive Energy Efficient sensor Network protocol). Another approach uses the location-based protocols, which utilize the information on the nodes location to relay data to the desired regions rather than the whole network. In most cases, location information is needed in order to calculate the distance between two particular nodes so that energy consumption can be estimated. MECN (Minimum Energy Communication Network), GAF (Geographic

Adaptive Fidelity) and GEAR (Geographic and Energy-Aware Routing) are among the location-based protocols in sensor networks.

5.4 Training the Sensor Nodes

A wireless sensor network is a self-organized network which does not depend on external help in establishing itself. The position of each node in the network is determined through a process called *training*. Basically, training involves a repeated transmission of signals with varying strengths and directions from the PE (sink) to the sensor nodes. One difficulty in locating the nodes is the fact that the Cartesian or the $xy-$coordinate system is not practical for implementation as the exact location of each node may not be traced. It is not wise to assume each node has a position locating device such as the GPS as the sensor network as a whole is self-organized. Therefore, an approximation based on a dynamic coordinate system will be more appropriate and practical in its implementation. Dynamic coordinate, in this context, refers to a coordinate system relative to the position of the sink. Training is said to be successfully completed once all the nodes that lie within the transmission range to the sink have been assigned with the dynamic coordinates.

We assume only one processing element is used in this model. This processing element has enough computing power and memory to train the sensor nodes, and to perform all the necessary computations from the data gathered from the nodes. In addition, each sensor node is assumed to be awake at all time to respond to the signals from PE during the training. This assumption may not describe the real scenario of the sensor nodes which are in sleep most of the time for saving energy. However, this assumption is necessary in the simulation in order to focus to the computational aspect of the model.

Our model is based on the work of Haseebulla [24] which implements the coronas (c) and wedges (w), coordinate system. In this coordinate system, a *corona* is an annular area whose center is at PE. A *wedge* is a sector measured from the $x-$axis in the anti-clockwise direction with its center at PE. The origin of the coordinate system, or the *sink*, is the starting coordinate at (0,0) which lies in the first sector in the innermost circle. Figure 5.1 (left) shows a simple corona-wedge

coordinate system with three coronas and eight wedges with their assigned coordinates.

Training the nodes in the network involves a successive transmission of signals from PE to the nodes. The strength of a signal corresponds directly to the corona number: the weakest indicates Corona 0 while the strongest is Corona $C-1$, where C is the number of coronas in the network. In addition, for training the nodes according to the wedge number, PE transmits its strongest signals successively in angular, anti-clockwise directions starting from the $x-$axis. PE determines the corona and wedge numbers of a node only when a transmitted signal receives a respond from the node. If no respond is received then the node is assumed to be outside of the transmission range and the node is said to be untrained.

The first phase of the training involves the coronas only. For C coronas, c_i for $i = 0,1,2,...,C-1$, the training starts from the innermost corona, c_0. The PE transmits its weakest signal in the circular region identified as Corona 0. Upon receiving the beacon, the nodes belonging to this corona immediately respond and they are instantly identified to belong to this corona. PE repeats by increasing the transmission strength to the second level, and the nodes that respond to the signal are those that may belong to Corona 0 or 1. However, the nodes in Corona 0 have been identified earlier and, therefore, their complements are assigned to Corona 1. The same process is repeated until the last corona, c_{C-1}, which corresponds to the maximum signal strength. It is obvious from this approach that only nodes that are within the transmission range will respond to the signals from PE. Those that don't respond are considered the outliers, and they are assumed to be outside of the network.

The next phase of training involves the wedges. Training is achieved by having the PE transmitting its maximum signals in directional positions according some predetermined angles measured from Wedge 0. In general, a network with W equal-width wedges requires a directional transmission of $2\pi/W$ as its angle. For example, the directional angle for a network with eight wedges is $2\pi/8$, as illustrated in Figure 5.1 (left).

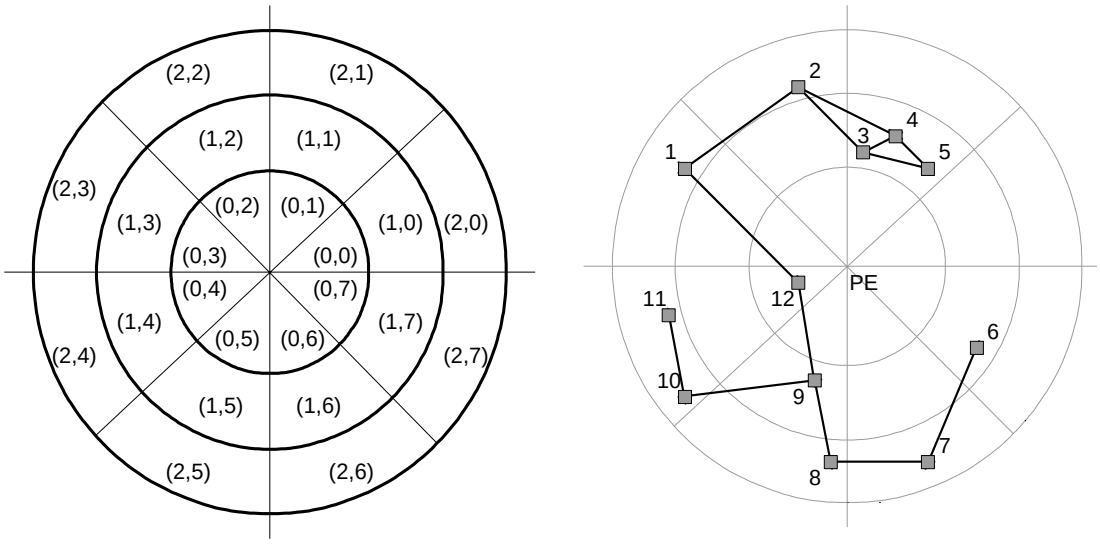


Figure 5.1. Dynamic coordinates and the location of the sensor nodes.

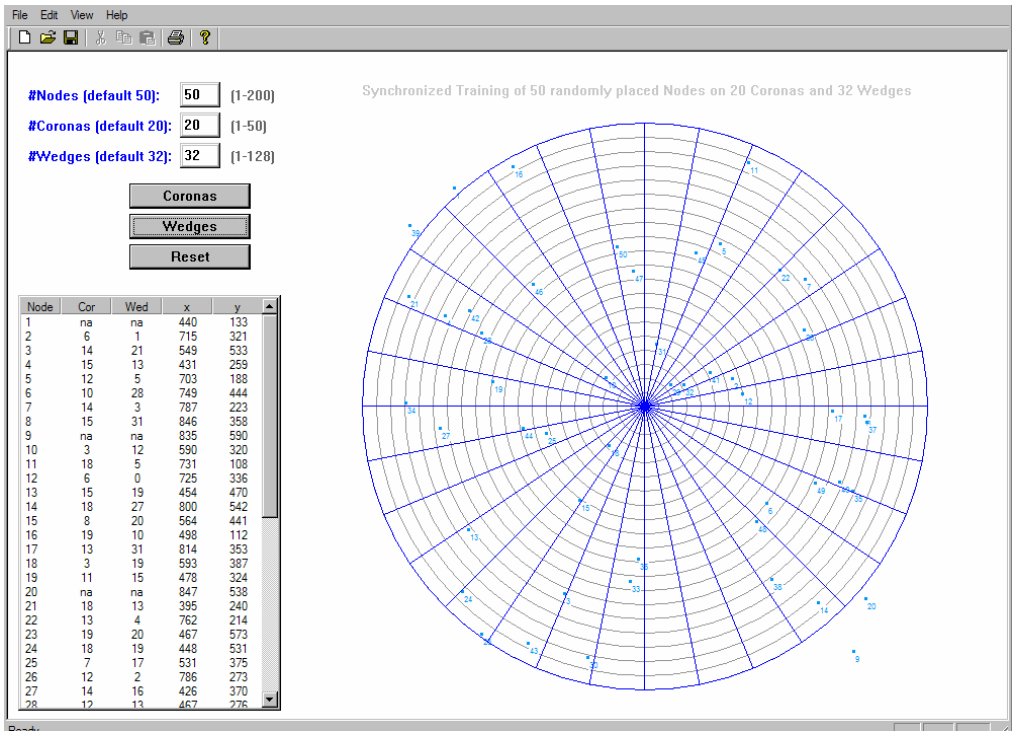
Figure 5.1 (right) shows 12 nodes scattered randomly within the transmission range of PE. The figure shows the result from training using three coronas and eight wedges. Each node has a limited transmission range as illustrated as edges in the graph. Note that the dynamic coordinates may not be unique as a coordinate location may not be attached to one node only. As shown in the figure, node 1 is located at (2,3) while nodes 3, 4 and 5 share the same coordinate, $(1,1)$.

Assuming the origin of the Cartesian coordinates at the sink, the position at (c, w) can be transformed to the Cartesian coordinates (x, y) using the following relationships:

$$x = c\cos\frac{2\pi}{W}w, \tag{5.1a}$$

$$y = c\sin\frac{2\pi}{W}w. \tag{5.1b}$$

Conversion to real coordinates using Equations (5.1a) and (5.1b) is very useful especially in cases where the real location of the sink is known using a measuring device such as the GPS.



Node	Cor	Wed	x	y
1	na	na	440	133
2	6	1	715	321
3	14	21	549	533
4	15	13	431	259
5	12	5	703	188
6	10	28	749	444
7	14	3	787	223
8	15	31	846	358
9	na	na	835	590
10	3	12	590	320
11	18	5	731	108
12	6	0	725	336
13	15	19	454	470
14	18	27	800	542
15	8	20	564	441
16	19	10	498	112
17	13	31	814	353
18	3	19	593	387
19	11	15	478	324
20	na	na	847	538
21	18	13	395	240
22	13	4	762	214
23	19	20	467	573
24	18	19	448	531
25	7	17	531	375
26	12	2	786	273
27	14	16	426	370
28	12	13	467	276

Figure 5.2. Training of 50 nodes using 20 coronas and 32 wedges in SPLAI.

The training of nodes into the dynamic coordinates is implemented visually in SPLAI. Figure 5.2 shows a training simulation of 50 sensor nodes that are scattered randomly, which are grouped into 20 coronas and 32 wedges in SPLAI. Their location in (c, w) are determined through a series of repeated transmissions from the sink (PE), and these values are converted into their corresponding (x, y) values using Equations (5.1a) and (5.1b).

5.5 Routing Model

Once training is completed, the next step is to form a routing mechanism to allow communication among the sensor nodes. This step is important as a value sensed by a node needs to be passed to PE to be used in the computational stage. PE requires all data read from the nodes in the network before performing the necessary computation. However, it is not possible for a node to transmit its value directly to PE as it has a short and limited transmission capability. In order to transmit its value to PE, the node will have to form a routing path through other nodes until it reaches PE.

Basically, routing between the nodes and PE in the network involves finding the shortest paths between pairs of points in the graph. This approach requires finding the shortest path from every sensor node to PE, which may take a considerable computing time and overhead. Another practical approach is to compute the minimum spanning tree of the graph with PE as its root. This approach has the advantage of eliminating redundant paths that may result from the shortest paths.

Two modes of routing for allowing communication in the sensor network are discussed in SPLAI. First is the *Interest Request Phase* which is a message transmission from the sink (PE) to the sensor nodes using a routing path based on the minimum spanning tree of the network. This path is necessary to enable PE to send request for values to the nodes. The other mode is the *Message Transmission Phase* which is a message transmission on the sensed values from the nodes to PE.

5.5.1 Interest Request Phase using the Minimum Spanning Tree

In SPLAI, we adopt the second approach of the training, that is, using the minimum spanning tree for routing from PE to the nodes. The training requires computing the minimum spanning tree of the graph with its root at PE using the Prim's algorithm. The task of finding the minimum spanning tree is performed by PE once the information about the network is known. The spanning tree is important in this context as it represents the interest request phase for PE as the processing unit needs to know the optimum path for communicating with all the sensor nodes in the network.

The main idea behind SPLAI is outlined as follows: PE disseminates an interest query packet for the values to each node in the network using the minimum spanning tree path. Upon receiving the interest packets, each node in the network starts periodically sending its temperature data or measurements back to the processing element using the same path. Since many or all of the nodes will be sending their data at the same time, it would be more efficient to combine these readings into a single packet. As data flows, it can be aggregated along the way. PE stores the data and performs the necessary computation to evaluate the temperature at any point based on the finite element technique.

SPLAI includes tasks such as training the nodes, routing for data communication and performing computation on the sensed data. The model consists of a data-centric algorithm as it involves the dissemination of data from the sensor nodes, and its transmission to PE. Upon receiving the required data, PE stores them in its memory before applying the necessary computational steps.

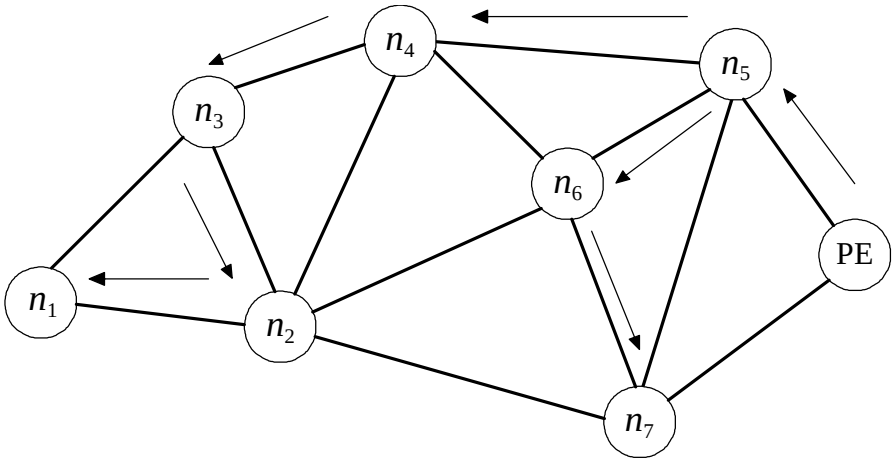


Figure 5.3. A route initiated by PE in Interest Request Phase using MST

Figure 5.3 shows a sensor network consisting of a processing element (PE) and seven randomly distributed sensor nodes, n_i for $i = 1,2,...,7$, scattered in a two-dimensional area Q. Each sensor node, n_i, has the corona-wedge coordinates of (c_i, w_i) which is transformed to (x_i, y_i) using Equations (5.1a) and (5.1b). Each sensor has a sensing range of R_i, that is, it can sense a metric such as temperature (or pressure and sound in a similar situation), that is within a distance of R_i from n_i. The figure also shows message routing path from PE to the nodes on the minimum spanning tree (shown as arrows).

The sensor network behaves like an *ad hoc network* where the nodes can both receive and transmit messages. Each node in the network has a transmitter and a receiver to perform these functions to act as a router. To facilitate message transmission, a routing protocol called DSDV, or *Destination Sequenced Distance Vector*, is adopted. In this protocol, a node updates its routing and neighbor information proactively whenever new information is received.

In our work, G represents a network of sensor nodes which is assumed to be connected and undirected in this model. The dissemination method of an interest query packet in the network is a graph-broadcasting model of sensor nodes. A crucial problem in this case is to find the minimum spanning tree of the graph with the root as the sender. This problem is solved using the Prim's algorithm, which is represented in its visual form in [28].

The algorithm begins with an empty graph, T. The strategy in the Prim's algorithm consists of building a tree in T one node at a time starting from any node in the graph. From the starting node, a link with the minimum weight to its neighboring node is added to T. Next, the shortest link emanating from the two nodes becomes the minimum link and added into T. This added link must not form a circuit in T, otherwise the move is rejected. The step repeats from the second node to the third, and subsequently until all the nodes have been included.

The following steps summarize the Prim's algorithm for finding the minimum spanning tree of a graph:

```
Read the information on the weighted graph G;
Create an empty graph, T;
Choose the first node, and add into T;
do until T becomes a spanning tree of G
    Choose the minimum link, originating from the node;
    if the new link does not have a circuit
        Add the link and its new node into T;
    else
        Reject the move;
```

5.5.2 Message Transmission Phase

The next step upon receiving the interest query packet of temperature value from the processing element is the message transmission phase. Each sensor node aggregates their sensed values to PE using the route determined from the DSDV routing protocol [27]. DSDV provides the shortest paths of all pairs of nodes in the network. In DSDV, each node has a table for updating its routing information with its adjacent

nodes, and this allows the simultaneous updates of the values all the way to the sink (PE). PE receives this vital information, and use these values for performing the subsequent computational steps.

Table 5.1. Routing table structure for n_2.

Destination	Next	h_n	s_n
n_1	1	1	n_1-160
n_2	2	0	n_2-200
n_3	3	1	n_3-666
n_4	4	1	n_4-76
n_5	6	2	n_5-888
n_6	6	1	n_6-550
n_7	6	2	n_7-444
PE	6	3	PE-250

Table 5.1 shows the routing information of the node n_2 in the graph from Figure 5.3 using SPLAI. In a basic distance vector routing protocol, every host maintains a routing table containing the distances from it to all possible available destinations. In the table, *destination* refers to the address of the destination node, while *next* is the next hop along the path of the current node to get to the destination. h_2 is the total number of hops required to get to the destination. The sequence number originated from destination is represented by s_n.

In DSDV, the number of hops is the metric for computing the path from the source to its destination. DSDV extends the basic Bellman-Ford and the Floyd-Warshall algorithms by attaching a sequence number to each route table entry. A route with a higher recent sequence numbers is always preferred as the basis for making the forwarding decision. In this case, the nodes can distinguish the stale routes from the new ones, and this avoids the formation of routing loops in the network. The message transmission phase is based on the DSDV routing protocol using shortest paths outlined by the Floyd-Warshall algorithm. The algorithm, as described in its visual form in [28], is summarized as follows:

a. Each sensor node is referred to its routing table entries.
b. A sensor node sends its temperature data to the *next* sensor node that is its immediate neighbor in the network.
c. The receiving sensor node then sends the value to its *next* sensor node until it reaches the PE.
d. If broken links occur in the network, the nearest sensor node will advertise a route table by assigning a metric of ∞ to its table. In this case, every sensor node in the network updates its table. Any new information will not be sent to the node in the next transmission.
e. The sequence number is then increased by 1 for every broken link detected.

The source in our model is PE, which then disseminates interest packets to other nodes in the network. PE stores this value and shares the temperature data with other nodes.

5.6 Computational Model using Finite-Element Method

In our model, PE is capable of performing numerical computations on the data provided by the sensor nodes. This step is performed once training has been completed, and the network has been successfully organized. The computational model is useful in many numerical applications such as in approximating the temperature of any point inside the network using the finite element method. We discuss the problem of computing the temperature of the points inside the network using the linear triangular approximation (LTA) technique in finite element method [23]. The linear triangular approximation method computes the temperature in any point inside a triangle by taking the temperatures at the vertices of the triangle as the finite elements.

We describe our computational model using an example of a network with one processing element and seven sensor nodes, as shown in Figure 5.3. Figure 5.4 shows the triangular elements formed from the graph in Figure 5.3. There are seven areas in the bounded region of the network, marked as e_i for $i = 1,2,...,7$.

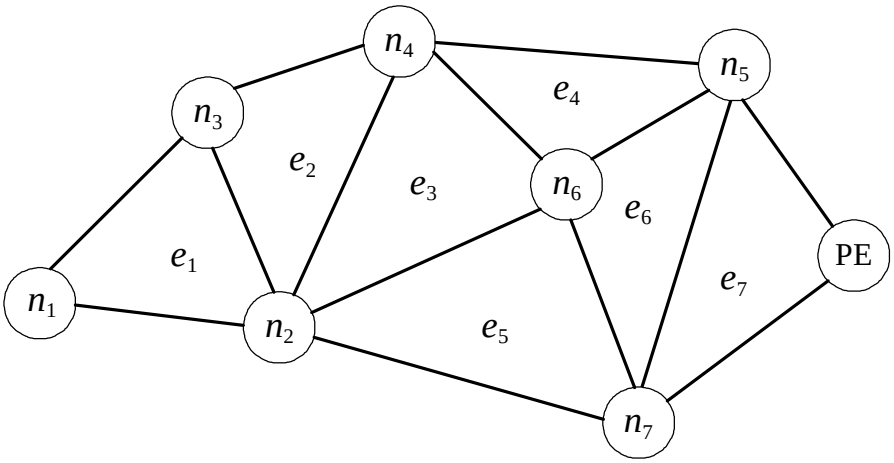


Figure 5.4. Sensor network with seven nodes and one processing element.

One of the main constraints in a sensor network is its limited energy supply. The network needs to limit its activities that consume energy. To achieve this objective, only one PE is used for the computations, while the sensor networks supply the necessary data. Upon user query, PE performs the computation to approximate the value at other location, which is out of distance of r_i from s_i using the values at three other node or locations.

Figure 5.4 shows the finite element model formed from the sensor nodes which are scattered in the plane. Each sensor node reads the temperature at its location then transmits this value to PE through the shortest path route. The temperature field within an element is given by

$$\theta = \lambda_1\theta_1 + \lambda_2\theta_2 + \lambda_3\theta_3, \text{ or } \theta = \lambda\theta^{(e)}, \tag{5.2}$$

where θ_i is a temperature and λ_i is the shape function at n_i. This is conveniently represented in terms of the pair α, β as follows:

$$\lambda = [\alpha, \beta, 1-\alpha-\beta], \tag{5.3}$$

$$\lambda_1 + \lambda_2 + \lambda_3 = 1. \tag{5.4}$$

The shape function describes how the temperature varies in each element. Using the isoparametric representation, the displacements inside the element are now

written using the shape functions and the nodal values of the unknown displacements field. We have

$$u = \lambda_1 q_1 + \lambda_2 q_3 + \lambda_3 q_5, \tag{5.5a}$$

$$v = \lambda_1 q_2 + \theta_2 q_4 + \theta_3 q_6, \tag{5.5b}$$

where q_1, q_2, q_3, q_4, q_5 *and* q_6 show two directions for each node in the triangle, and u and v are the nodal values of the unknown displacements. The above equation can be written as

$$x = \lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3, \tag{5.6a}$$

$$y = \lambda_1 y_1 + \lambda_2 y_2 + \lambda_3 y_3. \tag{5.6b}$$

Substituting the value of λ from (3.3) into Equations (5.6a) and (5.6b) produce

$$x = (x_1 - x_3)\alpha + (x_2 - x_3)\beta + x_3, \tag{5.7a}$$

$$y = (y_1 - y_3)\alpha + (y_2 - y_3)\beta + y_3, \tag{5.7b}$$

where $(x_1, y_1), (x_2, y_2)$ and (x_3, y_3) are the coordinates of s_1, s_2 and s_3, respectively. Using the notation, $x_{ij} = x_i - x_j$ and $y_{ij} = y_i - y_j$, Equation (5.5) becomes

$$x = x_{13}\alpha + x_{23}\beta + x_3, \tag{5.8a}$$

$$y = y_{13}\alpha + y_{23}\beta + y_3. \tag{5.8b}$$

The values of λ_1, λ_2 and λ_3 are obtained by solving Equations (5.8a) and (5.8b). It follows that the temperature at location (x, y) in each element in the network can be calculated by substituting these values into Equation (5.2).

Table 5.2. Element Connectivity.

Element (e)	(x_1, y_1)	(x_2, y_2)	(x_3, y_3)
1	2	3	1
2	2	4	3
3	6	4	2
4	6	5	4
5	7	6	2
6	7	5	6
7	8	5	7

Table 5.2 shows the connectivity of the elements in the network. As depicted in the table, most standard finite element codes use the convention of going around the element in a counterclockwise direction to avoid calculating a negative area.

Table 5.3. Information received by PE based on DSDV.

Node	Location, (x_i, y_i)	Temp., θ	Element, e
n_1	(2,4)	100	e_1=(4,8)
n_2	(6,8)	80	e_2 =(6,12)
n_3	(4,12)	120	e_3=(10,15.5)
n_4	(10,16)	200	e_4 =(14,10)
n_5	(16,14)	140	e_5 =(12,6)
n_6	(14,9)	60	e_6 =(15,10.5)
n_7	(16,4)	35	e_7 =(17,6)
PE	(22,8)	23	

Table 5.3 is an illustration of the network in Figure 5.3, which shows the temperature, location and element of each node. The temperature at n_5 is computed by referring to the information given in Table 5.1 and Table 5.2. By substituting the readings from these tables into Equations (5.9a) and (5.9b) at this node we obtain the following simultaneous equations:

$$10\alpha + 8\beta + 6 = 12$$
$$-4\alpha + \beta + 8 = 6$$

Solving the above equations, we get $\alpha = 0.5238$ and $\beta = 0.09525$. It follows that Equations (5.4) and (5.5) produce $\lambda_1 = 0.5238$, $\lambda_2 = 0.09525$ and $\lambda_3 = 0.38095$. Finally, we obtain the temperature of n_5, as follows:

$$\theta = \lambda\theta^{(e)} = \begin{bmatrix} 0.5238 & 0.09525 & 0.38095 \end{bmatrix} \begin{bmatrix} 35 \\ 60 \\ 80 \end{bmatrix} = 54.525.$$

Table 5.4. Temperature approximation at the finite elements.

e_i	1	2	3	4	5	6	7
θ_i	99.9	130	174	77.648	55.524	89.5	37.416

Applying the above steps to all other elements in the network we get the results as shown in Table 5.4. This phase provides instant information including the case whereby one or more sensors in the network are weak or not functioning properly.

In general, the routing and computation protocols in SPLAI for a sensor network using one processing elements can be summarized as follows:

a. The processing element, PE broadcasts an interest query packet to get the temperature data in the region by following the path specified by the minimum spanning tree.
b. When the packets reach the sensor nodes, they aggregate their temperatures to their nearest neighbors based on their routing table entries until the message reaches PE.
c. PE is responsible for receiving, storing, processing and sharing the temperature data with other networks. In the processing, PE can perform the computation to approximate the temperature of any location even if it is out of range from s_i.

5.7 SPLAI Simulation Model

SPLAI was developed using the C++ programming language running on the Windows environment. Figure 5.5 shows the interface of our SPLAI model network for massively deployed sensor nodes. The sensor nodes are scattered randomly in a rectangular area with a transmission range of 250 units. We describe this model using a case study using 19 sensor nodes and one PE. Figure 5.5 shows this scenario where node 20 is the processing element, or the sink, while the other nodes numbered from 1 to 19 are the sensor nodes. SPLAI also displays the link table and temperature record table.

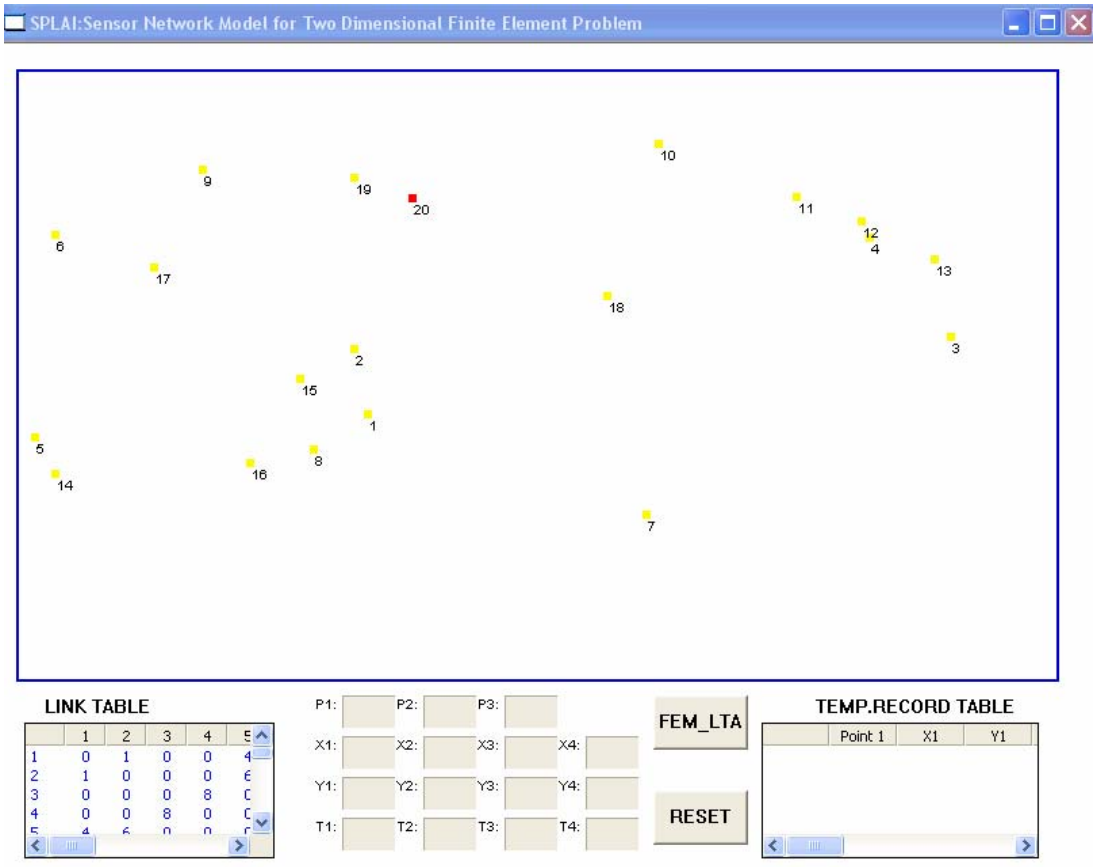


Figure 5.5. Illustrating SPLAI model interface.

Table 5.5 shows the link table between the sensor nodes based on their distance apart from the case in Figure 5.5. The values shown in the table provide a vital information about routing on the interest request phase from PE to the nodes and the message transmission phase from the nodes to PE.

Table 5.5. Link Table for 19 nodes and the PE.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	1	0	0	4	0	4	8	8	0	0	0	0	1	6	10	2	8	4	3
2	1	0	0	0	6	5	2	4	8	0	0	0	0	4	5	2	3	1	8	3
3	0	0	0	8	0	0	0	0	0	0	7	7	4	0	0	0	0	0	0	0
4	0	0	8	0	0	0	0	0	0	5	9	9	5	0	0	0	0	6	0	0
5	4	6	0	0	0	8	0	10	9	0	0	0	0	5	1	5	9	0	0	0
6	0	5	0	0	8	0	0	0	2	1	0	0	0	1	3	5	9	0	6	0
7	4	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0
8	8	4	0	0	10	2	0	0	4	0	0	0	0	7	6	6	6	6	4	7
9	8	8	0	0	9	1	0	4	0	0	0	0	0	10	3	5	7	0	2	2
10	0	0	0	5	0	0	0	0	0	0	1	4	6	0	0	0	0	9	3	7
11	0	0	7	9	0	0	0	0	0	1	0	6	3	0	0	0	0	5	0	0
12	0	0	7	9	0	0	0	0	0	4	6	0	7	0	0	0	0	4	0	0
13	0	0	4	5	0	0	0	0	0	6	3	7	0	0	0	0	0	7	0	0
14	1	4	0	0	5	1	0	7	10	0	0	0	0	0	5	9	8	0	0	0
15	6	5	0	0	1	3	0	6	3	0	0	0	0	5	0	3	10	6	5	9
16	10	2	0	0	5	5	0	6	5	0	0	0	0	9	3	0	7	0	3	1
17	2	3	0	0	9	9	0	6	7	0	0	0	0	8	10	7	0	0	1	2
18	8	1	0	6	0	0	9	6	0	9	5	4	7	0	6	0	0	0	1	5
19	4	8	0	0	0	6	0	4	2	3	0	0	0	0	5	3	1	1	0	5
20	3	3	0	0	0	0	0	7	2	7	0	0	0	0	9	1	2	5	5	0

Figure 5.6 shows the minimum spanning tree in the Interest Request Phase initiated by the PE to perform the path in the network before the sensors start sending and receiving the data. The tree is constructed using the Prim's algorithm. This phase provides an important step in having PE to present its request for values to the sensor nodes.

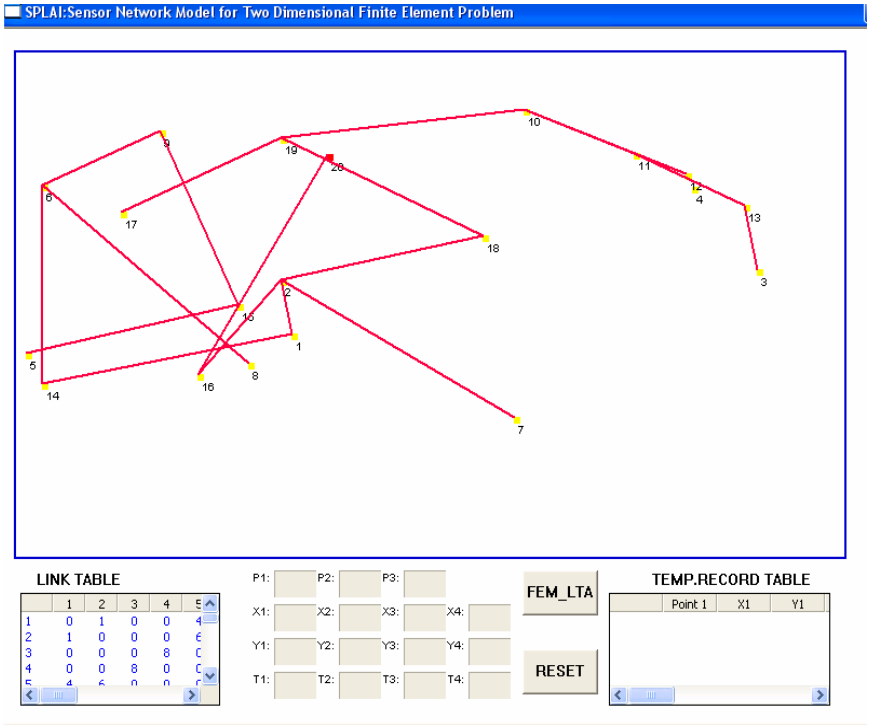


Figure 5.6. Illustrating the interest request phase.

Once the values are received from the sensor nodes, PE becomes ready to perform the computational steps for computing the temperatures at the elements. Figure 5.7 shows the data computation phase in SPLAI for approximating the temperatures using the linear triangular approximation of the finite element method. The location of the nodes is indicated as bold number 1, 2, 3, 4 and * in the simulation.

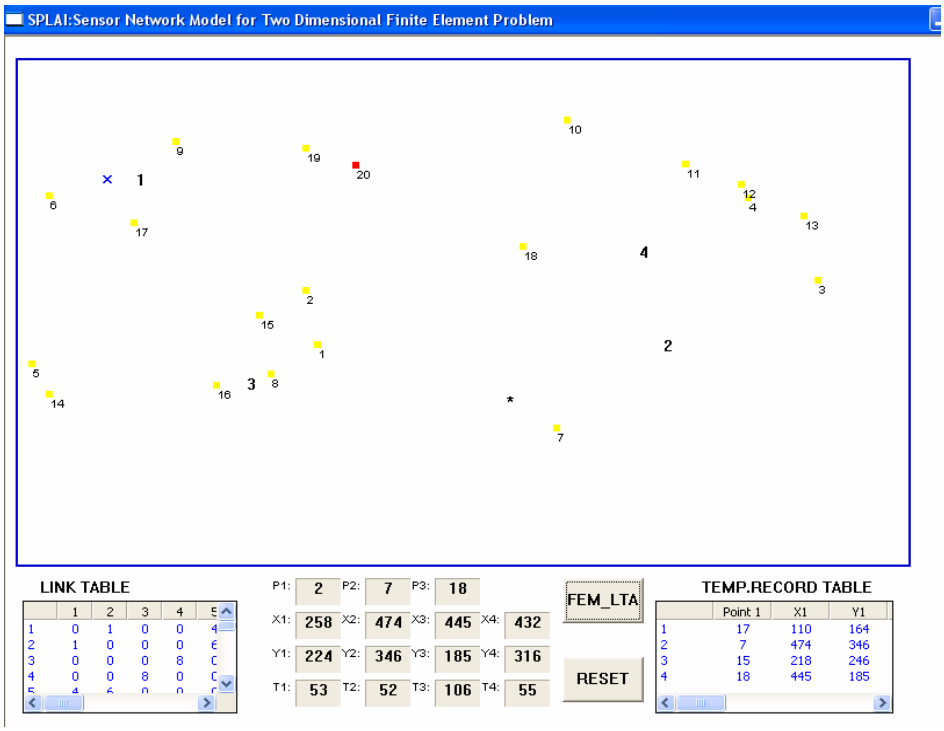


Figure 5.7. Illustrating the data computation phase.

Table 5.6 shows the results in the form of approximated temperature values of the 19 elements in the network using the finite element method.

Table 5.6. Indicating the result shown in the TEMP.RECORD TABLE .

	P1	X1	Y1	T1	P2	X2	Y2	T2	P3	X3	Y3	X4	Y4	*T4
1	17	110	164	83	9	146	92	33	6	37	140	113	119	43
2	7	474	346	52	3	699	215	119	18	445	185	567	266	89
3	15	218	246	89	16	181	308	37	8	228	298	208	300	54
4	18	445	185	106	13	687	158	71	10	483	73	546	183	98

5.8 Summary and Conclusion

In this chapter, we propose SPLAI as a tool for disseminating, transmitting and computing in a sensor network. Our protocol is based on the data-centric approach involving the creation of dynamic coordinates for each node before its real location is determined. The disseminating problem has been modeled as a spanning tree based on Prim's Algorithm, while the transmission is a table-driven shortest path protocol based on the DSDV algorithm. One obvious advantage from our approach is SPLAI considers the problem of limited energy supply among the sensor nodes through the computation phase of the finite element method using the linear triangular approximation (LTA) approach.

The computational model in SPLAI is based on one processing element. Obviously, it is possible to extend the work by incorporating more than one processing element. Two or more processing elements can be used to enhance the training and routing methods as well as transforming the computational step into a parallel model. The use of multiple processing elements has the advantage of speeding up training and routing which contributes in the energy constraint scenario of the sensor network. Energy saving is a crucial factor in a sensor network which has to be considered seriously before the network can be deployed. This requirement is an open problem that will definitely attract further research into this area.

References

[1] B.S.Ting, E.S.Kuh and L.Shirakawa, "The multilayer routing problem: algorithms and necessary and sufficient conditions for the single row, single layer case", *IEEE Trans. Circuits Systems*, vol. 23, pp. 768-778, 1976.

[2] R.Raghavan and S.Sahni, "Single-row routing", *IEEE Trans. Computers*, vol.32, no.3, pp.209-220, 1983.

[3] E.S.Kuh, T.Kashiwabara and T.Fujisawa, "On optimum single-row routing", *IEEE Trans. Circuits and Systems*, vol.26, no.6, pp.361-368, 1979.

[4] J.S.Deogun and N.A.Sherwani, "A decomposition scheme for single-row routing problems", *Technical Report Series #68*, Dept of Computer Science, Univ. of Nebraska, 1988.

[5] D.H.Du and L.H.Liu, "Heuristic algorithms for single row routing", *IEEE Trans. Computers*, vol.36 no.3, pp.312-319, 1987.

[6] S.Salleh, B.Sanugi, H.Jamaluddin, S.Olariu and A.Y.Zomaya, "Enhanced simulated annealing technique for the single-row routing problem", *Journal of Supercomputing*, vol. 21 no.3, pp.285-302, March 2002.

[7] S.Kirkpatrick, C.D.Gelatt and M.P.Vecchi, "Optimization by simulated annealing", *Science*, vol.220, no.4598, pp.671-678, 1983.

[8] R.A.Rutenbar, "Simulated annealing algorithms: an overview", *IEEE Circuits and Devices Magazine*, pp. 19-26, January 1989.

[9] R.Mathar and J.Mattfeldt, "Channel assignment in cellular radio networks", *IEEE Trans. Vehicular Technology*, vol.42 no.4, 1993.

[10] S.Salleh, S.Olariu, B.Sanugi and M.I.A.Aziz, "Single-row transformation of complete graphs", *Journal of Supercomputing*, 31(3): 265-279, 2005.

[11] M.Pioro and D.Medhi, "Routing, flow and capacity design communication and computer networks", Morgan-Kaufmann, 2004.

[12] J.J.Hopfield and D.W.Tank, "Neural computation of decisions in optimization problems", *Biological Cybernetics*. 1985. 52: 141–152.

[13] G.Galán-Marín and J.Muñoz-Pérez, "Design and analysis of maximum Hopfield networks", *IEEE Trans. on Neural Networks.* 2001. 12(2): 329-339.

[14] Y.Takefuji, "Neural network parallel computing", Kluwer Academic Publishers, Boston, 1992.

[15] Banard Carré (1979), *"Graphs and Networks"*, Clarendon Press, Oxford. England.

[16] Chartrand and Oellermann (1993). *"Applied and Algorithmic Graph Theory"*, McGraw hill, Inc., USA.

[17] Elizabeth M. Royer, Chai-Keong Toh (1999), *"A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks"*, IEEE Personal Communications, Vol. 6, No. 2, pp. 46-55.

[18] McHugh, James A. (1990), *"Algorithmic Graph theory"*, Prentice hall, Inc., USA.

[19] S. Murthy and J. J. Garcia-Luna-Aceves (1996), *"An Efficient Routing Protocol for Routing in Mobile Communication Networks"*, Oct. 1996, pp. 183–197.

[20] Vipin Kumar, Ananth Grama, Anshul Gupta, and George Karypis (1994), *"Introduction to Parallel Computing, Design and Analysis of Algorithms"*, The Benjamin Publishing Company, Inc.

[21] Agrawal, D.P and Zeng, O.A., Introduction to Wireless and Mobile Systems. Brooks/Cole-Thomson Learning, USA, 2003.

[22] Akkaya, K. and Younis, M., A Survey on routing protocols for wireless sensor networks. Ad Hoc Networks Elsevier Computer Science Publications, 2003.

[23] Chandrupatla,T.R. and Belegundu, A.D., Introduction to Finite Elements in Engineering,Third Edition. Prentice Hall, 2002.

[24] Haseebulla Khan, Localization in wireless networks, MSc. Thesis, Department of Computer Science, Old Dominion University, Oktober 2005.

[25] Heinzelman, W., Kulik, J., and Balakrishnan, H., Adaptive Protocols for Information in wireless Sensor networks. Fifth ACM/IEEE mobicom Conference, Seattle, WA, 1999.

[26] Intanagonwiwat, C., Directed Diffusion: A Scalable and robust communication paradigm for sensor networks, ACM Mobicom, 2000.

[27] Perkins, C.E., Bhagwat, B., Highly dynamic destination sequenced distance vector routing(DSDV) for mobile computers, Proceedings of SIGCOMM'94, pp.234-244, 1994.

[28] S.Salleh, A.Zomaya, S.Olariu and B.Sanugi, Numerical simulations and case studies using Visual C++. Net, Wiley-Interscience, New Jersey, 2005.

APPENDIX A

LIST OF PUBLICATIONS

LIST OF PUBLICATIONS

BOOK

1. **Shaharuddin Salleh**, Albert Zomaya, Stephan Olariu and Bahrom Sanugi, "Numerical simulations and case studies using Visual C++.net", ISBN: 0471694614, Wiley-Interscience, Hoboken, NJ, USA. June 2005. (copies of this book are available at many established libraries including Stanford University, Cambridge University, UC Berkeley, Monash University and NASA-Langley).

   Refer to http://as.wiley.com/WileyCDA/WileyTitle/productCd-0471694614.html. This book won the Best Science and Engineering Book Award for 2005 at the UTM Awards and Recognition Day, 1 August 2006.

JOURNAL PAPERS

2. **Shaharuddin Salleh**, Stephan Olariu, Bahrom Sanugi and Mohd Ismail Abdul Aziz, "Single-row transformation of complete graphs", *Journal of Supercomputing,* Springer Science, Amsterdam, vol.31 no.3, pp.249-263, March 2005.
3. **Shaharuddin Salleh**, Albert Zomaya, Stephan Olariu, Kiew Leh Yieng and Nur Arina B. Aziz, "Single-row mapping and transformation of connected graphs", *Journal of Supercomputing*, Springer-Science, (in press, will appear in 2007).
4. Sakhinah Abu Bakar, **Shaharuddin Salleh** and Muhammad Hisyam Lee "FRECAST: Message transmission model for dense graphs in ad hoc networks", *Int. Journal of Simulation and Process Modeling*, Inderscience Publishers, Geneva, vol.2 no.1, pp.4-11, January 2006.
5. Ruzana Ishak, **Shaharuddin Salleh**, Stephan Olariu and Mohd Ismail Abd Aziz, "SPLAI: Finite element computing model for wireless networks", *Journal of Mobile Information Systems*, IOS Press, Amsterdam, vol.2 no.1, pp. 77-92, 2006.
6. Qingwen Xu, Ruzana Ishak, Stephan Olariu and **Shaharuddin Salleh**, "On asynchronous training in sensor networks", *Journal of Mobile Multimedia*, Rinton Press, USA (in press, will appear in 2006).
7. Muhammad Hisyam Lee, **Shaharuddin Salleh**, Bahrom Sanugi and Stephan Olariu, "On the probability of route existence in mobile wireless networks", Journal of Simulations and Process Modeling, Inderscience Publishers (Geneva), (in press, will appear in 2007).

PROCEEDINGS PAPERS

8. **Shaharuddin Salleh**, Bahrom Sanugi and Mohd Ismail Abdul Aziz, "Dynamic Single-row routing for parallel computing systems", Proc. *Japan-Malaysia Seminar on Artificial Intelligence Applications in Industry,* Kuala Lumpur, 24-25 May 2003.
9. **Shaharuddin Salleh**, Bahrom Sanugi and Mohd Ismail Abdul Aziz, "Linear transformation of scattered nodes", Proc. Simposium Kebangsaan Sains Matematik ke 12, Universiti Islam Antarabangsa, Gombak, Selangor, 23-24 December, 2004.

APPENDIX B

LIST OF SOFTWARES DEVELOPED

LIST OF SOFTWARES DEVELOPED

1. ESSR v2.0 (described in Chapters 2 and 3).

2. AdCliq (described in Chapter 3).

3. FRECAST (described in Chapter 4).

4. SPLAI (described in Chapter 5).

APPENDIX C

RESEARCH GROUP

RESEARCH GROUP

1. Project Leader – Assoc. Prof. Dr. Shaharuddin Salleh
2. Researcher – Pn. Ruzana Ishak (PhD student)
3. Researcher – Pn. Sakhinah Abu Bakar (MSc. student)
4. Researcher – Pn. Syarifah Zyurina Nordin (MSc. student)
5. Researcher – Kew Leh Yieng (MSc. student)
6. Researcher – Norazaliza Mohd Jamil (BSc student)

Other Collaborators

1. Prof. Dr. Stephan Olariu, Old Dominion University, USA.
2. Prof. Dr. Albert Y. Zomaya, University of Sydney, Australia.
3. Prof. Dr. Bahrom Sanugi, Dept. of Mathematics, UTM.
4. Assoc. Prof. Dr. Mohd. Ismail Abdul Aziz, Dept. of Mathematics, UTM.

APPENDIX D

FINANCIAL EXPENDITURE

FINANCIAL EXPENDITURE

Total Amount Allocated: RM23,000

Expenses:

1.	Compaq Presario X1000 Notebook	RM7,000
2.	Research Exhibitions	RM1,000
3.	Symposium Kebangsaan Sains Matematik, KL, 2004	RM 3,000
4.	Allowances for Research Assistants	
	- Sakhinah Abu Bakar	RM3,600
	- Kiew Leh Yiend	RM3,600
	- Syarifah Zyurina Nordin	RM3,600
	- Norazaliza Mohd Jamil	RM 800