

# DECENTRALIZED AND ADAPTIVE SENSOR DATA ROUTING

A Thesis

Submitted to the Graduate Faculty of the  
Louisiana State University and  
Agricultural and Mechanical College  
in partial fulfillment of the  
requirements for the degree of  
Master of Science in Systems Science

in

The Department of Computer Science

by  
Mengxia Zhu  
B.S., Zhejiang University, 1996  
December, 2002

## Acknowledgements

I want to begin by thanking my husband, Qishi, for his persistent understanding and support. In addition to being a life partner, you also helped me with my research problems when I was puzzled. You are a truly gorgeous companion with whom I am privileged to share my life.

I wish to express my gratitude to my parents and younger brother for providing both the means and encouragement for me to pursue the cutting edge education in US.

Special thanks to Dr. S.S.Iyengar, my major professor at Louisiana State University and the Robotics Research Laboratory without whose assistance my research achievement during the last two years would have been significantly reduced.

Also I would like to express my most sincere appreciation to Richard Brooks in Applied Research Laboratory at Penn State, he has been directly guiding my research since I came to RRL. Thank you for introducing to me those brilliant ideas proposed in this thesis. Your unique and profound insight into problems deeply impressed me. And the weekly telephone conferences with you have been keeping me on the right track during the past two years. I can say without exaggeration that this thesis wouldn't be possible without your support.

I would like to thank Dr. Donald H. Kraft and Dr. Rajgopal Kannan of the Computer Science Department for your effort to ensure the quality of my thesis.

I also wish to thank the friendly office staff of the Computer Science Department at LSU, whose help with many issues is much appreciated.

# Table of Contents

Acknowledgments .....	ii
List of Tables .....	v
List of Figures .....	vi
Abstract .....	ix
Chapter 1. Introduction .....	1
Chapter 2. Overview of Current Sensor Data Routing Protocols .....	3
2.1 Proactive Sensor Data Routing .....	3
2.1.1 Destination-Sequence Distance-Vector Routing .....	3
2.1.2 Cluster Head Gateway Switch Routing .....	4
2.1.3 The Wireless Routing Protocol .....	6
2.2 Reactive Sensor Data Routing .....	6
2.2.1 Ad Hoc On-Demand Distance Vector Routing .....	6
2.2.2 Dynamic Source Routing .....	8
2.2.3 Temporally Ordered Routing Algorithm .....	10
Chapter 3. Cellular Automata Theory .....	12
3.1 Cellular Automata History .....	12
3.2 Cellular Automata Background .....	12
3.2.1 Cellular Automata Cells .....	13
3.2.2 Cellular Automata Rules .....	13
3.2.3 Cellular Automata Neighborhood .....	14
3.3 Simple CA Applications Examples .....	17
3.3.1 Game of Life .....	17
3.3.2 Maze Solving .....	18
Chapter 4. Simulation Cantor Tool .....	20
4.1 Overview .....	20
4.2 Cantor Details .....	21
Chapter 5. Spin Glass Model. ....	25
5.1 Collective Phenomena to Ising Model .....	25
5.2 Spin Glass Routing. ....	27
5.2.1 Implementation Details. ....	27
5.2.1.1 Simulation Parameters. ....	27
5.2.1.2 Spin Glass Rules. ....	30
5.2.2 Damping Effect Theory .....	33
5.2.3 Topological Disturbance .....	35
5.2.4 Adaptation Phases .....	36
5.3 Result Analysis. ....	39
5.3.1 Temperature Effect. ....	39
5.3.2 Cell Failure Probability .....	39
5.3.3 Adaptation to Disturbance at Different Temperatures .....	42

5.3.4 Disturbance Locations and Energy Map .....	42
5.3.5 Damping Effect .....	44
5.3.6 Placing Obstacles .....	45
Chapter 6. Multi-fractal Model .....	47
6.1 Multi-fractal Background .....	47
6.1.1 Fractal Dimensions .....	47
6.1.2 Self Similarity .....	48
6.2 Diffusion Limited Aggregation Fractal .....	49
6.3 Multi-fractal Routing .....	51
6.3.1 Multi-fractal Routing Theory .....	51
6.3.2 Properties of Inhibition Curve .....	52
6.3.3 Weight Factor for Cells .....	53
6.4 Result Analysis .....	54
6.4.1 Snowflake Like Crystallization Approach .....	54
6.4.2 All One Probabilities .....	55
6.4.3 Different Inhibition Curves .....	56
6.4.3.1 Linear Curve .....	56
6.4.3.2 Quadratic Curve .....	59
6.4.4 Adaptation to Topological Disturbance .....	62
6.4.5 Discussion of Probabilities Set .....	63
Chapter 7. Conclusion .....	65
7.1 Extension to General CA Model .....	65
7.1.1 Time Synchronization .....	65
7.1.2 Arbitrarily Shaped Grid and Cell .....	66
7.2 Communication Overhead Analysis .....	66
7.3 Comparison of Our Routing Models with Other Models .....	67
Chapter 8. Future Work .....	69
8.1 Practical Sensor Grid Configuration .....	69
8.2 From Flat to Hierarchical Structure .....	71
8.3 3D Animation .....	72
References .....	74
Vita .....	75

## List of Tables

2.1	DSDVR routing table for node X. ....	4
4.1	Parameters of Cellular Automata. ....	20
4.2	Cantor features. ....	22
5.1	Example of Spin Glass routing table. ....	29
5.2	Potential energy values of different waves versus steps. ....	38
7.1	Comparison of routing protocols. ....	68

## List of Figures

2.1	DSDVR example. . . . .	3
2.2	Cluster head gateway switch routing. . . . .	5
2.3	Propagation of route request packet,RREQ. . . . .	7
2.4	Path traversed by route reply packet, RREP. . . . .	8
2.5	Propagation of RREQ and Building of route record during route discovery. . . . .	9
2.6	Propagation of route reply with the route record. . . . .	10
2.7	Temporally ordered routing algorithms. . . . .	11
3.1	Von Neumann neighborhood. . . . .	14
3.2	Moore neighborhood. . . . .	15
3.3	Extended Moore neighborhood. . . . .	15
3.4	Parity rule of CA with an initial single dot. . . . .	16
3.5	Pattern of parity rule at time step 188. . . . .	17
3.6	An example of Game of Life at initial step, 10,25,72. . . . .	18
3.7	Maze problem. . . . .	18
3.8	Maze problem with dead ends removed. . . . .	19
4.1	A 3D arbitrary network neighborhood example. . . . .	22
4.2	Cantor data flow. . . . .	24
5.1	One-dimensional spin array. . . . .	26
5.2	Example of a two-dimensional grid. . . . .	28
5.3	Eight possible spin directions of each spin in routing model. . . . .	31
5.4	Boltzmann probability distribution computation example. . . . .	32
5.5	Potential field of an example grid. . . . .	37
5.6	Final potential energy map. . . . .	38

5.7	Adaptation curve with various temperatures. ....	40
5.8	Failure probabilities at 0.005,0.01,0.03,0.05,0.1 and 0.3. ....	41
5.9	Adaptation to disturbance at various temperature. ....	42
5.10	Adaptation curve and energy map when close door at 10,7. ....	43
5.11	Adaptation curve and energy map when close door at 2,6. ....	43
5.12	Disturbance introduced at gen20. ....	44
5.13	No disturbance introduced and high failure probability from gen10 to 20. ....	44
5.14	Placing a large number of obstacles in sequence. ....	45
5.15	Placing two obstacles. ....	46
6.1	Koch curve. ....	49
6.2	Initial single seed of DLA. ....	50
6.3	DLA growing tree ....	50
6.4	Linear inhibition curve with $P$ to be 1. ....	53
6.5	Quadratic inhibition curve with $P$ to be 0.4. ....	53
6.6	Tree stop growing with snowflake logic. ....	55
6.7	Tree with all 1 probabilities ....	56
6.8	Linear inhibition curve with weight factor for free cell and open door. ....	57
6.9	Tree with linear inhibition curve with weight factor 0.35 at gen50. ....	57
6.10	Tree with linear curve with weight factor 0.3 at gen50. ....	58
6.11	Tree with linear curve with weight factor 0.2 at gen70. ....	59
6.12	Quadratic inhibition curve with weight factors for free cell and open door. ....	60
6.13	Tree with quadratic curve at gen50. ....	60
6.14	Tree with quadratic curve with weight factor 0.3 at gen50. ....	61
6.15	Tree with quadratic curve with weight factor 0.2 at gen70. ....	61
6.16	Final routing tree at gen60. ....	62

6.17	Adaptation curve to two disturbances. . . . .	63
8.1	A 2D real ad hoc WNS sensor distribution example. . . . .	70
8.2	Relationships between hierarchy levels. . . . .	71
8.3	An example of a real military battlefield scenario from RRB presentation. . . . .	72
8.4	Example of a 3D CA lay out with six neighbors. . . . .	73

## Abstract

Wireless sensor network (WSN) has been attracting research efforts due to the rapidly increasing applications in military and civilian fields. An important issue in wireless sensor network is how to send information in an efficient and adaptive way. Information can be directly sent back to the base station or through a sequence of intermediate nodes. In the later case, it becomes the problem of routing. Current routing protocols can be categorized into two groups, namely table-drive (proactive) routing protocols and source-initiated on-demand (reactive) routing. For ad hoc wireless sensor network, routing protocols must deal with some unique constraints such as energy conservation, low bandwidth, high error rate and unpredictable topology, of which wired network might not possess. Thus, a routing protocol, which is energy efficient, self-adaptive and error tolerant is highly demanded. A new peer to peer (P2P) routing notion based on the theory of cellular automata has been put forward to solve this problem. We proposed two different models, namely Spin Glass (Physics) inspired model and Multi-fractal (Chemistry) inspired model. Our new routing models are distributed in computation and self-adaptive to topological disturbance. All these merits can not only save significant amount of communication and computation cost but also well adapt to the highly volatile environment of ad hoc WSN. With the cellular automata Cantor modeling tool, we implemented two dynamic link libraries (DLL) in C++ and the corresponding graphic display procedures in Tcl/tk. Results of each model's routing ability are discussed and hopefully it will lead to new peer to peer algorithms, which can combine the advantages of current models.

# Chapter 1

## Introduction

Since the advent of wireless sensor network in 1970's, more and more applications both in civilian and military fields are emerging. Two kinds of mobile wireless sensor network exist at present. One is known as the infrastructure network. The mobile unit is connected with the nearest base station that is within its communication radius. The current mobile telephone system adopts this mode. The other one is called infrastructure-less mobile network, also ad hoc network. No fixed routers are needed and all mobile units are capable of movement and still being able to self-organize and establish communication in an arbitrary manner. What we will discuss is the second type.

Some typical commercial WSN scenarios include conference, manufacturing monitoring, emergency services and medical examination etc. People nowadays attend a meeting with their laptops. Communication would be much easier if a wireless network can be setup instantly instead of traditional wired network. The continuous and intense investment on wireless sensor network from DARPA also instigates the advance of WSN in modern military application. Rambling soldiers equipped with communicators need to communicate with each other or with the headquarters in an ad hoc manner without having to rely on base station. Mobile detective robots furnished with audio and visual sensors also need to send back their data to processing center via ad hoc communication channel.

Routing is referred to as how communication among mobile nodes is established in the network. There are two variations of routing protocols, namely the Table-Drive Routing Protocol and Demand-driven. Due to the limited battery power supply, energy conservation is one of the crucial issues in ad hoc WSN. Energy are consumed by sensing, data processing and communication processes. In a wireless sensor network, communication consumes the majority

of the energy. In order to prolong the lifetime of the network, minimizing communication among nodes is highly desired. Here we proposed a new routing idea using cellular automata, which is decentralized in computing and self-adaptive to topological change. Two models are put forward to tackle routing problem, namely Spin Glass inspired model and Multi-fractal inspired model. They aim to establish the shortest paths from a data sink node to the rest of the nodes in the network. The key point of this new approach is that it attempts to reach a global goal based on local decision made by each sensor node. No communication on a global scale is needed. Thus, Heavy load of communication is avoided to save energy. Moreover, high efficiency is obtained by the nature of parallel Computation. The CA based routing is also highly adaptive to topological change, which is a fascinating aspect for ad hoc sensor network. High node failure and further sensor deployment needs can also be satisfied. Although our initial simulation cannot exactly reflect the actual ad hoc sensor network scenario, we discussed some necessary issues about how to transform our current model to suit it. The main idea of my thesis is to propose a routing protocol idea and hopefully further investigation can be done on it to lead to mature practical application.

This thesis is organized as follows: Chapter 2 gives an overview of ad hoc mobile wireless networks routing protocols. Chapter 3 talks about the background of cellular automata and its applications. Chapter 4 describes the simulation CATOR tool. Chapter 5 presents the Spin Glass model, the theory behind it, implementation detail and the result analysis. Chapter 6 is devoted to the Multi-fractal model. Chapter 7 comes the conclusion and Chapter 8 gives the plan for future work.

## Chapter 2

# Overview of Current Sensor Data Routing Protocols

### 2.1 Proactive Sensor Data Routing

Proactive Sensor Data Routing is also called table-driven routing protocols. It maintains up-to-date routing tables from each node to every other node in the network. Topology change is propagated throughout the network to update the routing tables of each node. The storage requirement of routing tables and the transmission overhead of topology change are the basic drawbacks of this protocol. The following three approaches are three variations of reactive sensor data routing. They differ from each other mainly in the number of routing tables required and the way of broadcasting network structure change.

#### 2.1.1 Destination-Sequenced Distance-Vector Routing

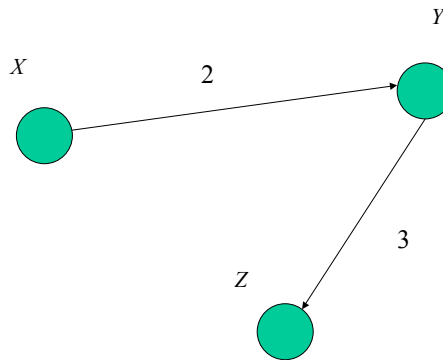


Fig.2.1 DSDVR example

Destination sequenced distance vector routing (DSDVR) is derived from the classical Bellman-Ford routing mechanism. Each node maintains routing information to all other nodes and the corresponding hops value. Regular updating the routing table incur big communication overhead

within the network and no sleep mode is allowed in the mode. Each routing table has the entries like shown below in Table 2.1

Table 2.1 DSDVR routing table for node X [2]

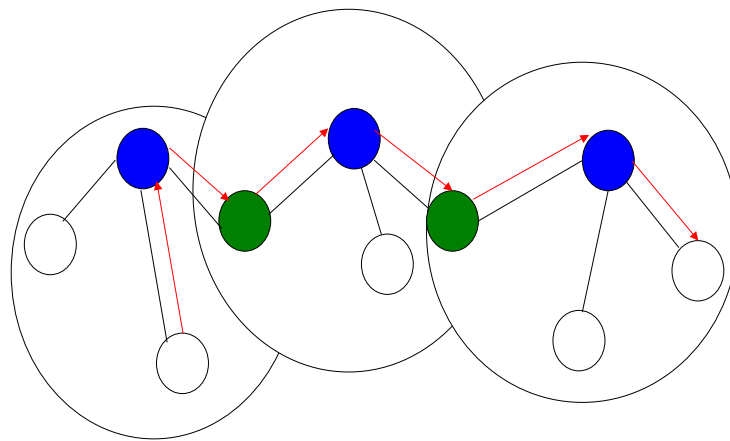
Destination	Next	Hops	Sequence#	Install Time	Stable Data
X	X	0	X-550	25300	P_X
Y	Y	2	Y-102	52891	P_Y
Z	Y	5	Z-588	12577	P_Z

Next gives the next hop node. Hops give the number of hops to the destination. Sequence number originated from destination with a number to represent its freshness. Hence mobile nodes can distinguish stale routes from new ones, also ensure loop freeness. Install time represents when entry was established. Stable data points to a table, which contains information about stableness of that route. And it can be used to damp the network fluctuation. Routing table updates are done on a periodic basis. Two types of packets can be used to update the routing tables. Full dump packet carries all available routing information. Incremental packet only contains those entries that have been changed since the last full dump. This can decrease the amount of traffic generated. This protocol is simple and loops free through the usage of sequence number. However, No sleep mode is allowed, which would prevent nodes from entering the energy saving mode. Scalability is a big problem for this protocol.

### 2.1.2 Cluster Head Gateway Switch Routing

The cluster head gateway switch routing protocol takes a hierarchical structure, with a cluster head controls a group of sensor nodes within its radio transmission range. There types of nodes exist in the network and they are node, gateway and cluster head. A gateway node serves as an

intermediate between cluster heads. A distributed cluster head selection algorithm is utilized to select a node as the cluster head. However, frequent cluster head vote can degrade the performance since limited energy and other resources are consumed on cluster head vote instead of useful packet relay. Hence, a least cluster change (LCC) clustering algorithm comes in. In LCC, cluster head change occurs only if network change causes two cluster heads to fall into one cluster or one of the nodes moves out of the range of all the cluster heads.



White circle: node  
 Blue circle: cluster head  
 Green circle: gateway

Fig 2.2 Cluster head gateway switch routing

Figure 2.2 illustrates a routing example between two nodes in two clusters. Each node should keep a cluster member table, which stores the destination cluster head for each node in the network. Each node periodically broadcasts dynamic cluster head member table using DSDV algorithm. Upon reception of the cluster head member table, nodes update their cluster head member tables. A routing table used to determine the next hop in order to reach the destination is also needed. When a packet arrives, a node will look up its cluster member table and routing table

to determine the nearest cluster head along the route to the destination. Then, it would know the next node to jump to in order to reach that selected cluster head. Packet is sent to that node [1].

### 2.1.3 The Wireless Routing Protocol

Wireless routing protocol is a table-based protocol with the intention of maintaining routing information among all nodes in the network. Each node in the network maintains a routing, distance, link cost table and a message retransmission list tables. Nodes send update messages to its neighbors after receiving updates from neighbors or detecting a link change to a neighbor. In case of link loss between two nodes, update messages are sent to all their neighbors. The neighbors then modify their distance table entries and try to find new possible paths through other valid nodes. New paths are relayed to the original link loss nodes to update their tables according to new established paths. Nodes get aware of the existence of their neighbors from the receipt of acknowledgements. If there is no change in routing table since last update, node has to send an idle hello message to ensure connectivity. Otherwise, system would assume link failure [1].

This algorithm checks for consistency of all its neighbors every time it detects a change in link of any of its neighbors. Looping is eliminated and convergence is fast [2].

## 2.2 Reactive Sensor Data Routing

Reactive methods are called on-demand routing protocols. Since they run on the demand of a source node to a destination node, the control packet overhead is greatly reduced compared with table driven one. This protocol starts from the route discovery procedure followed by the route maintenance procedure until either destination becomes inaccessible or the route is no longer desired.

### 2.2.1 Ad Hoc On-Demand Distance Vector Routing

Ad hoc on-demand distance vector routing was built from DSDV algorithm previously described. Instead of maintaining a complete list of routes as in the DSDV algorithm, only nodes that are on a selected path are required to maintain routing information or participate in routing table exchanges. The number of required broadcast is minimized as well.

As in figure 2.3, when a node is seeking a path to destination, it broadcasts a route request packet (RREQ) to its neighbors, which then pass the request to its neighbors. The flood continues until either the destination or an intermediate node with an up-to-date route to the destination is reached. The broadcast ID and the IP address uniquely designate a RREQ. Each node receiving the message creates a reverse route to source based on the first copy of the broadcast packet received. As show is Fig2.4, destination unicasts a route reply message back to the neighbor from which it first received the RREQ packet. As the RREQ is routed back to the source, nodes along the path create forward route entries. Thus, each node remembers only the next hop required to reach the destination, not the whole route. A route timer is associated with a route entry, if the route entry is not used for a specific lifetime, the route entry will be deleted as a stale route [1]. This method requires periodic updates of routing information. AODV broadcasts periodic hello messages. Failure to receive consecutive number of hello messages determines failure of a node. In that case source initiates a new route discovery [2].

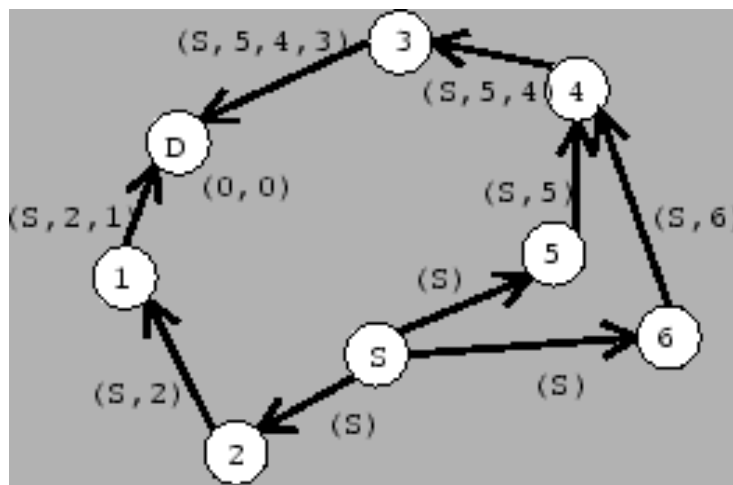


Fig 2.3 Propagation of route request packet, RREQ [2]

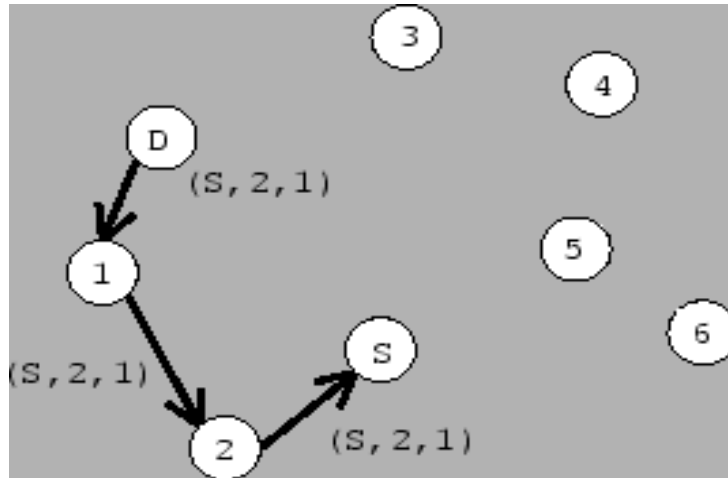


Fig 2.4 Path traversed by route reply packet, RREP [2]

### 2.2.2 Dynamic Source Routing

Dynamic source routing protocol (DSR) is an on-demand routing protocol. Each node maintains a route cache recording the source routes that it is aware of. Route cache entries are being constantly updated once new routes are learned. In Fig2.5, when a node S requires a route to a destination D, it first examines its route cache to see if a route to the destination is already exists. If a fresh route is found, it will use the route in the cache. Otherwise, it initiates the route discovery phase by broadcast the RREQ packet to its neighbors, which will do the same action and so on. Propagation will stop if destination node or intermediate node, which contain unexpired route to destination, is located. Node receiving the RREQ packet would first check if it knows of any route to the destination through route cache. If not, it adds its own address to the route record and then continues to forward the packet along its outgoing links. To limit the number of route requests propagated, a node processes a route request packet only if it has not already seen the packet and its address is not present in the route record of the packet. A route reply is generated by destination or intermediate node. Destination node will place the route record into the route reply. An intermediate node will append its cached route to the route record.

Fig2.6 shows the transmission of the route reply from D with associated route record back to the source node S.

DSR utilizes route error packets and acknowledgments for route maintenance. Any topological change or fatal transmission errors will be reflected in the route cache of nodes. If a node receives an error message due to fatal transmission error or failure link, it then deletes the erroneous hop from all of its route cache entries, and selects a new route [1]. The study on DSR indicates that DSR, wise usage of route cache can greatly reduce the amount of control messages. DSR does not make any periodic update messaging, thus avoiding wasting bandwidth. This protocol learns from what it hears, not only the routes it needs. [2].

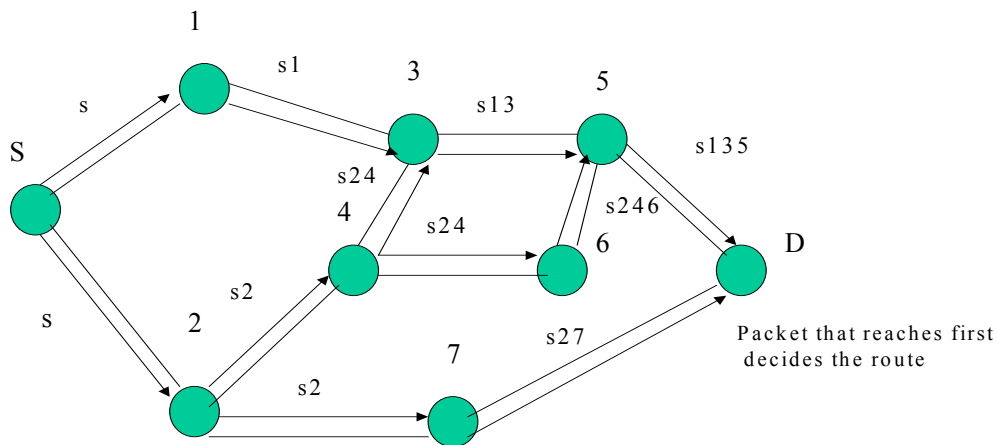


Fig 2.5 Propagation of RREQ and building of route record during route discovery

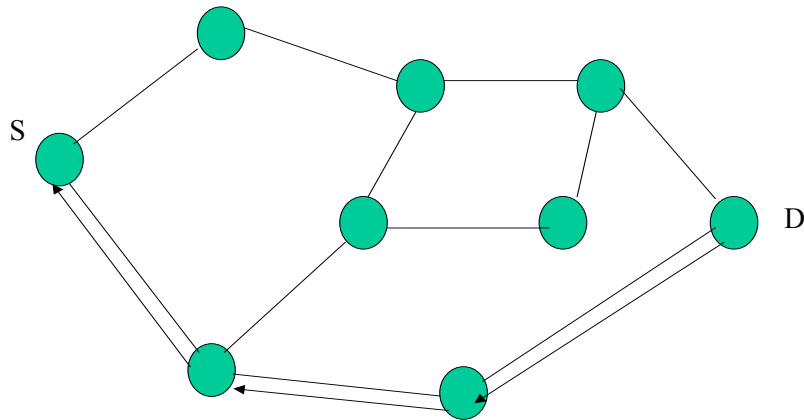


Fig 2.6 Propagation of route reply with the route record

### 2.2.3 Temporally Ordered Routing Algorithm

The temporally ordered routing algorithm (TORA) is a highly adaptive distributed routing algorithm based on link reversal. TORA is source initiated and can discover multiple routes between any source and destination pairs. The key concept of TORA is the localization of messages exchange in case of topological change. A node needs to maintain the routing information about adjacent nodes [1]. During the route establishment phases, nodes use a height metric to construct a directed acyclic graph (DAG) routed at the destination. Links are assigned a direction according to the relative height metric of neighboring nodes. In case of topological change, DAG is reestablished at the same destination.

A good analogy would be water flowing down the hill through pipes. Hilltop is the source, pipes are links, and pipe connections are nodes. TORA assigns level numbers to each node down the hill. Water will flow from node with big height metric to node with small height metric. A water flow pattern is formed driven by the height metric at different nodes [2].

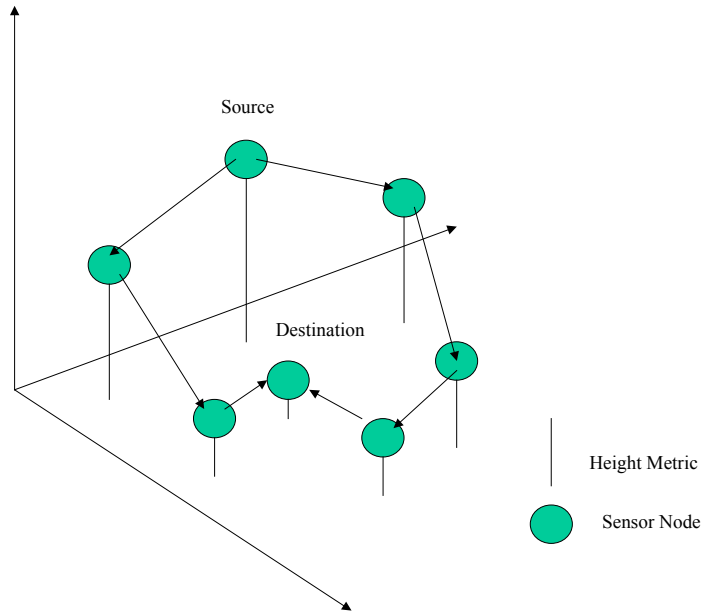


Fig 2.7 Temporally ordered routing algorithms [1]

Actually, this routing protocol shares some similar properties with our Spin Glass model in localization of control messages and establishment of height metric graph. You would be able to see that when we proceed to Chapter 5.

## Chapter 3

### Cellular Automata Theory

#### 3.1 Cellular Automata History

John von Neumann first conceived cellular Automata in the early 1950's. By 1947, instigated by the ideas on automata developed by Post (1936) and Turing (1936), von Neumann started his investigation on self-reproductive device. At first, he tried to describe a continuous model of a self-reproducing automaton by non-linear partial differential equations. However, von Neumann soon found it difficult to provide the rigorous and explicit rules and instructions needed to construct such an automaton. He switched his effort to a model of self-reproduction using an array of computing elements, which was suggested by Stanislaw Ulam. Von Neumann furthermore discerned the parallel computational ability embedded in this idea. Unfortunately, his death in 1957 prohibited him from conducting further investigation on it.

#### 3.2 Cellular Automata Background

What is a cellular automaton? Definition of a cellular automaton would be that an array of identically programmed automata, or "cells", which interact with nearby cells and advance in discrete time and space in one, two, three or even higher dimensions. We can visualize cellular automata system as a universe. A uniform three-dimensional lattice represents the space; each site has one or more bits to represent states and rules to compute next new state based on its neighborhood states; Time advances in discrete steps instead of continuity.

A common clock simultaneously drives the running of all cells. Thus, all cells can evolve synchronously. In our WSN, it may be difficult to maintain a common physical clock as time advances, but physical time must be needed to relate events in the physical world, such as object tracking, distributed beam-forming, duplicate detection, and temporal order delivery as kalman filter. And this is where time synchronization comes into play. We discussed it in Chapter 7.

We can say that a cellular automaton can stand for a general paradigm for parallel computation, just as what Turing machine do for serial computation [4]. And this parallelism is exactly what predicted by von Neumann at his lifetime but without further study.

CA has successfully modeled a lot of complex and non-equilibrium phenomena. In other words, collective behavior can be modeled by the sum of simply interactive entities. Although local rules are perfectly known, there is still uncertainty as to the extrapolation of the global behaviors. We cannot simply sum up local behaviors to get the global behaviors as we can do to bricks. The key point of CA is that there is no centralized authority to control the global behavior, but the local interaction among identical entities. This unique property of CA makes CA a proper tool to solve some problems requiring high efficiency and limited communication.

### 3.2.1 Cellular Automata Cells

Homogenous entities, each with several bits to represent states and embedded look-up table for cells to find out next new state at different entries. It is essentially a programmed automaton, sniffing information only from its neighborhood and evolves in a serial manner at the microscopic level. At the macroscopic level, the computation is in a parallel manner. Identical cells also make simulation much easier. In our WSN, arbitrarily shaped cells are necessary to cope with the variety of sensor equipment and terrain situation. It will be discussed in Chapter 7.

### 3.2.2 Cellular Automata Rules

The receipt that each cells use to figure out its next state, simple however, can exemplify many complex phenomena ranging from Physics, Chemistry to Biology. The traditional rules are crispy, which gives deterministic behavior for the cells. A generalized probabilistic rule is proposed to specify the probability of a given state. A cell would evolve into the next new state out of several possible states at a probability. This turns out to be a very powerful extension to traditional deterministic rules. More natural phenomena can be modeled with this extension included.

### 3.2.3 Cellular Automata Neighborhood

Each cell is associated with its adjacent cells as its neighborhood. There are different ways of defining neighborhood. Communication is limited within the neighborhoods. Global communication overhead is avoided in this way also data transmission error rate is reduced for wireless sensor network. Let's illustrate neighborhood by a two dimensional grid.

#### a. von Neumann Neighborhood

Cells directly to the north and the south, east and west from each cell are called the von Neumann neighborhood of this cell. The radius of this definition is 1.

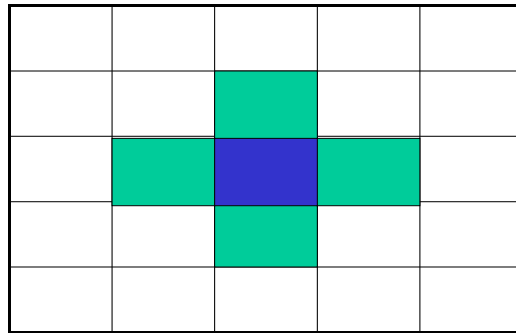


Fig 3.1 von Neumann neighborhood

#### b. Moore Neighborhood

Moore neighborhood is an enlargement of the von Neumann neighborhood. In addition to the cells in the von Neumann neighborhood, four cells located at the diagonal are also included. The radius is also 1.

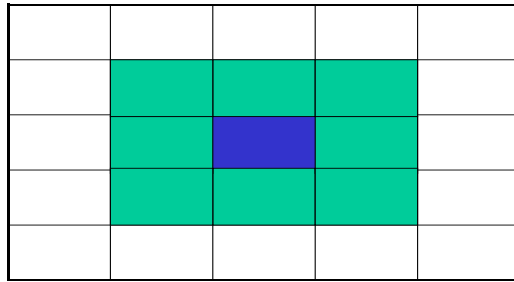


Fig 3.2 Moore neighborhood

c. Extended Moore Neighborhood

Similar to Moore Neighborhood, with the radius enlarged to be 2. An additional layer of adjacent cells is included in the neighborhood.

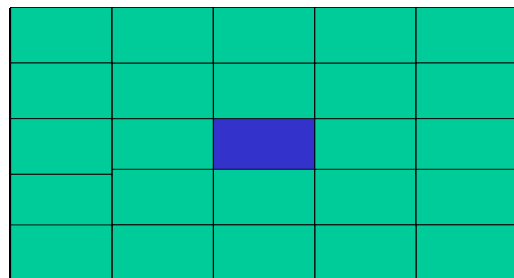


Fig 3.3 Extended Moore neighborhood

It would a natural way to simulate discrete dynamic and complex phenomena with cellular automata. However, this flexibility and generality comes with a cost. Instead of several variables used in a partial differential equation to describe a continuous dynamical system, a large number of variables are needed for cellular automata system with each site the same number of states. A large number of cells are required for large-scale system with complex behaviors. Moreover, the time steps required for evolution would also be large [3]. Also, the there is little

flexibility in defining parameters of rules. The traditional Boolean function of evolution rules imposed constraints on some processes based on probabilities. Moreover, there are some inherent drawbacks coming with the discrete nature of CA.

We present a very simple example of CA to illustrate how complexity is derived based on simple rules. Edward Fredkin initially proposed parity rule in the 1970s. Suppose we have a two dimensional square grid, each site is labeled by its position  $(i, j)$  where  $i, j$  are column and row indices. A function  $S_t(i, j)$  is associated with each cell to represent its state. CA rules specifies how new states  $S_{t+1}(i, j)$  are evolved from old states  $S_t(i, j)$ . We start from time step 0 with an initial configuration on the square lattice. Parity rules are defined as follows:

Each cell computes the sum of its von Neumann Neighborhood states. If the sum is even, the new state is 0 (white); else it is 1 (black).

$$S_{t+1}(i, j) \leftarrow S_t(i+1, j) \text{ xor } S_t(i-1, j) \text{ xor } S_t(i, j+1) \text{ xor } S_t(i, j-1)$$

As we can see from the following figures, simple rules can unravel complex patterns [5]. Also, the pattern shown in Fig 3.5 contains a good mixture of regularity and irregularity [16].

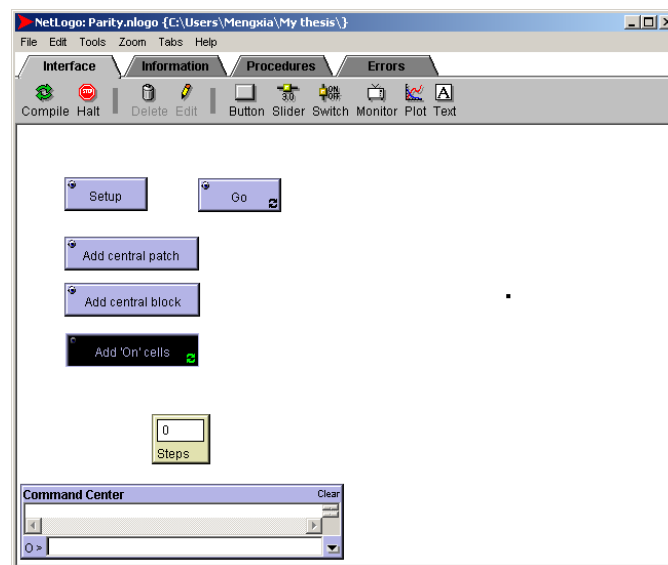


Fig 3.4 Parity Rule of CA with an initial single dot (Generated by Netlogo1.1)

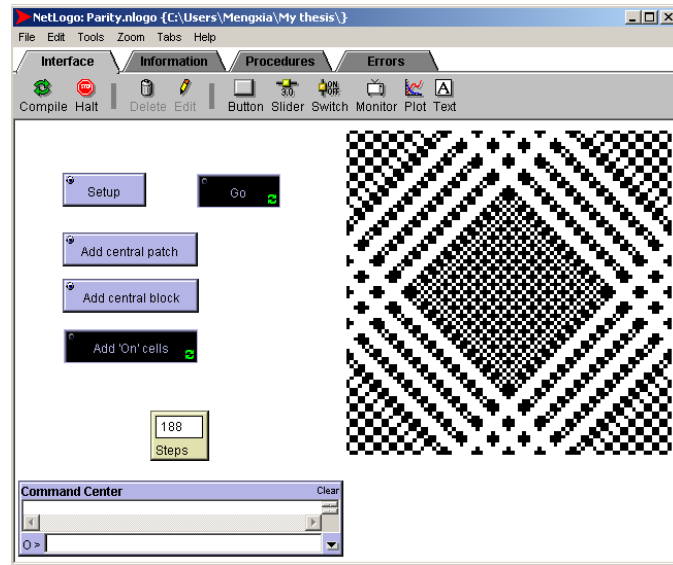


Fig.3.5 Pattern of parity rule at time step 188

### 3.3 Simple CA Applications Examples

#### 3.3.1 Game of Life

Mathematician John Conway proposed his famous game of life in 1970s. Each cell in a two-dimensional lattice can be either live (populated) or dead (unpopulated). Neighborhood is of von Neumann type. Updating rules of the game of life is as follows:

Each cell with one or no neighbors dies, as if by isolation or loneliness.

Each cell with four or more neighbors dies from overpopulation or overcrowd-ness.

Each cell with two or three neighbors remains unchanged.

An unpopulated cell with three neighbors becomes populated, just as being newly born.

In Fig 3.6, occupied yellow cells are used to represent live individuals. Starting from initial distribution, complex distribution patterns evolve in later generations. Irregularity and nested structure can be both observed. We could simulate the life distribution at later generation by following certain biological individual interaction rules. It can help us to predict species distribution of a certain region in a long term.

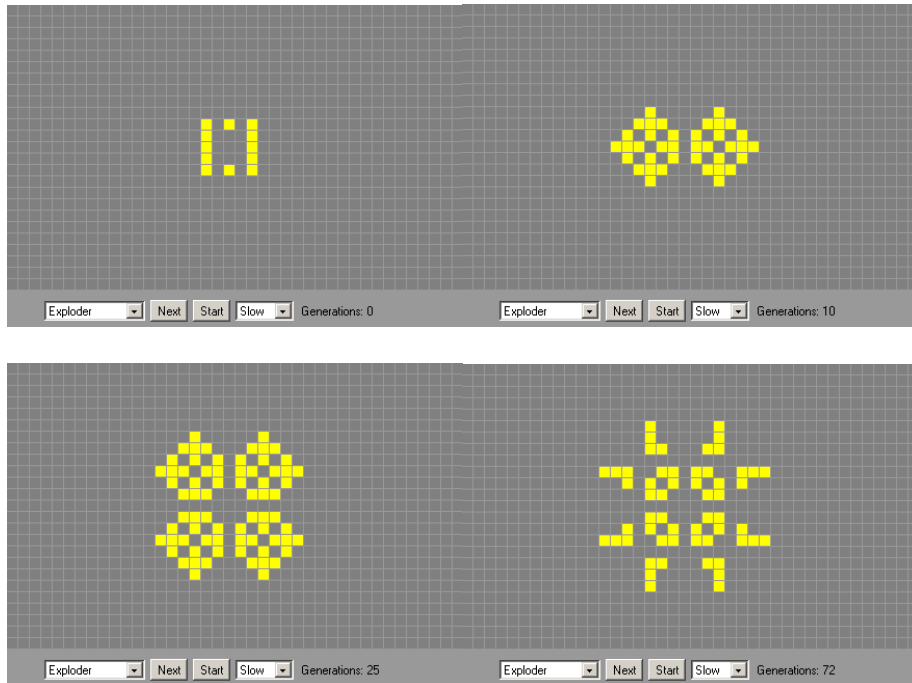


Fig 3.6 An example of Game of life at initial step, 10,25,72 [6]

### 3.3.2 Maze Solving

We all should be familiar with the maze and mouse game. Suppose there is maze with only one entrance and one exit like shown in Fig 3.7. How can a mouse find a route from the entrance to the exit?

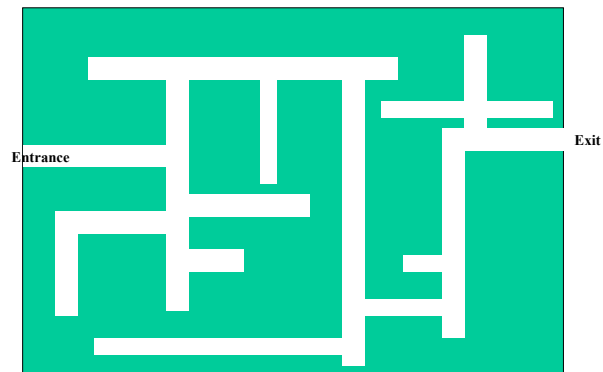


Fig 3.7 Maze problem



## Chapter 4

### Simulation Cantor Tool

#### 4.1 Overview

The Cantor tool is a simulator based on the Generic Automata with Interacting Agents (GAIA) model. And it is capable of modeling dynamic, discrete (in time and space) event systems, which consist of large, interacting individuals and display collective phenomena. Cantor was developed by ARL at PSU in 2001 and has been successfully used to simulate many Cellular Automata problems including traffic engineering, network communication and sensor data routing etc.

GAIA can be used to model mobile agent travel in the network, processing data and changing its state as it migrates. The mobile agent simulates biological ant pheromone in Pheromone routing model, which is implemented by ARL at PSU. In our Spin Glass and Multi-fractal model, no mobile agents are needed. Cantor can be used to tackle the forward problem, which considers local behaviors and determines their global consequences. The backward problem is about how to derive local behaviors with globally desirable consequences. The Cantor model proves to solve the forward problem very well but the backward problem still remains unsolved. We intend to use Cantor to find a resolution for backward problem. Table 4.1 display Cellular Automata Parameters used in Cantor.

Table 4.1. Parameters of Cellular Automata [15]

<b>Cellular Automata Parameters</b>	
$d$	Dimension of the automaton
$l[d]$	Vector indicating the number of elements in each dimension
$r$	Radius of the automaton
$\mathcal{A}$	Set of transition functions. Each element may have a unique transition function
$S[]$	State vector for each automaton
$B$	Set of behaviors, not tied to specific CA elements

$l[d]$  gives the maximum number of elements in each dimension of  $d$ . For example, if we want to establish a 2D 25x50 grid,  $l[x]$  equals 25 and  $l[y]$  equals 50. Transition functions in  $\Delta$  are used to describe network topology by either allowing or prohibiting communications between nodes. Radius of automata defines the scope of neighborhood of each node. It is sort of like the radio transmission radius of a wireless sensor node. All nodes within this radius serve as the neighborhood of this node. Each node has its own instance of the state vector  $S[]$  of the same structure. State vector can be used to simultaneously capture multiple aspects of the problem space. For example, we have potential value and spin direction for each node in Spin Glass model. Not all aspects can be visualized at the same time, but since the state consists of a vector, the unused aspect can be stored for later use. The set of behaviors  $B$  is not tied to any particular cell element. We can choose to bind specific behavior to specific cells or run the same behavior for all cells. Behaviors can also move from cell to cell and even to mobile agents.

The behavior of a specific element is defined by the interaction between the  $S[]$ ,  $B$  and  $\Delta$  local to the element, just as described in Cellular Automata theory. At every time step, each node looks up its own transition table, and determines its new values for  $S[]$  based on its and its neighbors' current state vectors. This evaluation occurs concurrently for all nodes during a particular time step. The transition function can consist of a proper subset of behaviors  $B$ . Because behavior can migrate from nodes to nodes, node can have different behaviors at various time steps [15].

Fig 4.1 displays the neighborhood of a node in a 3D system. System can be in any arbitrary number of dimensions. Node  $S$  can have up to  $n$  neighbors, which are within its radius  $r$ .

#### 4.2 Cantor Details

To utilize this model, a simulator based on the GAIA model was constructed by ARL at PSU. The simulator consists of a front end that emulates a WSN routing network and a back end that allows the evolution to be visualized. Cantor users have all the features in Table 4.2.

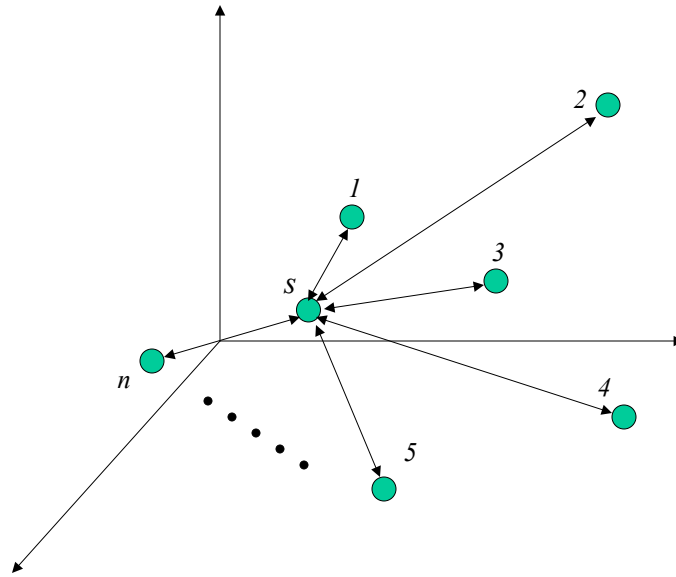


Figure 4.1. A 3D arbitrary network neighborhoods example

Table 4.2. Cantor features [15]

Features of Cantor	
1	Construct $k$ -dimensional cellular grids
2	Construct rule based cellular neighborhoods
3	Assign evolutionary rules on a cell-by-cell, region-by-region or grid basis
4	Record evolution of cell states
5	Model free agents in the cellular space
6	Construct re-writable, swappable and evolving rules
7	Visualize the results in 2 or 3 dimensions
8	Reprogram the system using a scripting language
9	Perform mathematical analysis of cellular grids

There are three architecture subsystems in Cantor, namely the Cellular Automata Library (static and dynamic CA library), the TCL Extension and the Vtk Library.

The static portion of CA library defines the basic classes of CA, such as agent, grid, cell, neighborhoods, behavior etc., and a dynamic portion (DLL) that allows the user to customize cell states and behaviors according to their specific needs. In our case, we have Spin Glass DLL and

Multi-fractal DLL. The CA Library is written in C++ and can be run under Unix and Windows. The grid evolves by executing the behavior for each cell in turn. Once every cell in grid finishes its updating, a new generation starts. The updated state vector won't take effect until that generation for the grid is finished. And before new generation is initiated, updated state vectors will overwrite all old state vectors. This enforces the consistency of parallel computing by serial like execution. The CA Library also provides for cell-based agents that reside at cells and may have their own state and behaviors. In the Pheromone model developed at PSU, the mobile agent is used to simulate ant pheromone. Evolution of the grid is recorded in XML files. In order to visualize selected attributes in the state vector, XML allows selective access through tags in XML file. XML parser is included in the dynamic library, which allows DLL to parse out their own XML outputs depending on various visualization attributes. The data analysis routines are now under construction. They may appear in the scripting language. Data analysis tools include, entropy analysis, probability and statistical summaries of generations and runs.

The TCL Extension is simply a wrapper for the CA Library. The extension registers certain functions of the CA Library with the TCL interpreter. A user defined TCL script can load the parameterized calls to CA Library such as construct and initialize the grid, run a simulation, visualize a simulation via a TCL console.

Tcl/Tk, two different products originally developed at the University of California Berkeley, are a scripting language and a graphical interface development tool, respectively. The Vtk Library is a series of TCL/Vtk scripts designed to graphically display the evolution and interaction of cells and agents through generations. The library takes parsed XML data output as commands lines to the visualization system. State attributes are usually used to parameterize scripts functions. Thus, Users can visualize grid evolution of specific cell state attributes. For instance, a user may wish to visualize the spin direction at each cell. The Vtk visualization displays each cell as arrow representing the spin direction at that cell during that generation of evolution. The arrow directions changes in accordance with the spin direction value in state vector to display different

spin directions at each generation at this cell. As evolution continues, the user may see patterns of routing emerge based on spin directions at cells over time.

Data flow in the Cantor modeling tool is shown below. It first passes from the TCL script through the TCL extension to the CA Library. At each generation throughout the evolution, state information is sent to an XML history file and user defined parse function can parse the XML files to data files for either a VTK visualization or data analysis [15].

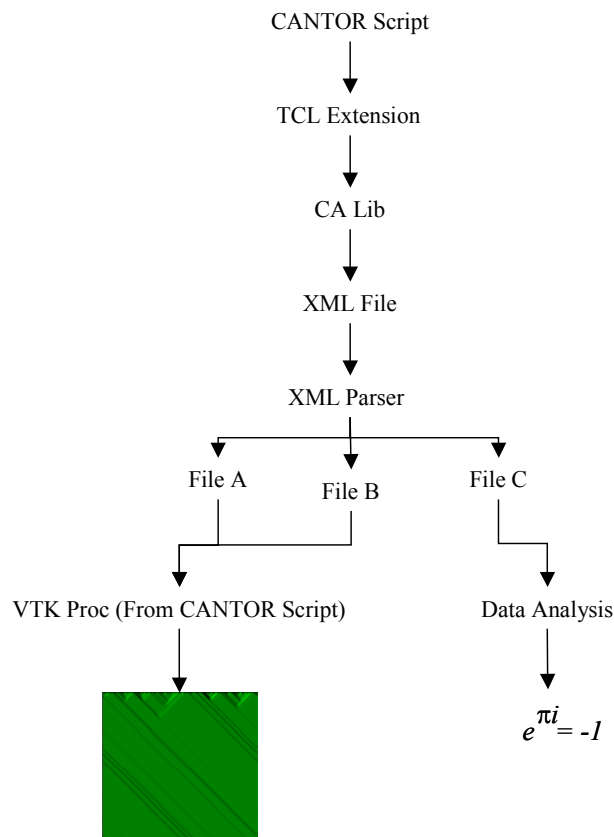


Fig 4.2. Cantor data flow [15]

## Chapter 5

### Spin Glass Model

#### 5.1 Collective Phenomena to Ising Model

We can always observe that a large number of identical individuals interacting with each other by following simple rules can display a macro-scale phenomenon, which is not displayed by individual. Examples include gases, fluid dynamics, sound waves, biological evolution, economics, and magnetization etc. Because cellular automata share the common properties of locality of interaction and identical individuals with collective phenomena, cellular automata is found to be able to model many collective phenomena with more complicated structured individuals and rules. Once getting into motion, the system is driven by the competition of two forces, namely promoting orderly structure and breaking up this order. An equilibrium state is achieved when there is no long-term winner between two forces at certain critical parameter.

Ising system, a concept in physics, is the one of the earliest CA models under investigation. Individual of the Ising system is the spin, which can be viewed as little magnets. Spins occupy fixed position in a one-dimensional or two-dimensional or even higher dimensional settings and the polarity is the only variable feature. Spin orientations is restricted to two orientations, depicted as up or down arrows. As we know, neighboring magnets pose force on each other depending on their relative orientation and distance. The magnetic field tunes up the overall state in which those spins are. If we visualize the interaction between two spins as a bond, the bond is at rest for parallel spins and under tension for anti-parallel spins. This kind of bond is called ferromagnetic bond. We assign 1 for the coupling energy for aligned spins, 0 for anti-parallel spins and the following configuration can be represented as shown below:

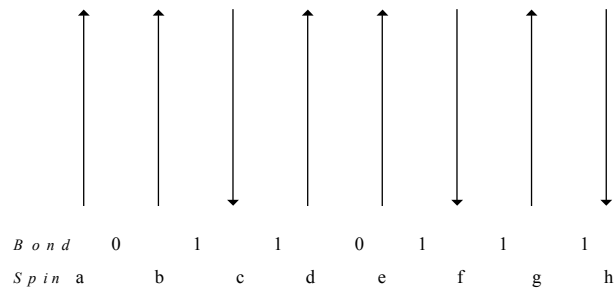


Fig 5.1 One-dimensional spin array

Ising-spin model can be in either one of the three conditions.

- a. Isolated system, energy is strictly conserved without heat exchange with the environment.  
A spin flip will not affect the whole system energy.
- b. System can exchange energy with its environment. Suppose each spin has an energy bank, spin can draw or deposit energy in the bank, retaining strict energy conservation with the bank energy counted in.
- c. System exchange energy with its environment. Visualize the environment as a massive heat bath, which can give and take energy from system with certain probability but itself is not affected by it. Energy conservation is in a statistical sense.

Let's look at the above example:

For the d spin, if it reverses its direction, left bond would be 0 and right bond would be 1. Since there is no energy gain or loss, the flip would be allowed under the isolated system condition.

Generally speaking, if the number of the up orientation spins equals the number of the down orientation spins in a spin's neighborhood, flip of that spin is allowed.

If each spin is furnished with a thrifty bank which can store up to two units of energy, the evolution obeys the following rules:

- a. Flip of a spin without energy gain or loss is allowed.
- b. If flip of a spin releases two units energy, deposit the energy only when the bank is empty.

c. If flip of a spin requires two units energy, withdraw two units of energy from the bank when the bank is full.

We can imagine energy bank as our conventional bank. We deposit money when we have extra and withdraw it when we need it [4].

Spin Glass is a particular variation of Ising model, in which bonds are of two types, namely ferromagnetic nature and anti-ferromagnetic nature. In anti-ferromagnetic bond, parallel spins are at tension and anti-parallel spins are at rest. We use Spin Glass model just as a notation.

## 5.2. Spin Glass Routing

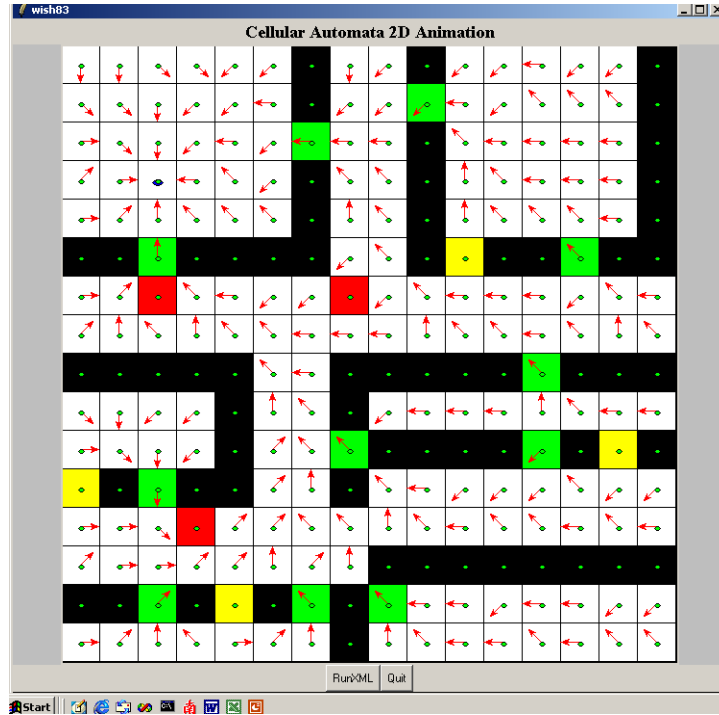
Our Spin Glass routing model adopts the third condition, in which energy conservation is only at the statistical level. Instead of using many variables to represent energy banks, a single parameter is used to indicate the willingness to supply energy, namely temperature, which is a good analogy as the magnetic field. One of the most significant developments of 19-th century physics was the discovery of the appropriate probability function to characterize the relative importance of the numerous microscopic configurations. Suppose the system is in thermal equilibrium with the heat bath and they are in a common temperature, says  $T$ . The probability that the specified state actually occurs is given by the Boltzmann probability distribution function. This is a significant extension to traditional deterministic cellular automata model. Probabilities of each of the possible outcome are calculated.

### 5.2.1 Implementation Details

#### 5.2.1.1 Simulation Parameters

We use a two-dimensional grid divided up to five types of cells, namely open door, close door, obstacle, wall and free cell. Fig 5.2 shows an example 16x16 grid.

Moore type neighborhood was introduced to include eight adjacent cells in the neighborhood to reflect the fact that sensor data transmission can go to all these directions.



Black block: Wall      Green block: Open Door      Yellow Block: Close Door  
 Red Block: Obstacle      White Block: Free Cell

Fig 5.2 Example of a two-dimensional grid

Each cell is associated with a weight parameter, which represents the energy needed to reach it via its neighborhood. Ideally, we can specify eight different weights each stands for the energy cost required from eight different directions. Since radio transmission energy from different directions is likely to be very different due to terrain variation, weather and ion density conditions. We aim to find lowest accumulated energy cost routes from any sensor to the data sink. To be more efficient, we should allow weight to be dynamic to be able to reflect the current network data congestion, in case of heavy data stream through a specific node, the weight of that node can be increased. Our Spin Glass routing algorithm will tend to search other route with less data stream load to lessen the congestion. Also, we can dynamically specify certain area to be highly subject to eavesdropping and weight increase in that area will reduce the routes usage of that area.

Temperature is used to tune up the macroscopic phase condition of the system. At high temperature, random force dominates. On the contrary, at low temperature, orderly force wins. It

is sort of like a magnetic field, which has an overall effect on all little magnets. Spins can be frozen below certain freezing point and no spin flips would happen no matter what. As the temperature goes above a preset threshold, system is free of any inhibition effect. There is a decreasing damping effect when the temperature is stepping from the freezing point to threshold.

In order to simulate the malfunction of nodes, node failure probabilities are introduced. In case of abnormal behavior, a node will randomly select a spin direction without following the Spin Glass logic.

Table 5.1 Example of Spin Glass routing table

Status	Sensor Node
Weight	Pointer to Float array
Energy Value	15
Next Hop Direction	North
Neighborhood	Pointer to cells array

A typical practical routing table each cell is required to maintain has entries as shown by Table 5.1. Status is the property of cell such as wall, obstacle, sensor node, open door etc, and it can be represented by an enumerate variable. Weight can be a pointer which points to a float array with each value represent the weight of a specific neighbor. Energy value is the shortest distance to the data sink with initial value to be infinite. Next hop is the spin direction of the current cell. Neighborhood points to cells array, which consist of the neighborhood of the cell, and the array can be dynamic updated in case of node failure or node redeployment. For our current simulation, we use a single weight for all neighbors and neighborhood is set to be Moore type by default. This simplification eases our job. However, it can be easily extended to fit the real scenario as discussed in Chapter 8.

### 5.2.1.2 Spin Glass Rules

In Spin Glass routing model, a potential field was established by propagation stemming from the data sink with zero to be its energy value. All neighbors of the data sink would calculate their energy value to be zero plus its own weight, which represent the energy required to reach it via its neighbors. In cellular automata mode, every cell will simultaneously look around its neighbors and pick one with the lowest energy value and plus its weight to be its energy value. And so on until all cells get their energy values to overwrite initial infinite value. The energy value stands for the shortest distance to the data sink, or the amount of energy required reaching the data sink. This step is similar to Lee algorithm, which was used to find a minimal path between two terminals if such path exists [17]. However, high malfunction probability and non-determinism in WSN routing makes simple application of Lee algorithm inappropriate, here comes the concept of Spin Glass.

At each step of establishing potential field, another type of cellular automata rule is applied. Each cell need to decide its spin direction out of eight possible directions instead of the traditional spin up and down directions based on current potential field.

- a. If a cell tries to point to a neighbor with higher energy value, the bond is at tense requiring extra energy from heat bath. The energy gap,  $E(s)$  is positive by subtracting its energy value from that of the neighbor for such hill climbing.
- b. On the contrary, if a cell tries to point to a neighbor with lower energy value, the bond is at ease releasing energy stored in the bond to the heat bath.  $E(s)$  is negative by the subtraction of its energy value from that of the neighbor.

The probabilities of taking each of the eight possible directions are given by the Boltzmann probability distribution function.

$$P[s] = e^{-E(s)/KT} / \sum_A e^{-E(A)/KT}$$

$P[s]$  : Possibility of pointing to a spin direction  $s$  .

$E(s)$  : Energy gap of pointing to a spin direction  $s$  .

$E(A)$  : Energy gap of pointing to all possible eight directions.

$K$  : Boltzmann Constant.

$T$  : Temperature.

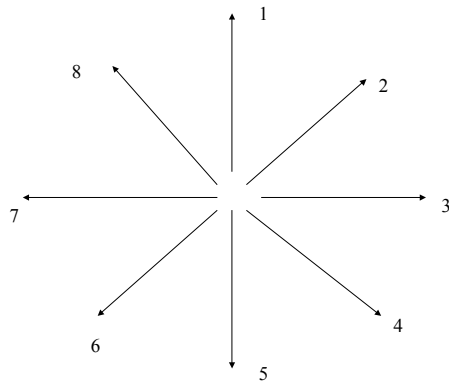


Fig 5.3 Eight possible spin directions of each spin in routing model

Mathematically speaking, pointing to a low energy neighbor always has higher probability compared with to a higher energy neighbor. So how about the macroscopic configuration of the system resulting from local interactions? The key point is the temperature, which expresses the willingness to supply or absorb energy. Under high temperature, heat bath has equal willingness to supply or absorb energy. However, heat bath would rather absorb energy than supply energy under low temperature. In other words, heat bath has difficulty in providing energy needed for hill climbing under low temperature. Thus, most cells in equilibrium state will release energy instead of grab energy and most bonds are at rest. Under high temperature, heat bath has no problem as to provide energy for hill climbing or accepting released energy. Cells in equilibrium state have equal opportunity to be in tense or relaxed state.

If we look at the mathematical expression for such relative probability of state A and state B, it would be:

$$P[A] / P[B] = e^{-D/KT}$$

If the  $KT$  is much larger than the energy gap difference  $D$  between states A and state B, namely  $D$  equals  $(E(A) - E(B))$ . The probability of taking either spin direction A or B would be approximately the same. If the  $KT$  is much lower than  $D$ . The spin direction is far more likely to be in the lower energy state.

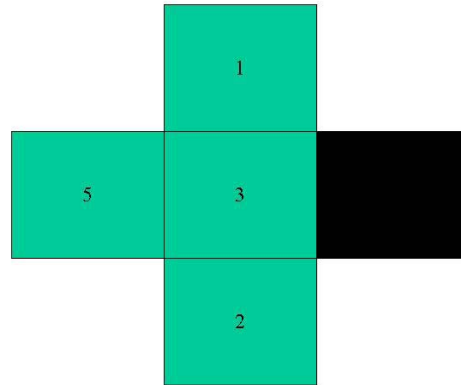


Fig 5.4 Boltzmann probability distribution computation example

We give a simple example as how each node calculates its probabilities for four spin directions. As shown by Fig 5.4, we use von Neumann neighborhood instead of Moor neighborhood to simplify the computation. The node to its east is a wall with infinite energy value and the rest of nodes have the energy value as indicated in the graph.  $N$  represents north,  $E$  represents east,  $S$  represents south and  $W$  represents west.  $K$  is set to be 0.00138,  $T$  is set to be 100.

$$P[N] = e^{-(1-3)/KT} / e^{-(1-3)/KT} + e^{-(5-3)/KT} + e^{-(2-3)/KT} + 0 = 0.999$$

$$P[E] = 0$$

$$P[S] = e^{-(2-3)/KT} / e^{-(1-3)/KT} + e^{-(5-3)/KT} + e^{-(2-3)/KT} + 0 = 7.12 \times 10^{-4}$$

$$P[W] = e^{-(5-3)/KT} / e^{-(1-3)/KT} + e^{-(5-3)/KT} + e^{-(2-3)/KT} + 0 = 2.57 \times 10^{-13}$$

If  $T$  is set to be 1000, we can get the followings:

$$P[N] = e^{-(1-3)/KT} / e^{-(1-3)/KT} + e^{-(5-3)/KT} + e^{-(2-3)/KT} + 0 = 0.649$$

$$P[E] = 0$$

$$P[S] = e^{-(2-3)/KT} / e^{-(1-3)/KT} + e^{-(5-3)/KT} + e^{-(2-3)/KT} + 0 = 0.3147$$

$$P[W] = e^{-(5-3)/KT} / e^{-(1-3)/KT} + e^{-(5-3)/KT} + e^{-(2-3)/KT} + 0 = 0.03578$$

From above data, we can see that at low temperature, nodes have a much higher probabilities to point to a neighbor with low energy value. However, at higher temperature, the probabilities gap is not that significant.

In our program, we use a random number generator to choose spin direction based on distribution probabilities. Both intuitively and mathematically speaking, the system is more likely to find shorter paths to data sink under low temperature compared with high temperature.

### 5.2.2 Damping Effect Theory

Another thing we need to simulate is that under certain critical temperature, system is in freezing state; No matter how much energy difference there is, no spin direction switch would happen. However, there is an increasing stiffness for the spin as the temperature decrease from a certain threshold to freezing point. We introduce damping effect in our program.

When temperature is at or below freezing point, all the spins are frozen and not able to flip their direction in response to potential energy change. There is an increasingly thawing effect with temperature increase. Until reaching certain point of threshold, the spins will be totally free of inhibition. The inhibition effect at low temperature also affects the establishment of potential energy map. At low temperature, cells would be in a kind of lazy state and communication with its neighboring cells slows down.

In analogy to natural life, hibernation in animal is a very common natural metabolism to concede to harsh weather. On the other hand, in case of emergency such as flood or volcano ejection,

hibernation would keep the animals from running away from danger. It is the same thing in our model. When there is a high cell failure probability, damping effect would restrict the cells from acting abnormally. However, in case of topological disturbance, damping effect would impede system's ability to adapt to new environments.

Our program set zero to be the freezing point and threshold can be set in the properties file. So stepping from threshold point down to zero, there is an increasingly stronger inhibition effect on the spins. The damping effect proves to be very effective to counteract high cell failure probability. When cells are more likely to make random decision instead of following the normal Spin Glass logic, it would be a good strategy to restrict their action rather than let them act erroneously when system is subject to high failure probability.

We can generalized four conditions as following:

- a. No environmental disturbance and low failure probability: Damping effect would only result in slow adaptation to environment. No advantages.
- b. No environmental disturbance, high cells failure probability: Damping effect would counteract the high failure probability.
- c. With environmental disturbance, low failure probability: Damping effect would only keep the system from adapting to new environment. We came to the conclusion that in case of environmental disturbance, we should carefully choose the temperature. Not too high lead to a random configuration and not too low not being able to get acquainted with new environment.
- d. With environmental disturbance, high cell failure probability: Damping effect will offset the high failure probability to some extend. However, damping effect can also impede system from adaptation to new environment. It is very hard to find a perfect balance between the two factors.

### 5.2.3 Topological Disturbance

In case of topological disturbance, new potential field needs to be re-established and the scope of the area being affected depends on the nature of the disturbance. Consequently, affected cells will recalculate probability distribution on eight directions to adapt to the potential energy change corresponding to the topological change. Our Spin Glass model can be seen as potential field driven. If there is a neighbor who has lower energy compared with current pointing neighbor, new neighbor' being selected chance will be much higher than that of the current pointing neighbor under low temperature. Chances would be approximately the same under high temperature as we discussed before.

#### a. Close door or place obstacles

Modified cells and all the cells pointing to the modified cells directly and indirectly are potential candidates to be affected by the disturbance. Those cells may or may not be affected by that specific disturbance. Different grid scenarios lead to various intensities of influence. Let's view the original modified cell as the turbulence center, all those cells directly pointing to it are members of the first wave, then all cells pointing to the first wave members are members of the second wave, continue doing this to get a increasingly larger scope of wave. Just like ripple formed by drop of a pebble in the pond.

Ideally, less affected wave members, less disturbance effect. In the worst case, if cells in the far most wave are forced to alter their potential energy, maximum effect will be noticed. Questions would be under what conditions a minor and major change would occur? We have reasons to predict that when close doors and place obstacles at some critical point such as important traffic connection, a significant effect will be noticed. Generally speaking, close doors usually have bigger effect than placing obstacles, since door always serve as critical outlets.

#### b. Open door or remove obstacles

This condition would be different from the last condition discussed above. The overall potential field would try to reach a lower energy configuration under current new environment. Similarly,

if removal of obstacles or opening of doors happen in an rocky area, in other words, an area with high number of wall, obstacles or closed doors, the neighborhood of the changed cell would most likely switch to it as an outlet, then a significant effect would be observed. Otherwise, changes in a spacious area won't cause much difference. It is also a good analogy to imagine that when a collapse happens somewhere in a dam, water will rush out from that outlet. If we just remove a rock in the middle of a river, water will just tenderly run over where that rock was.

#### 5.2.4 Adaptation Phases

There are two dynamic phases during the process of adaptation to environmental disturbances in case of closing doors or placing obstacles. When there are disturbance at certain nodes, in the first phase, the disturbance signal will be propagated to the affected region starting from the disturbed center. All the cells in the affected region would be aware of such disturbance by propagation. It is like a sort of broadcast. I first told all my neighborhood that the shared front door has been closed, then each of my neighborhood would then tell his or her neighborhood about this information, and so on until some neighborhood is reached who knows an alternative way to data sink. Then, second phase starts. The new route is propagated back to the affected area from that neighbor back to the affected area until the all affected neighborhood get the new route.

Speaking of our Spin Glass model, the potential energy of each affected cells increases by one for each generation (step) ever since it starts to increase. The elevation process stops until the back propagation reaches it. So the total number of generations need to adapt to the environmental disturbance is exactly identical to the biggest increase in the potential energy of the cell in the region. Let's illustrate it by a simple 8x9 grid example in Fig 5.5.

Suppose a cell at (4,1) which was open door, is closed. In order to illustrate how the two phases work in the adaptation to the disturbance. We use different colors to represent different waves. The first wave is blue with potential energy 4; the second wave is green with potential energy 5, such and such. Note that it is based on the original grid configuration. The Table 5.2 illustrates the potential energy change of various waves at each step.

2	1	1	1	2	3	4	5	6
2	1	Data-sink	1	2	3	Wall	5	6
2	1	1	1	2	3	Wall	6	6
2	2	2	2	2	3	Wall	7	7
Wall	3 closed	Wall	Wall	Wall	Wall	Wall	8	8
4	4	4	5	6	7	8	9	9
5	5	5	5	6	7	Wall	Wall	Wall
6	6	Wall	Wall	7	7	8	9	10

Fig 5.5 Potential field of an example grid

After step5, system enters into the second phase. Back propagation. Cell at (5,6) with potential energy 9 is reached and it stops growing its potential value. Then, start from this cell, back propagation begins. The members of the different waves are reorganized according to the new grid configuration. The members of a specific wave will stop growing its potential value, once the back propagation reaches it. So we got the final potential energy shown in Fig 5.6 after generation 11.

We also use different color to represent rearranged wave members. This back propagation phase starts from step6 to setp11. The total number of generation to adapt to the environmental disturbance is 11. Which is exactly identical to the biggest energy difference in the affected region. And it is  $15-4=11$ .

Table 5.2 Potential energy values of different waves versus steps

Potential energy	Step0	Step1	Step2	Step3	Step4	Step5
First Wave	4	5	6	7	8	9
Second wave	5	5	6	7	8	9
Third wave	6	6	6	7	8	9
Fourth wave	7	7	7	7	8	9
Fifth wave	8	8	8	8	8	9

2	1	1	1	2	3	4	5	6
2	1	Data-sink	1	2	3	Wall	5	6
2	1	1	1	2	3	Wall	6	6
2	2	2	2	2	3	Wall	7	7
Wall	closed	Wall	Wall	Wall	Wall	Wall	8	8
15	14	13	12	11	10	9	9	9
15	14	13	12	11	10	Wall	Wall	Wall
15	14	Wall	Wall	11	11	11	12	13

Fig 5.6 Final potential energy map

Once the potential energy map is stabilized, the spin direction can be stabilized as well. In case of open a door or remove an obstacle, there is only one adaptation phase starting from the disturbance cell, until a new potential field is established.

### 5.3 Result Analysis

Take the 16x16 grid as an example; we study the simulation result with different parameters. Adaptation curve is a figure with generation number versus average Hops at that generation, which is sum of hops divided by the number of cells.

#### 5.3.1 Temperature Effect

From Fig5.7, we can see that the average hop is much lower at low temperature compared with high temperature. This confirms our theory that low temperature leads to better routes to data sink, since the majority of cell bonds are at rest at equilibrium, pointing to neighbors with low potential value. This kind of local order creates a macro magnetization and this macro orientation further leads to overall routes to the data sink. Under high temperature, proportion of bonds at rest and at tense is at the balance. This local random results in minimized magnetization and lack of overall orientation gives much longer or even no routes to the data sink.

#### 5.3.2 Cell Failure Probability

Fig 5.8 shows that when various levels of cell failure probabilities are introduced, how system adapts with that level of failure probability at low temperature. A cell wills random select a spin direction opposing to follow the usual Spin Glass logic. It simulates the erroneous rate of sensor nodes in a sensor network. Results show that our system can tolerate no more than 5% of failure probabilities. It means that we can tolerate five percent sensor nodes to act erroneously and still be able to discover routes to data sink at a satisfactory level. In a real battlefield, cell failure probability may vary greatly. It is our hope that tolerant ability of Spin Glass model can help to damp some of the oscillation.

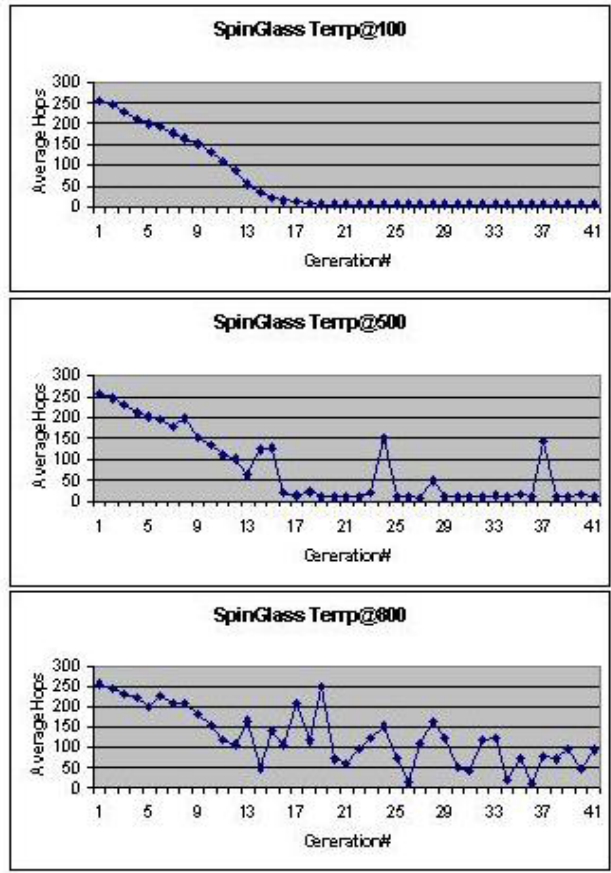


Fig 5.7 Adaptation curve with various temperatures (e.g. 100, 500, 800)

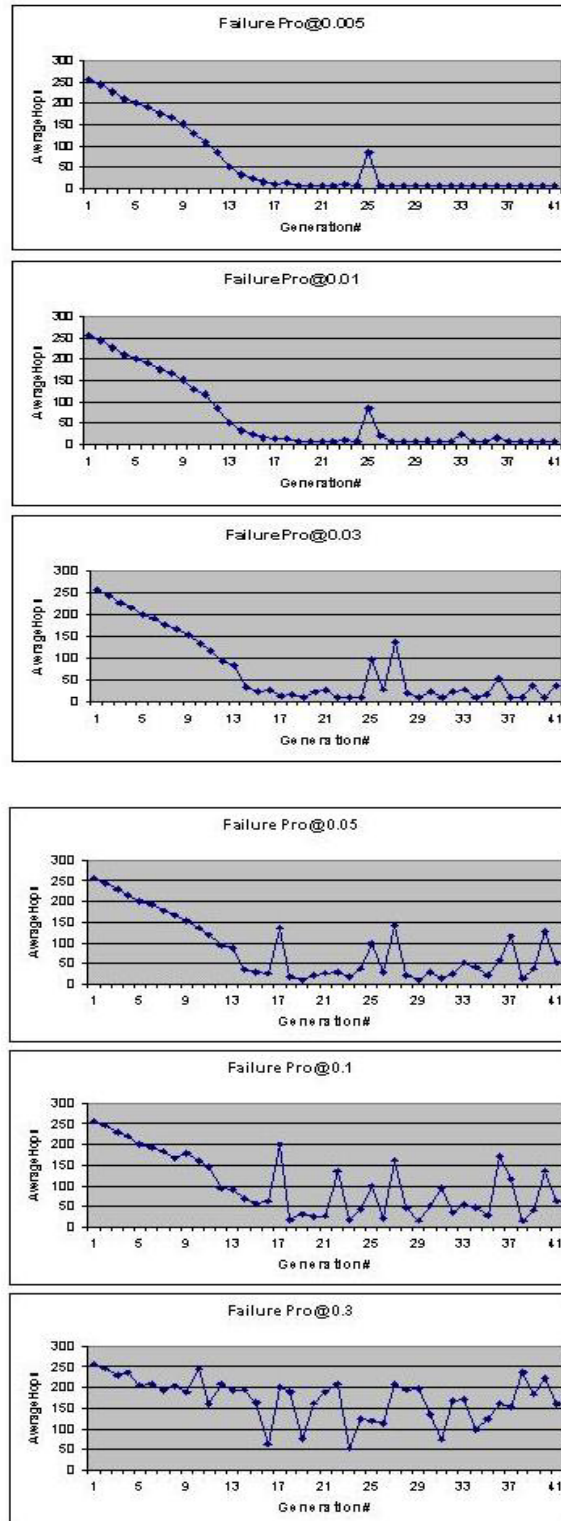


Fig 5.8 Failure probabilities at 0.005, 0.01,0.03,0.05,0.1 and 0.3

### 5.3.3 Adaptation to Disturbance at Different Temperatures

System can adapt to topological disturbance introduced at any generation, high temperature would hinder the process of adaptation. Fig5.9 shows how system adapts to four disturbances introduced at generation 25, 30, 35 and 40. Three temperatures are set at 100, 500 and 800. The total number of generation is set to be 50. We can see that under low temperature 100, there is a steady decrease in the average hops. The higher the temperature is, the harder for the system to find efficient routing routes.

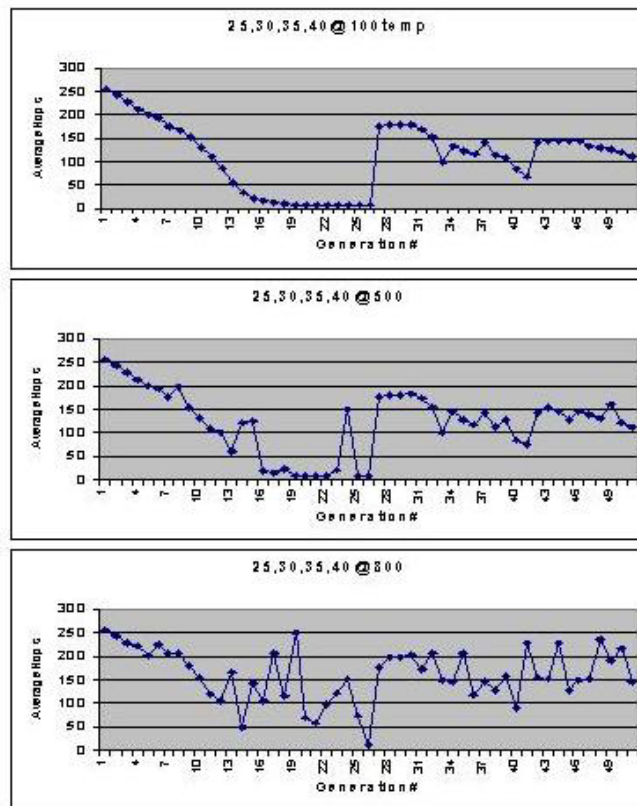


Fig 5.9 Adaptation to disturbance at various temperatures

### 5.3.4 Disturbance Locations and Energy Map

We show the energy maps after closing doors at two different locations. Both the adaptation curve and the energy map show that if there is significant color distribution change in energy map, adaptation curve tends to have more fluctuation. Closing door at (10,7) has comparable

smaller effect on the energy map than closing door at (2,6). Energy map offers us another aspect to look at the disturbance effect.

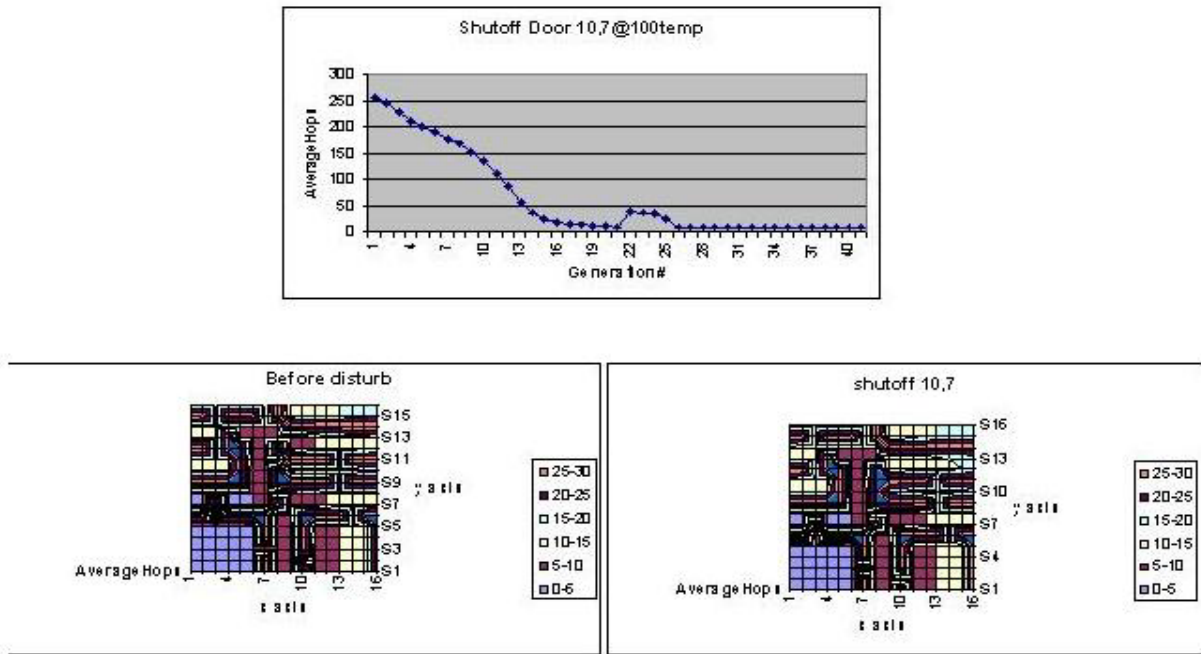


Fig 5.10 Adaptation curve and energy map when close door at 10,7

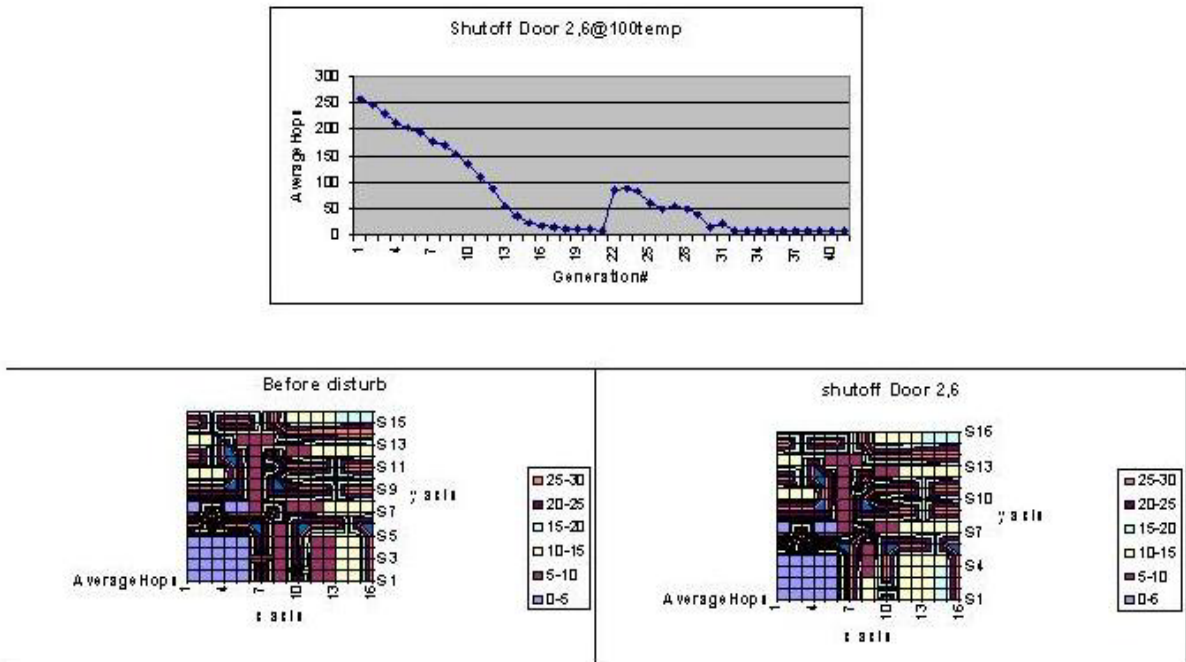


Fig 5.11 Adaptation curve and energy map when close door at 2,6

### 5.3.5 Damping Effect

We study the damping effect on the system under various conditions. Results show that damping effect is effective in offsetting the high failure probabilities but will slow down the process of adapting to topological change.

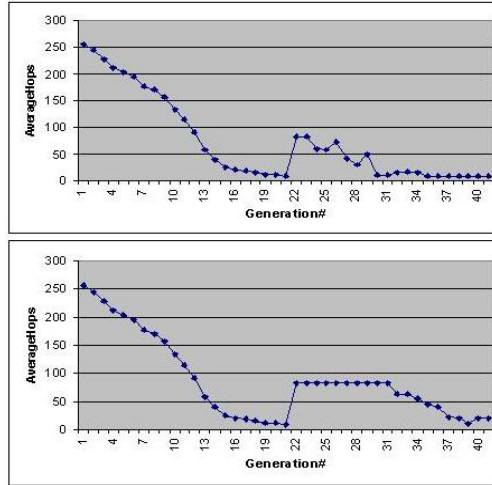


Fig 5.12 Disturbance is introduced at gen20

Upper is without damping effect. Temperature is set to be 100 all the time.  
Lower is with damping effect from gen20 to gen30. Temperature is set at 5.

Fig 5.12 shows that in case of disturbance and low failure probability, damping effect can only result in impeded adaptation. It is obvious that the upper without damping effect has a better performance than the lower one with damping effect.

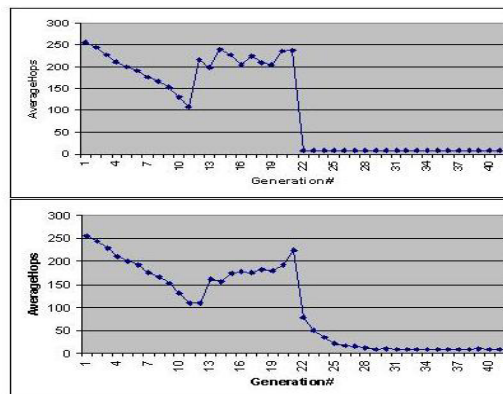


Fig 5.13 No disturbance introduced and high failure probability from gen10 to 20

Upper: Temperature is set to be 100 through the whole process without damping effect.  
Lower: Temperature is set to be 5 from gen10 to gen20 with damping effect.

Fig 5.13 shows the system under high failure probability during generation 10 to generation 20. No disturbance is introduced. Will damping effect help this time? The upper figure shows the system without damping effect. The lower figure shows the system with damping effect during the high failure probability. Results show that damping effect can refrain the system from acting to high failure probability. Performance is better during the high failure probability period. We also noticed that there is a sharp drop for the system without damping effect once out of the high failure time window. However, system slowly improves its performance with damping effect. It is due to the damping effect on the potential field re-establishment. Communication among cells also slows down. So, the build up of a new potential field lags behind with damping effect.

### 5.3.6 Placing Obstacles

Placing obstacles usually has mild effect on the system as shown by Fig5.14, but occasionally it is not the case. Let's illustrate why it is the case by following two examples.

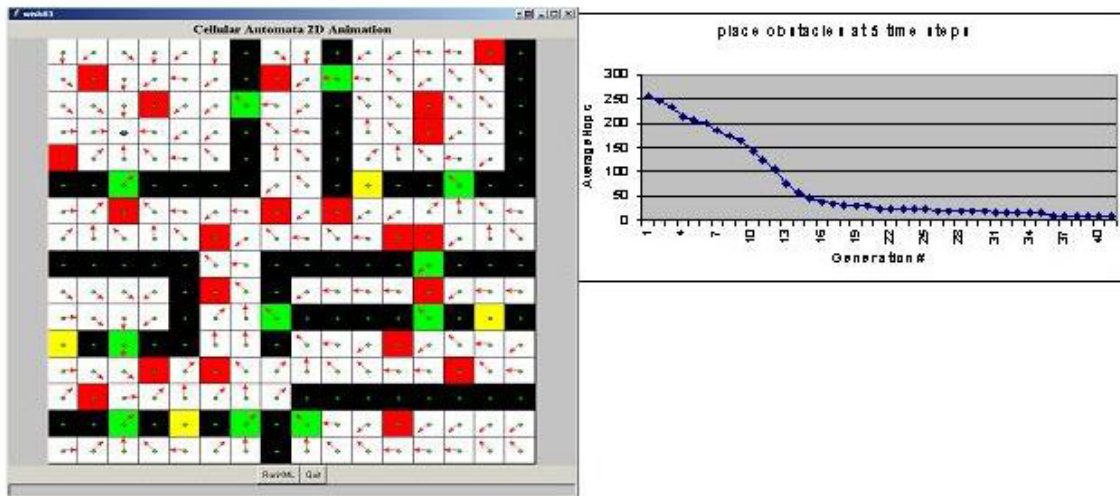


Fig 5.14 Placing a large number of obstacles in sequence

At gen15: 35,60,117  
 At gen20: 149,209,235,23  
 At gen25: 123,124,44,187  
 At gen30: 105,197  
 At gen35: 14,17,64,156,205

Total placed 18 obstacles. We still couldn't see much overall effect. We assume that the position where the obstacles are placed is the key point. We do another test to confirm our assumption.

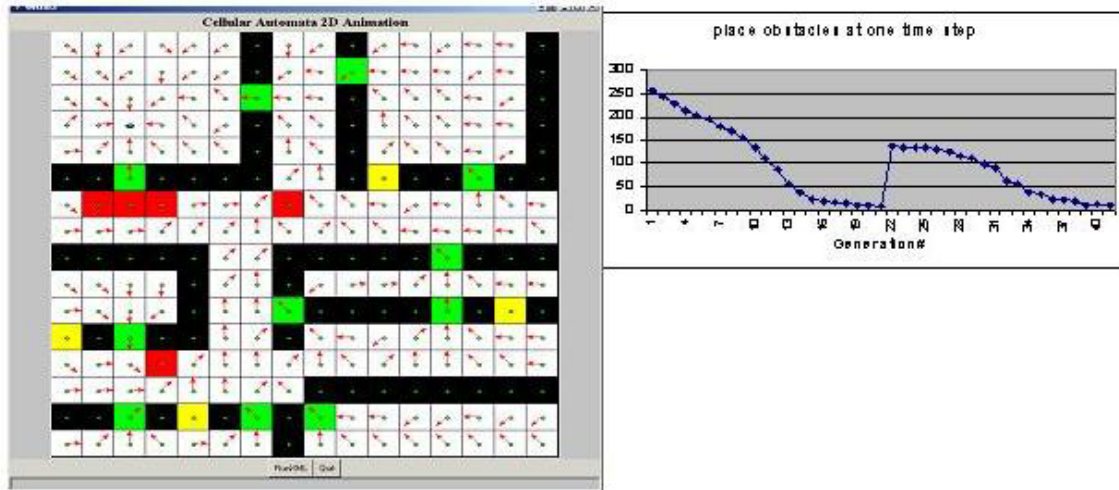


Fig 5.15 Placing two obstacles

Two obstacles are placed at gen20 shown in Fig 5.15. Although only two obstacles are placed, there is a significant turbulence in adaptation curve compared with that of the previous one. It demonstrates that not the number of obstacles but the position where the obstacles play the key role.

Above simulation results show that our Spin Glass routing model displays good ability to find routing paths to data sink with good simulation to WSN scenario. It also tolerates topological disturbance very well, which is a key point in a highly dynamic and ad hoc WSN. However, a large amount of memory is required to maintain such algorithm and local communication overhead is considerably high. In the next chapter, we will discuss another routing model inspired by a Chemistry phenomenon, Diffusion Limited Aggregation (DLA).

## Chapter 6

### Multi-fractal Model

#### 6.1 Multi-fractal Background

People observed that many physical quantities don't obey conventional scaling laws. First introduced in turbulence by B.B Mandelbrot, then this phenomenon is extended to many general cases, such as DLA (Diffusion Limited Aggregation) pattern analysis, population distribution, internet data traffic modeling. A single exponent cannot characterize the moment of the distribution, however, an infinite hierarchy of exponent is needed [8].

What kinds of processes would produce multi-fractal structures? Study shows that multiplicative iteration of random processes creates multi-fractal structure, while additive processes generate mono-fractals. Multi-fractals can be defined as mathematical generalizations of fractals, objects displaying "fractional dimension", "scale invariance" and "self-similarity"[7]. Mono-fractal can be characterized by a single exponent but Multi-fractal can only be characterized by a spectrum of exponents, namely infinite hierarchy of exponents. People usually call the exponent index to be fractal dimension.

##### 6.1.1 Fractal Dimensions

There are many kinds of geometry, such as Euclidian geometry, Cartesian geometry Spherical geometry, Turtle geometry and fractal geometry, which allows some objects to have non-integer dimension values. Speaking of topological dimensions, a line has one-dimension; a plane has two dimensions; a cube has three dimensions. Fractal geometry has not been studies until late 1970s. It mainly emphasized on the importance of nested shapes that contain arbitrarily intricate patterns, and such pattern is quite common in nature [16]. In fractal geometry, dimension is used to quantify the texture of surfaces or index the measurement of the complexity of geometric object,

which has no standard shapes, such as cloud, tree and river. Fractal dimension is always greater than or equal to the topological dimension.

Also, fractal dimension is one of the basic concepts for characterizing fractals with scaling symmetry, which is another notion for self-similarity. F.Hausdorff first proposed the non-integral dimension. However, box-dimensions are much more practical to work with. For example, the capacity-dimension  $D_c$  introduced by A.N.Kolmogorov, defines a partition of the fractal into equally sized cubes with edge size  $\varepsilon$ .

$$D_c = \lim_{\varepsilon \rightarrow 0} \frac{\ln N(\varepsilon)}{\ln 1/\varepsilon} \quad N(\varepsilon) : \text{Minimum number of cubes for covering the set.}$$

Information-dimension  $D_I$  is defined as:

$$D_I = \lim_{\varepsilon \rightarrow 0} \frac{-P_i(\varepsilon) \ln P_i(\varepsilon)}{\ln 1/\varepsilon} \quad P_i(\varepsilon) : \text{Probability of finding a point of the fractal in the } i\text{-th cube}$$

of size  $\varepsilon$ .

The correlation-dimension  $D_K$  is defined as:

$$D_K = \lim_{\varepsilon \rightarrow 0} \frac{-\ln C(\varepsilon)}{\ln 1/\varepsilon}$$

$$\text{With the correlation integral } C(\varepsilon) = \int_{|r| < \varepsilon} d^d r c(r)$$

All those dimensions may be different for one fractal. In the case of mono-fractal, they are identical. In case of multi-fractal, a spectrum of dimensions (infinite hierarchical dimension representation) is needed for description [8].

### 6.1.2 Self Similarity

Self-similarity means looks the same at different zoom levels. A fractal can be subdivided into parts, which at least approximately resemble the whole. It is also called nested structure. A tree can be thought of as a fractal since if you look at the tree as a whole, you see trunk, with branches

coming out of it. Then if you look at a smaller portion of it, say a branch, you see a similar thing, namely, a stick with twigs coming out of it.

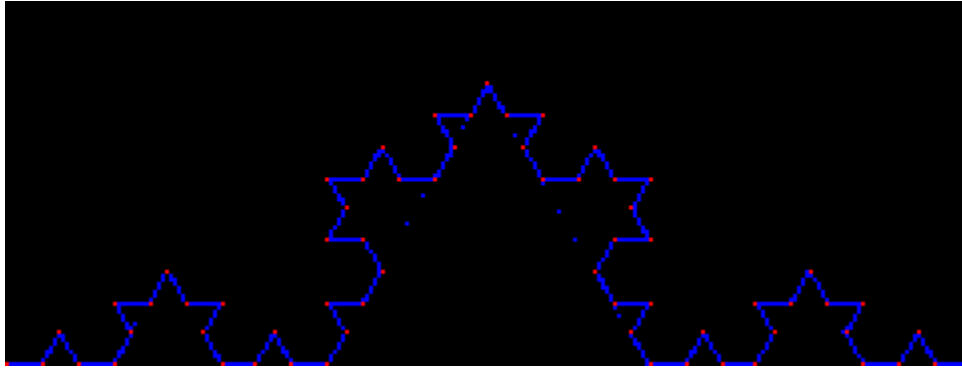


Fig 6.1 Koch curve ( generated by Netlogo1.1)

Helge von Koch, a Swedish mathematician, introduced what is now called the Koch curve in 1904. It first begins with a straight line, and then divides the line into three equal parts. The middle third is replaced by an equilateral triangle and the base is taken away. We repeat the same thing to each of the four remaining lines. We would get the above shape at the third iteration. This fractal has infinite perimeter and it is also self-similar.

## 6.2. Diffusion Limited Aggregation Fractal

If you still can recall the electrophoresis experiment in Chemistry, we have a solution of iron sulfate. The positive ions are attracted to the cathode and the negative ions are going to the anode. The positive iron ions eventually reach the cathode and become iron molecules upon contact after grabbing extra electrons. Many iron ions continue to deposited iron molecules and finally build up a layer of iron on the electrode. The texture complexity of the iron layer, namely the fractal dimension is affected by many factors, such as concentration of the solution, the voltage on the electrode etc [9].

This classic crystal-growing prototype for gas and fluid is called DLA, first introduced by Witten and Sander in the early 1980s. Beginning with one or more foreign seeds, upon contact with the seeds and satisfy certain crystallization conditions, wandering gas or fluid particles become solidified in a certain way. Fig 6.2 and Fig 6.3 illustrate a growing example starting from a single seed.

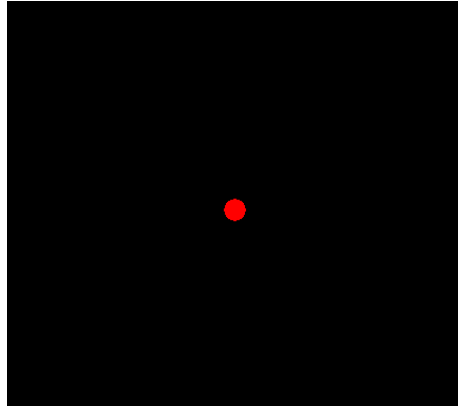


Fig 6.2 Initial single seed of DLA

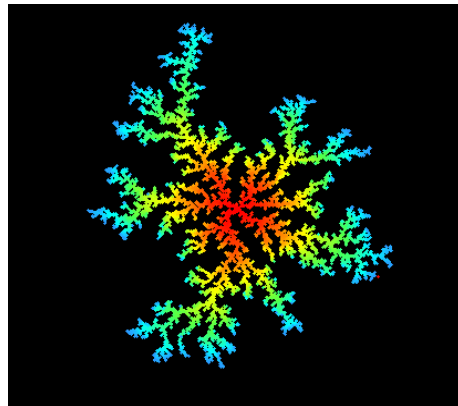


Fig 6.3 DLA growing tree (generated by Netlogo 1.1 )

## 6.3 Multi-fractal Routing

### 6.3.1 Multi-fractal Routing Theory

Data sink is set to be single seed. A routing tree will grow starting from the seed. One major alteration from DLA is that instead of letting particles randomly float in the grid, we fix each particle, namely a free cell in our model in a certain position with limited vibration. A free cell has the possibility to join the routing tree only if the tree stretches out to its neighborhood.

However, if nodes were allowed to join the tree unequivocally, the tree would grow too fast in a wild manner. In crystallization, there is crystallization growth inhibition exerted by the crystallization site to the nearby particles. This inhibition can be physically explained by interfacial surface tension and latent heat diffusion effects. When a particle becomes crystallized, heat is released and the heat will inhibit the crystallization of nearby particles. In snowflake CA, if and only if one of the hexagons neighbors is crystallized, that site can be crystallized. However, we can't directly apply this rule to our Multi-fractal routing tree growth. The reason is that we need to find route to the data sink for every free cell. Then, we need to specify a set of probabilities of joining the routing tree based on the number of neighbors in the routing tree, ranging from 1 to 8 for Moore neighborhood. And this is much as we do in the DLA to set a concentration threshold for joining the tree to control the sediment speed and structure. Usually, the more neighbors in the routing tree, the less likely for the node to join in the routing tree. In case of more than one neighbor in tree for a node to pick, we would select the one with the lowest hops to the data sink. As time proceeds, more and more nodes would solidify themselves somewhere in the routing tree, which stems from the data sink.

In case of topological disturbance, if certain nodes of routing tree were cut off, say an open door was closed or additional obstacles were placed. Those branches related to that node will be torn off the routing tree, and subsequent new branches will grow from the ripping points.

The Multi-fractal approach is less robust to topological disturbance than Spin Glass model, since the adaptive mode of Multi-fractal is based on a go and fix type. The overall routing tree will not

be demolished in response to topological change. Only pathological branch will be cut off, the adaptation is not based on an overall consideration. The final routing tree structure would depend on the sequence of topological disturbances. However, efficiency is obtained by saving computation for already stabilized cells.

### 6.3.2 Properties of Inhibition Curve

One key point I omitted is how exactly the crystallization inhibition effect intensifies as the number of neighbors in the routing tree increases. We first investigate the very simple linear one. An initial condition, the probability  $P$  for only one neighbor in the tree needs to be set first. Let's start from one. In other words, as long as there is one neighbor in routing tree, join in the tree unequivocally. If all its neighbors are in the routing tree, stay out of the tree. We use  $P[1]$  to represent the probability for one neighbor in routing tree and so on. How about when there are only at most six neighbors that possibly could be in a routing tree for example?  $P[6]$  should be zero instead of  $P[8]$ . Squash the line to try to maintain the same area under the curve by multiplying each point below six a certain factors.

#### a. Linear Function:

Fig 6.4 shows the example of a linear function together with an adjusted curve with six neighbors.

$$kx + b = y \quad x \in [0,8]$$

Suppose  $x = 1, y = p$  and  $x = 8, y = 0$ , by solving the equations,

$$k + b = p$$

$$8k + b = 0$$

$$b = 8p/7 \quad \text{and} \quad k = -p/7$$

#### b. Quadratic Function:

Fig 6.5 shows the quadratic curve together with a squashed curve with six neighbors.

$$a(x - b)^2 = y \quad x \in [0,8]$$

Suppose  $x = 1, y = p$  and  $x = 8, y = 0$ ,

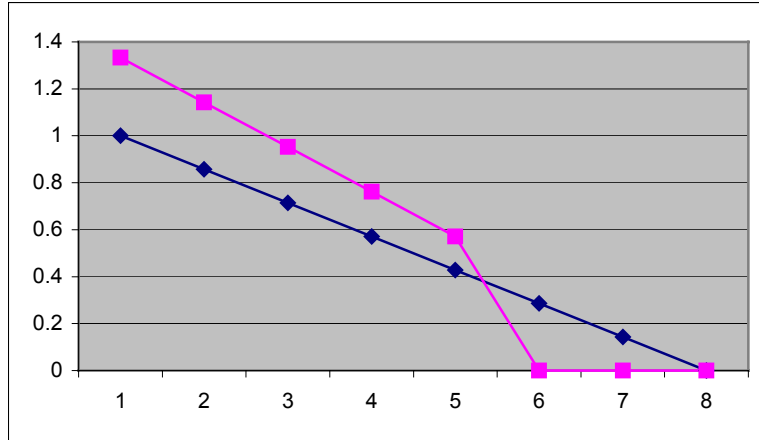


Fig 6.4 Linear inhibition curve with  $P$  to be 1

We only need to set the value of  $p$  to get the exact function. In our program, we first set eight probabilities value from  $P[1]$  to  $P[8]$ , which is for sensor nodes with eight neighbors possibly in a routing tree. For those nodes with only  $n$  neighbors, a multiplier, as  $8/n$  will be applied to the probabilities value set range from  $P[1]$  to  $P[n-1]$ .

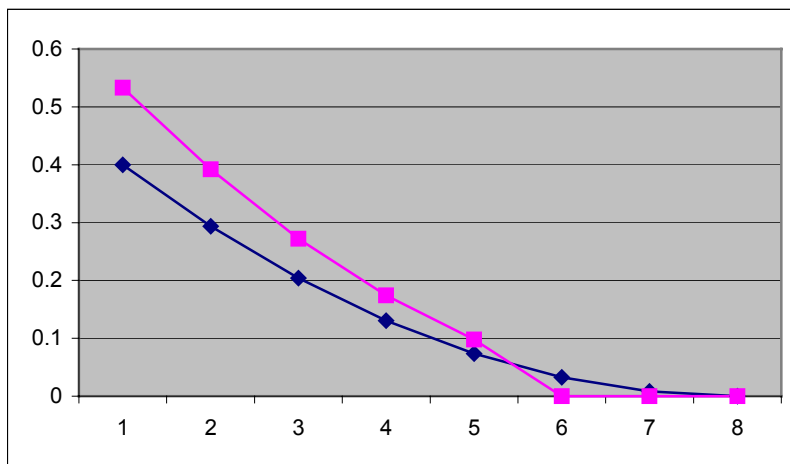


Fig 6.5 Quadratic inhibition curve with  $P$  to be 0.4

### 6.3.3 Weight Factor for Cells

We encountered some cases when many cells at one side of an open door are already in the routing tree, routing tree would have problem to stretch out through the door to the other side of the door. One resolution of increasing probabilities values would favor a dense routing tree, and it

is certainly opposed to our intention. On second thoughts, since doors always serve as critical connections, priority to open door in tree growing seems reasonable. Differentiation between free cells and open doors by pose weight factor to probabilities set is a simple and straightforward solution. For example, we have probabilities set as following:

```
<JoinTree0>0 </JoinTree0>  
<JoinTree1>0.4</JoinTree1>  
<JoinTree2>0.2</JoinTree2>  
<JoinTree3>0.1</JoinTree3>  
<JoinTree4>0</JoinTree4>  
<JoinTree5>0</JoinTree5>  
<JoinTree6>0</JoinTree6>  
<JoinTree7>0</JoinTree7>  
<JoinTree8>0</JoinTree8>
```

For example, the weight factor for free cells and open doors are 0.4 and 1.2 respectively. The probabilities set for free cells would be

```
<JoinTree0>0 </JoinTree0>  
<JoinTree1>0.16</JoinTree1>  
<JoinTree2>0.08</JoinTree2>  
<JoinTree3>0.04</JoinTree3>  
<JoinTree4>0</JoinTree4>  
<JoinTree5>0</JoinTree5>  
<JoinTree6>0</JoinTree6>  
<JoinTree7>0</JoinTree7>  
<JoinTree8>0</JoinTree8>
```

By multiplying the probabilities value by the weight factor, we can get the probabilities set for free cells and open doors. And in this case, open doors have three times higher probabilities to join the tree compared with free cells. This simple weighting factor can alleviate the above open door problem and still help to maintain a sparse tree.

## 6.4 Result Analysis

### 6.4.1 Snowflake Like Crystallization Approach

Crystallization is allowed when only one neighbor out of hexagonal neighbors is already in the routing tree for snowflake formation. If we transplant this idea into our routing tree growth, result shows that the routing tree would almost immediately stop growing, since no more cells with only one neighbor in the routing tree can be found. Fig6.6 shows the simulation result with snowflake

logic. In our case, we have the Moore neighborhood type. If we use von Neumann neighborhood type, more cells can join to the routing tree before reaching the dead end.

The corresponding probabilities values are as following:

```
<JoinTree0>0</JoinTree0>  
<JoinTree1>1</JoinTree1>  
<JoinTree2>0</JoinTree2>  
<JoinTree3>0</JoinTree3>  
<JoinTree4>0</JoinTree4>  
<JoinTree5>0</JoinTree5>  
<JoinTree6>0</JoinTree6>  
<JoinTree7>0</JoinTree7>  
<JoinTree8>0</JoinTree8>
```

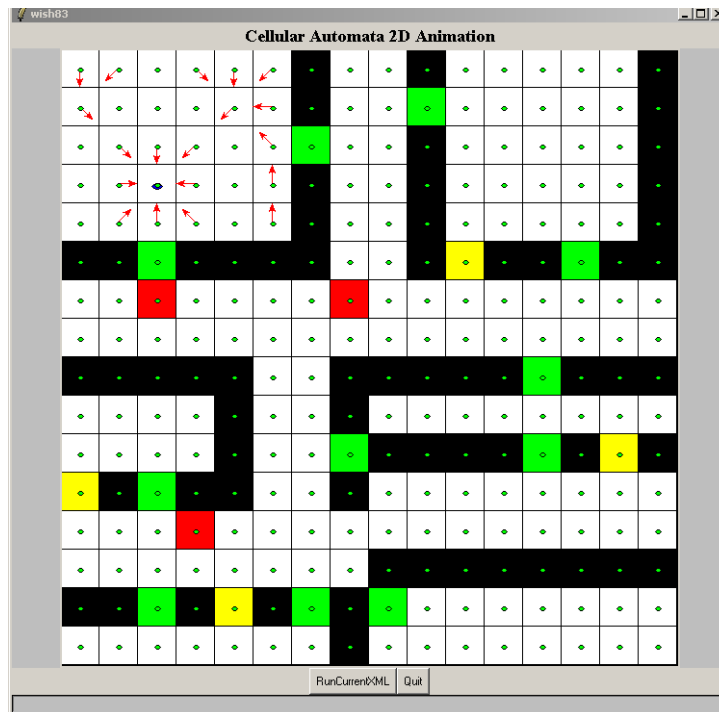


Fig 6.6 Tree stop growing with snowflake logic

#### 6.4.2 All One Probabilities

By adjusting the joining tree probabilities, we can control the growing speed and structure of the routing tree. If we ignore the growth inhibition effect, we reset the probabilities set to be all 1:

```

<JoinTree0>0</JoinTree0>
<JoinTree1>1</JoinTree1>
<JoinTree2>1</JoinTree2>
<JoinTree3>1</JoinTree3>
<JoinTree4>1</JoinTree4>
<JoinTree5>1</JoinTree5>
<JoinTree6>1</JoinTree6>
<JoinTree7>1</JoinTree7>
<JoinTree8>1</JoinTree8>

```

Fig 6.7 shows the routing tree at generation 30 with all probabilities to be one. The routing tree has a better performance than the previous one with lower average hops. However, the maximally dense routing tree is not what we want.

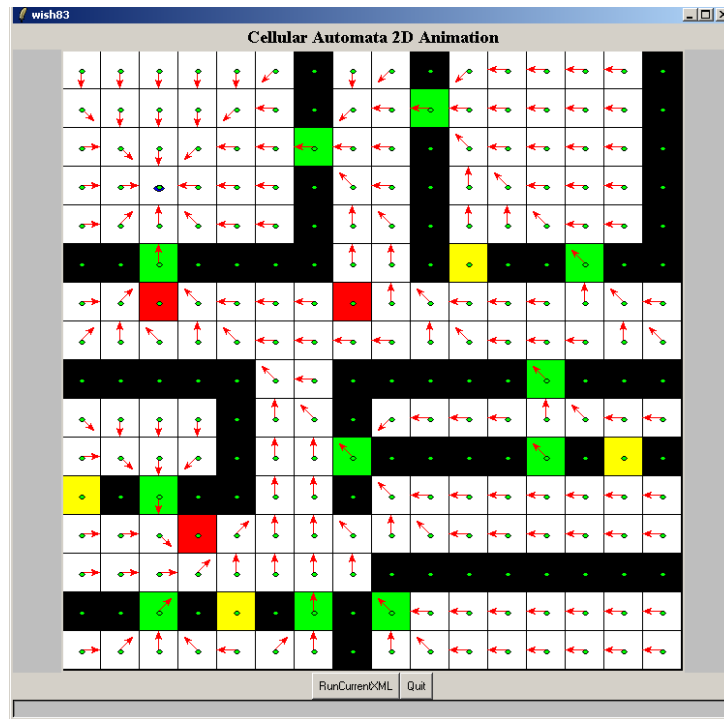


Fig 6.7 Tree with all 1 probabilities

### 6.4.3 Different Inhibition Curves

#### 6.4.3.1 Linear Curve

```

<JoinTree0>0</JoinTree0>
<JoinTree1>0.3</JoinTree1>
<JoinTree2>0.25713</JoinTree2>
<JoinTree3>0.21429</JoinTree3>
<JoinTree4>0.17142</JoinTree4>
<JoinTree5>0.12858</JoinTree5>

```

```

<JoinTree6>0.08571</JoinTree6>
<JoinTree7>0.04287</JoinTree7>
<JoinTree8>0</JoinTree8>
<OpendoorWeight>3</OpendoorWeight>
<FreecellWeight>0.35</FreecellWeight>

```

We fix the weight factor for open door to be 3 and only the weight factor for free cell is modified.

The following simulation results are all run with weight factor for open door to be 3.

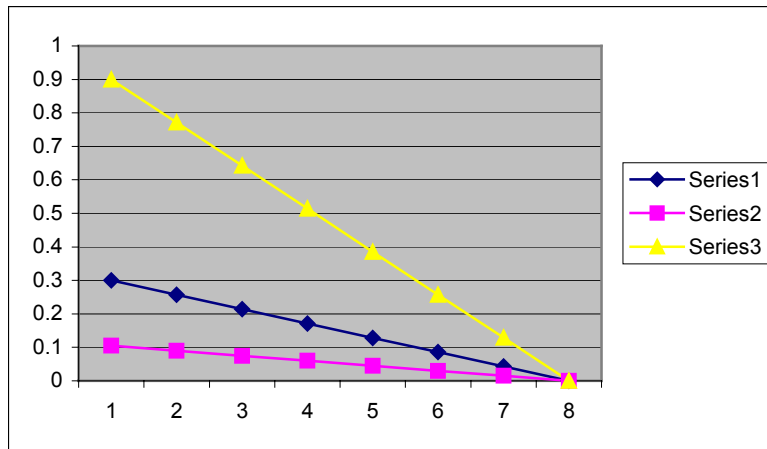


Fig 6.8 Linear inhibition curve with weight factors for free cell and open door

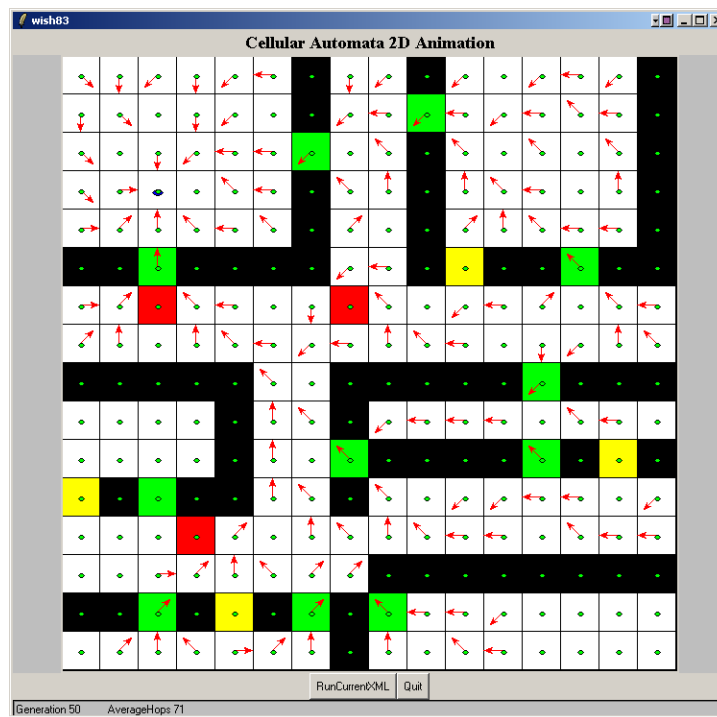


Fig 6.9 Tree with linear curve with weight factor 0.35 at gen50

The routing tree with above probabilities set is still too dense. We tried to decrease the weight factor for free cell to 0.3. A new routing tree is shown in Fig 6.10. We noticed that this routing tree huddled together in the upper grid. The retarded growth and the high density of the routing tree demands further adjustments.

We increased the number of generation to 70 and further decreased the weight factor for free cells to 0.2. We got a new routing tree in Fig 6.11. Although the system took 70 generations to grow such a routing tree, the tree still couldn't span most area of the grid. Bearing this problem in minds, we switch attention to quadratic inhibition curve and see how it can affect the pattern of the routing tree.

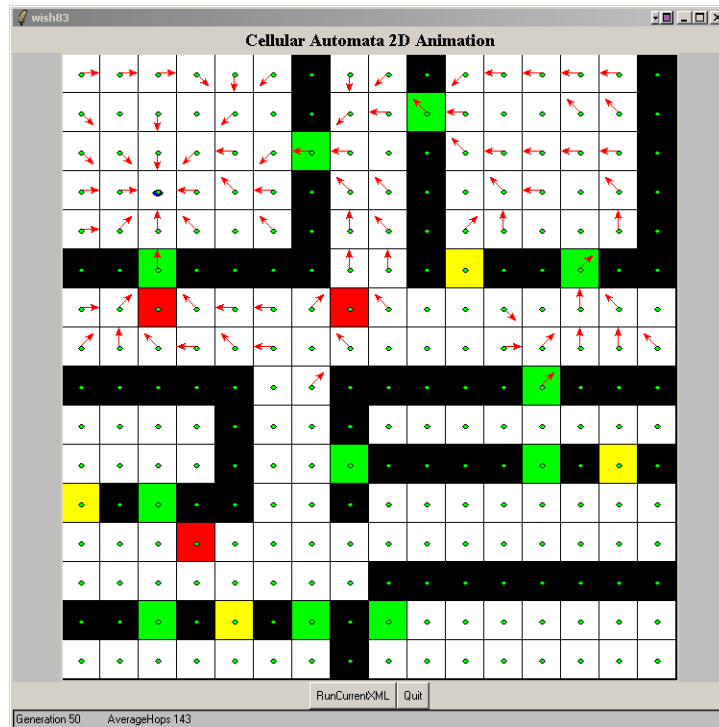


Fig 6.10 Tree with linear curve with weight factor 0.3 at gen50

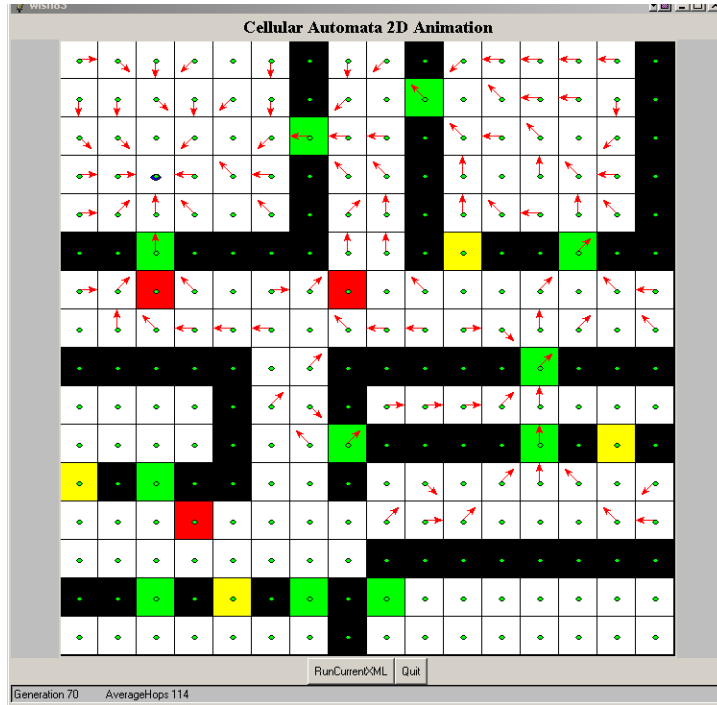


Fig 6.11 Tree with linear curve with weight factor 0.2 at gen70

#### 6.4.3.2 Quadratic Curve

Instead, we tried the quadratic inhibition curve.

```

<JoinTree0>0</JoinTree0>
<JoinTree1>0.3</JoinTree1>
<JoinTree2>0.2204</JoinTree2>
<JoinTree3>0.15306</JoinTree3>
<JoinTree4>0.09796</JoinTree4>
<JoinTree5>0.0551</JoinTree5>
<JoinTree6>0.02449</JoinTree6>
<JoinTree7>0.006122</JoinTree7>
<JoinTree8>0</JoinTree8>
<OpendoorWeight>3</OpendoorWeight>
<FreecellWeight>0.35</FreecellWeight>

```

The routing tree with quadratic inhibition curve in Fig 6.13 is still a little bit too dense. We decreased the weight factor for free cell to 0.3. Then we got a new routing tree as shown in Fig 6.14 with decreased weight factor for free cells. In order to get an even sparser tree, which can span most of the grid, we increase the number of generations to 70 and decrease the free cell weight factor to 0.2. New tree shown in Fig 6.15.

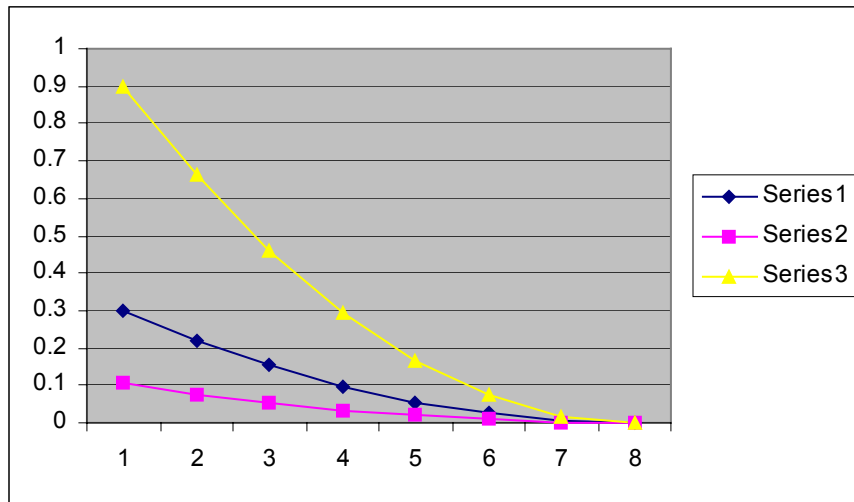


Fig 6.12 Quadratic inhibition curve with weight factors for free cell and open door

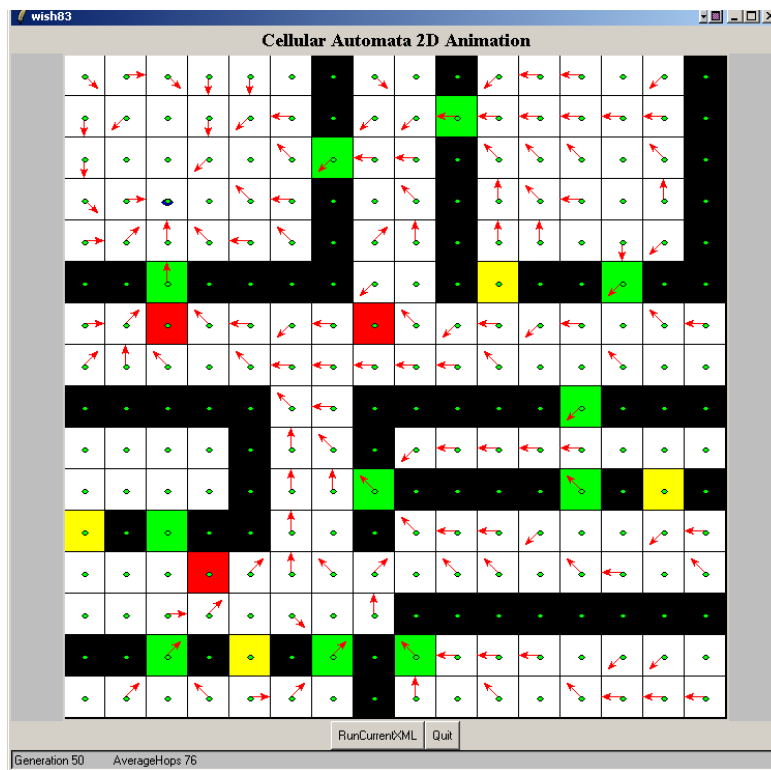


Fig.6.13 Tree with quadratic curve at gen50

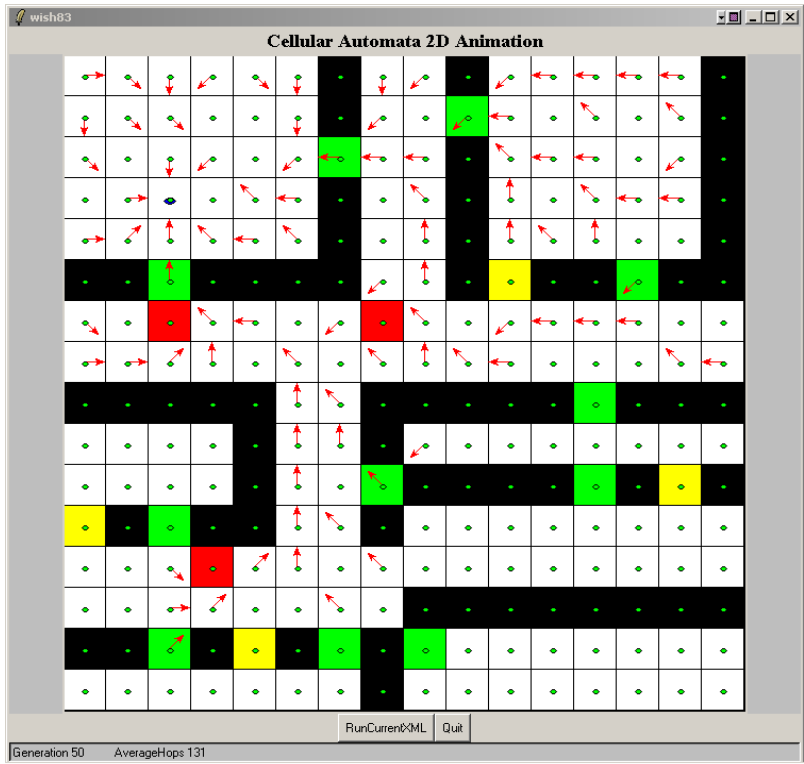


Fig 6.14 Tree with quadratic curve with weight factor 0.3 at gen50

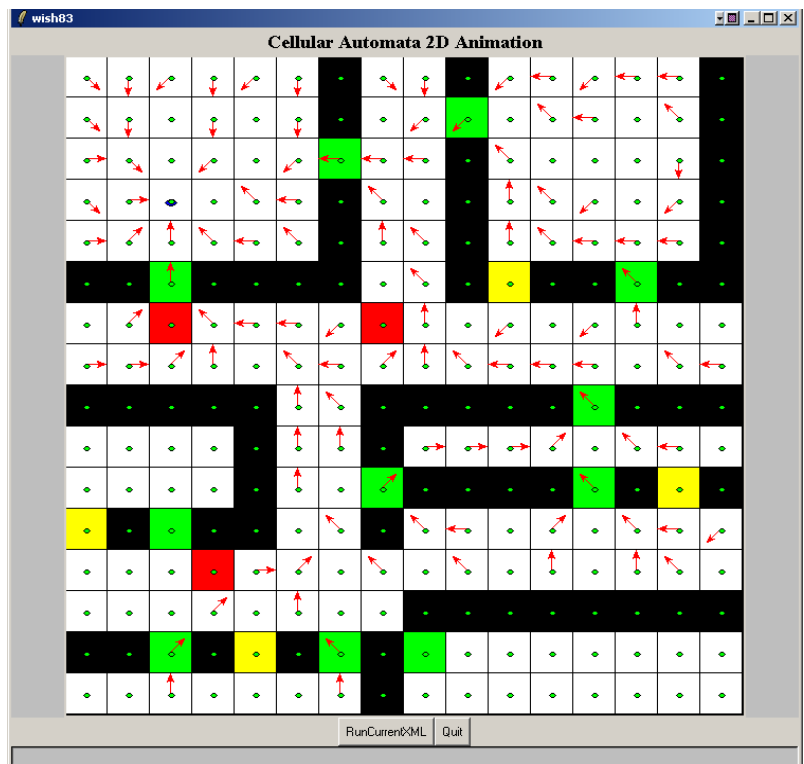


Fig6.15 Tree with quadratic curve with weight factor 0.2 at gen70

#### 6.4.4 Adaptation to Topological Disturbance

Probabilities set are:

```
<JoinTree0>0 </JoinTree0>  
<JoinTree1>0.4</JoinTree1>  
<JoinTree2>0.2</JoinTree2>  
<JoinTree3>0.1</JoinTree3>  
<JoinTree4>0.08</JoinTree4>  
<JoinTree5>0.04</JoinTree5>  
<JoinTree6>0.02</JoinTree6>  
<JoinTree7>0.01</JoinTree7>  
<JoinTree8>0.005</JoinTree8>
```

Two disturbances are introduced at generation 20 and generation 40. At generation 20, doors at locations 2,6 and 1,9 are closed. At generation 40, door at location 8,12 is closed. And door at 1,9 is reopened. Three more obstacles are added as shown with one old obstacle removed.

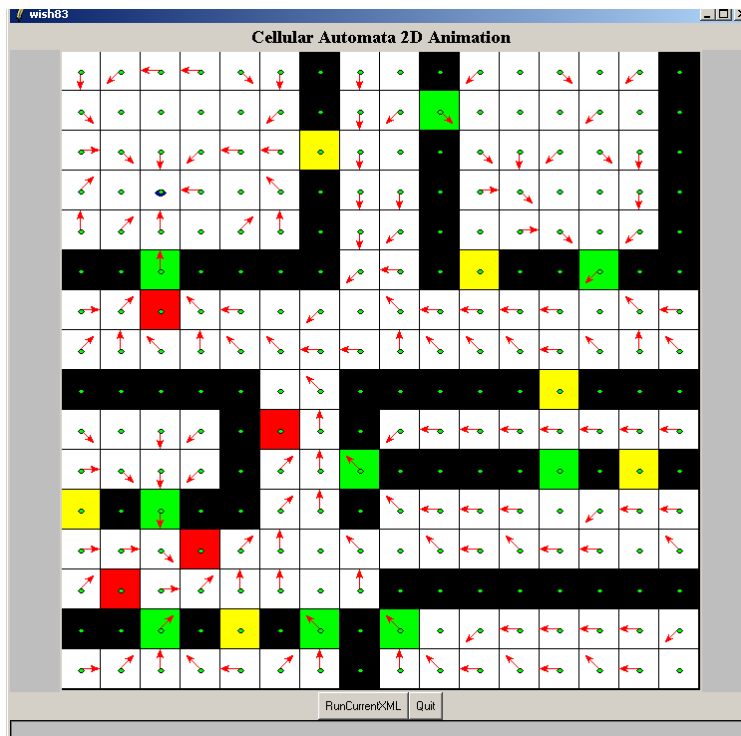


Fig 6.16 Final routing tree at gen60

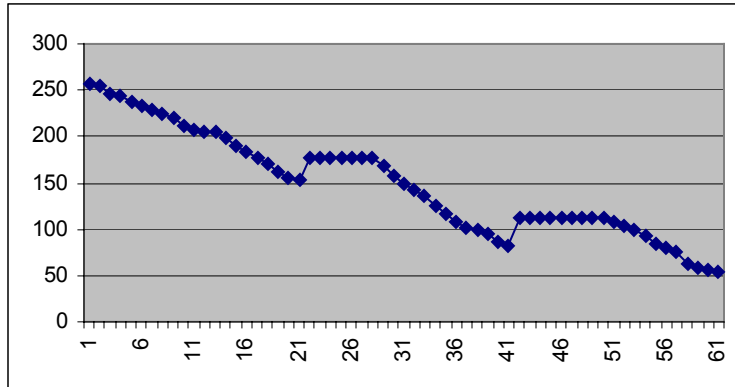


Fig 6.17 Adaptation curve to two disturbances

We can see flat period after each disturbance, which is due to the process of cutting off pathological branches. New branches won't start grow until all bad branches are cut off. The longer the fat period lasts, more effect caused by this disturbance.

Our Multi-fractal model shows a good capability of adapting to arbitrary topological disturbance. Since it is on a run and fix mode, new routing tree may not be the efficient one for the new grid setting. Energy and computation is saved this way and performance is acceptable shown by examples.

#### 6.4.5 Discussion of Probabilities Set

Why natural DLA rules don't apply in our case? The reason is that we aim to find routes for EVERY sensor in the network, not some critical nodes scattered in the network. For flat structure sensor network, every node needs to establish a message route to the data sink. This type of communication incurs heavy transmission overhead and network congestion with a very dense routing tree. If we only need to build up a communication network among some crucial nodes in a flat structure network, DLA rule can be utilized to find a sparse tree, which would expand to prioritized-regions with low routing tree densities. Also, if we have a hierarchical structure sensor network with leave node, cluster head and root (data sink), Cluster head is responsible for collecting and analyzing information from its leave node and we only need to maintain connections among data sink and cluster heads and possible some gateway nodes. Usually

speaking, cluster heads would rather scatter around the sensor network than huddle together. DLA rules can help to grow a sparse routing tree, which establish connections only between data sink and cluster heads. Moreover, DLA rules can inhibit neighboring sensor nodes or cluster heads from joining the tree together. Since neighboring nodes or cluster heads are very likely to sense similar terrain information, the neighboring growth inhibition can save precious resource to prolong the system lifetime. Obviously, a sparse tree with less communication pipes will consume less energy than a dense tree.

## Chapter 7

### Conclusion

#### 7.1 Extensions to General CA Model

In chapter 3, we discussed some essential properties of general Cellular Automata model. Versatility of CA requires more and more extension to the general model to be made in order to simulate unceasingly emerging applications. Dynamics and high uncertainty embedded in ad hoc wireless sensor network call for a highly flexible CA model rather than a rigid one.

##### 7.1.1 Time Synchronization

In a general CA model, a common clock drives the evolution of all cells, which enables synchronized actions among all identical cells. However, a common physical clock through the whole evolving period may not be feasible in a WSN after certain time steps even starting with an initial synchronization state. Time synchronization in WSN still requires further investigation and continuous effort.

In WSN, Physical time of sensor reading is a key point in sensor fusion, where individual reading are transformed into high level sensing data. For example, we want to estimate the velocity of an object. A serial of reading recorded by wireless distributed sensors at different locations need to be correlated, and a common physical clock among these sensors becomes essentially necessary. However, time synchronization scheme for traditional network are not suited for WSN. Traditional protocols for physical clock synchronization usually rely on exchange of clock information between clients and one or a few servers. The message relaying is a serious problem, since it depends on two important assumptions:

- a. Nodes not necessarily adjacent are connected before the time they need to be synchronized.

- b. The message delay between two nodes to be synchronized can be estimated over time with high precision.

Due to the high dynamics and ad hoc of WSN, above two conditions cannot always be satisfied. There is no connection between two nodes in prior until necessary. The message delay time between two wireless nodes are highly unpredictable and is not negligible. In a wired network, the delay time usually can be estimated with high precision or can be neglected. Some time synchronization approaches for WSN have been proposed such as: Typical GPS infrastructure at precision of 200ns, Römer's scheme at precision of 1ms, Elson's RBD at precision of 1 $\mu$ s and Multi-hop extension to RBS at a lower precision than RBD. No single mechanism is efficient in all aspects [12].

#### 7.1.2 Arbitrarily Shaped Grid and Cell

In a general CA models, all cells are identical. In order to best describe practical scenario in WSN, arbitrary shape of cells as well as grid should also be allowed. First, the terrain under sensing may have a quite irregular shape. Second, huge variety in the size and shape of sensing equipments and various terrain conditions make identically divided cells unpractical. With respect to this fact, our CA model should allow cell size as well as grid shape to take different shapes.

#### 7.2 Communication Overhead Analysis

As we pointed out, energy conservation is one of the most important aspects in designing WSN. The radio communication among nodes consumes the majority of the limited battery supply equipped with the nodes. Upon energy depletion or destruction, node will drop out of the network. In order to prolong system lifetime, we should cut down the communication to the minimal level. Due to the high unpredictability of wireless network infrastructure, periodic hello and acknowledgement messages exchange are carried out to maintain the network topology. Each node will broadcast hello message to all its possible neighbors within its radio communication scope and then wait for acknowledgements. The number of acknowledgements received within the time out time will be the number of viable neighbors of this node at current time step.

In Spin Glass model, each node sends its potential value to all its neighbors at each time step. After receive all its neighbor's potential values, each node calculates its new potential value and spin direction is decided based on Boltzmann distribution equation. In this model, there are at most eight neighbors for each node for Moore neighborhood. The communication cost for a single potential value is supposed to be one byte. Total communication cost at worst would be:

$$E = 8xGxN$$

*G is generation number N is total number of sensor nodes*

In Multi-fractal model, only the nodes in the routing tree broadcast their distance value to their neighbors. Nodes not in the routing tree will decide to join or not join the routing tree based on the number of messages received from its neighbors. The disadvantage of this approach lies in the fact that communication burden is put on the nodes in the routing tree. The earlier a node joins the routing tree, the more likely it will deplete its energy and dies. If some nodes are in low energy state, it should be able to select a substitute node to replace itself. Total communication cost would be:

$$E = 8xGxN_r$$

*G is generation number  $N_r$  is total number of sensor nodes in the routing tree*

It shows that Multi-fractal model consume less energy than Spin Glass model. However, this comes with a cost. Multi-fractal model is less adaptive to disturbance than Spin Glass model.

### 7.3 Comparison of Our Routing Models with Other Models

Table 7.1 gives a brief comparison of our routing models with other routing models. Our new models are distributed in computation and can both allow flat and hierarchical sensor network structures.

Our current simulation doesn't serve as an exact model for practical ad hoc WSN. However, we can easily extend our model to adapt to real ad hoc WSN settings. We will discuss some issues, which needed to be considered in Chapter 8.

Table 7.1 Comparison of routing protocols [1]

	Network structure	Update data to?	Communication Complexity	Route Metric	Parallel Computing
DSDV	Flat	Neighborhood	$O(N)$	Shortest	NO
CGSR	Hierarchical	Neighborhood & Cluster Head	$O(N)$	Shortest	NO
WRP	Flat	Neighborhood	$O(N)$	Shortest	NO
AODV	Flat	Neighborhood	$O(2N)$	Freshest and shortest	NO
DSR	Flat	Neighborhood	$O(2N)$	Shortest	NO
TORA	Flat	Neighborhood	$O(2N)$	Shortest	NO
Spin Glass	Flat/ Hierarchical	Neighborhood	$O(8N)$	Shortest	YES
Multi-fractal	Flat/ Hierarchical	Neighborhood	$O(8N)$	Shortest	YES

## Chapter 8

### Future Work

Our current CA model doesn't represent real ad hoc wireless sensor network. Several modifications need to be made to adjust to practical application.

#### 8.1 Practical Sensor Grid Configuration

In an ad hoc WSN, sensors are very likely first dropped off from an aircraft in a terrain of interest; Sensor nodes fail when subject to destruction or depletion of battery; New sensor nodes will be supplemented for a priority area. The high unpredictability of network structure obstructs the hope for a regular structure network structure. A cell can be empty or occupied by a sensor node and each occupied can have neighbors ranging from zero to eight depending on real scenario. Two more cell status is needed to distinguish cell with and without sensor nodes, namely empty cell and sensor node. We cannot assume such ideal setting with each cell occupied by a sensor node and each cell has exactly eight neighbors.

In real setting, each sensor node should be aware of its neighbors by periodically sending messages to its all possible neighbors and waiting for acknowledgement. Communication requirement for cellular automata rules are limited within current valid neighbors. It is possible that there might not be a valid route between some sensor nodes and data sink in a practical WSN setting. Fig 8.1 shows a sensor and its neighborhood in a real ad hoc WNS setting.

If we look at sensor  $s$ , its radio transmission scope is radius  $r$ . All sensors within the green circle are the neighborhood of sensor  $s$ . As shown above,  $s$  has six neighborhood as  $a$ ,  $b$ ,  $c$ ,  $d$ ,  $e$  and  $f$ . And this neighborhood consists of two adjacent layers of cells. Neighborhood establishment is initiated by hello and acknowledgement message exchange. Cells can be occupied by a sensor or unoccupied.

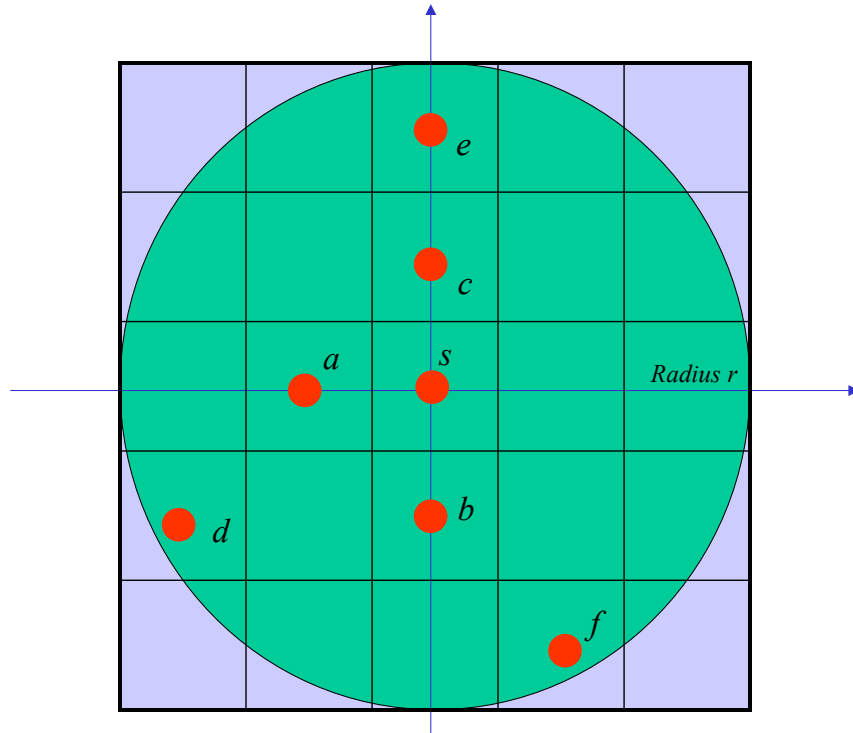


Fig 8.1 A 2D real ad hoc WNS sensor distribution example

In Spin Glass model, potential field is established starting from the data sink. Suppose  $s$  is the data sink, its weight for different neighbors are different depending on the amount of transmission energy needed for such hop to that neighbor. For instance, weight to  $a$  approximately equals weight to  $b$  but weight to  $f$  should be higher than that to  $a$ . Each neighbor of  $s$  will determine its energy value by adding the energy value of  $s$  to the corresponding weight value to that neighbor. And so on, until all cells find their energy values. Spin direction in the improved model should consist of radius number in addition to one of the eight directions used in our old model, since we need to differentiate between  $e$  and  $c$ .

In Multi-fractal model, there is no much different between the improved model and the old model, except the new neighborhood establishment step. Each cell will decide to join the routing tree or not depending on the number of its neighbors in the routing tree.

## 8.2 From Flat to Hierarchical Structure

Instead of flat structure, hierarchical structured sensor network is becoming more and more popular in military and civilian applications with un-comparable merits. There are three kinds of nodes, namely leaf node, cluster head and root node. As shown by Fig 8.2:

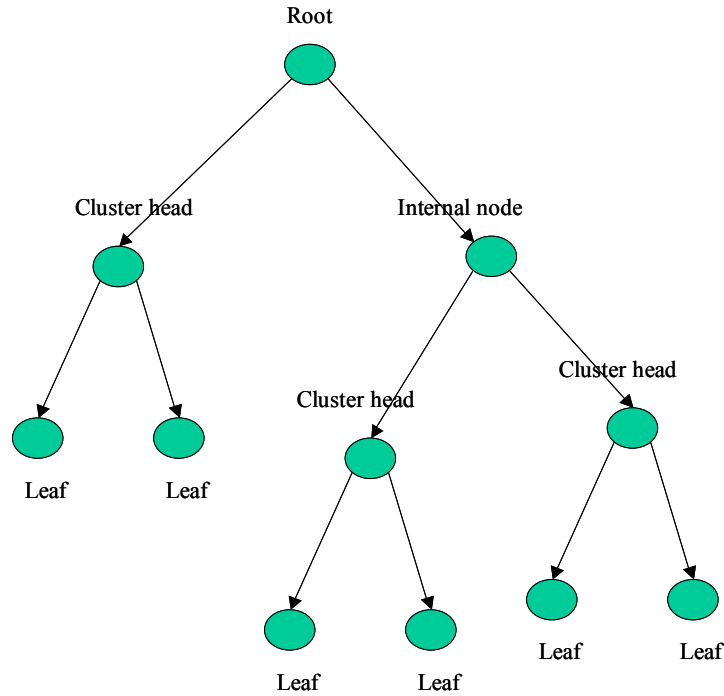


Fig 8.2 Relationships between hierarchy levels

Internal nodes are inserted between the root node and cluster heads. Internal nodes are implemented by designing defining either root or cluster head nodes so that they can be connected recursively. Strategies to self organize topology structure is described in [13].

Connections is only required among data sink and cluster heads. Only data sink and the cluster heads are treated as cellular automata cells and routing structure will be established following our Spin Glass or multi-fractal models. In this sense, our Spin Glass model is similar to Temporally Ordered Routing Algorithm. Our Spin Glass model has the advantage of parallel computation over TORA. Our multi-fractal model can grow a sparse routing tree without nearby nodes in the

tree simultaneously. The sparse tree can save limited power supply in our ad hoc wireless sensor network and prolong our system lifetime.

### 8.3 3D Animations



Fig 8.3 An example of a real military battlefield scenario from RRB presentation

Our current model characterizes a two-dimensional grid. Our system can be easily extended to three-dimensional setting. Suppose each cell has six neighbors each share a face with it, we can have six different weights standing for energy cost from different directions. For instance, if energy is measured as robot wandering energy cost, going up hill direction weight will always be greater than the that of down hill direction. We can build 3D spatial CA settings for particular scenarios such as mountain, river or plain areas. By using different height and arrows to represent

the terrain condition and next hop direction respectively, we can visualize a 3D grid setting. Same Spin Glass and Multi-fractal rules can be applied to above 3D cellular automata, evolution of routes establishment can be visualized at each time step. However, neighborhood is not necessarily limited to six. A more complex neighborhood can be employed to meet practical needs.



Fig 8.4 Example of a 3D CA lay out with six neighbors

## References

- [1] Elizabeth M. Royer. A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks IEEE Personal Communication April 1999
- [2] Emer Master Thesis Titled as Routing Protocols Available at <http://www.cmpe.boun.edu.tr/~emre/research/msthesis/node17.html>
- [3] Kendall Preston, Jr and Michael J. B. Duff Modern Cellular Automata theory and applications. Plenum Press 1984
- [4] Tommaso Toffoli and Norman Margolus Cellular Automata Machines The MIT press Cambridge, Massachusetts London 1987
- [5] Bastien Chopard, Pascal Luthi and Alexandre Masselot University of Geneva, Switzerland Cellular automata and lattice Boltzmann techniques: an approach to model and simulate complex systems
- [6] <http://www.bitstorm.org/gameoflife/>
- [7] John A. Major, ASA, MAAA and Yakov Lantsman. Part 1 Actuarial Application of Multifractal Modelling
- [8] B.M.Gammel Matpack C++ Numerics and Graphics Library Jun 13, 2002
- [9] Harold Brochmann Fractal Geometry 2002
- [10] Rudolf H.Riedi Introduction to Multifractals Dept of ECE. Rice University October 1999
- [11] Richard J.Gaylord Kazume Nishidate Modeling Nature Cellular Automata Simulations with Mathematica
- [12] Jeremy Elson and Kay Romer Wireless Sensor Networks: A New Regime For Time Synchronization Proceedings of the first workshop on hot topics in networks Oct 2002
- [13] Deborah Estrin and Ramesh Govindan etc. " Next Century Challenges: Scalable Coordination in Sensor Networks ", USC / Information Sciences Institute
- [14] J. Mannermaa, K. Kalliomaki, T. Manten, and S. Turunen. Timing performance of various GPS receivers. In proceedings of the 1999 Joint Meeting of the European Frequency and Time Forum and Time Forum April 1999
- [15] Nathan L. Orr Master Thesis entitled A Message-Based Taxonomy Of Mobile Code for Quantifying Network Communication August 2002
- [16] Stephen Wolfram A New Kind of Science Wolfram Media Inc, ISBN I-57955-008-8 2002
- [17] C.Y. Lee, An algorithm for path connection and its application IRE Trans. Electronic Computer, vol. EC-10, 1961, pp. 346-365

## Vita

Mengxia Zhu was born in Nanjing, China, on Oct 30, 1973. She received her bachelor's degree in biomedical engineering from Zhejiang University, China, in 1996. She is currently a master's student in Computer Science Department at Louisiana State University. As a member of Robotics Research Laboratory, she has been working in a joint DOD (Department of Defense) MURI (Multidisciplinary University initiative Program) project titled as "Automated Self-Configuring Surveillance Networks" with other students and faculties at Penn State University, University of Wisconsin, Duke University, and Cornell University. Her research areas include Web Image Retrieval using Neural Network, Discrete Event System Controller and Wireless Sensor Network Routing using Cellular Automata.