Load balancing in a network using Ant colony optimization technique

A thesis report submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology in Computer Science

By

Sachin Thakur
Roll No.: 10506016

&

Saurabh Tripathi

Roll No.:10506037

Under the guidance of

Mr. Pabitra Mohan Khilar
Sr. Lecturer
Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela – 769 008

National Institute of Technology Rourkela
Rourkela-769008, Orissa

Certificate

This is to certify that the work in this Thesis Report entitled "**Load balancing in a network using Ant colony optimization technique** " submitted by **Saurabh Tripathi**(10506037) and **Sachin Thakur**(10506016) has been carried out under my supervision in partial fulfillment of the requirements for the degree of Bachelor of Technology in Computer Science during session 2008-2009 in the Department of Computer Science and Engineering, National Institute of Technology, Rourkela, and this work has not been submitted elsewhere.

Place: NIT Rourkela
Date :

(**Pabitra Mohan Khilar**)
Sr. Lecturer
Department of CSE
NIT Rourkela

Acknowledgements

No thesis is created entirely by an individual, many people have helped to create this thesis and each of their contribution has been valuable. My deepest gratitude goes to my thesis supervisor, Mr. **Pabitra Mohan Khilar**, Sr. Lecturer, Department of CSE, for his guidance, support, motivation and encouragement through out the period this work was carried out. His readiness for consultation at all times, his educative comments, his concern and assistance even with practical things have been invaluable.

I am grateful to **Dr. Banshidhar Majhi**, *Professor and Head, Dept. of CSE* for his excellent support during my work. I would also like to thank all professors and lecturers, and members of the department of Computer Science and Engineering for their generous help in various ways for the completion of this thesis. A vote of thanks to my fellow students for their friendly co-operation.

Sachin Thakur
10506016

Saurabh Tripathi
10506037

Abstract

This thesis describes a method of achieving load balancing in telecommunications networks. A simulated network models a typical distribution of calls between nodes; nodes carrying an excess of traffic can become congested, causing calls to be lost. In addition to calls, the network also supports a population of simple mobile agents with behaviours modelled on the trail laying abilities of ants. The ants move across the network between randomly chosen pairs of nodes; as they move they deposit simulated pheromones as a function of their distance from their source node, and the congestion encountered on their journey. They select their path at each intermediate node according the distribution of simulated pheromones at each node. Calls between nodes are routed as a function of the pheromone distributions at each intermediate node. The performance of the network is measured by the average no of hops taken to complete the calls. In this thesis ,the results of using the ant-based control (ABC) are compared with those achieved by using fixed shortest-path routes,(dijkstra's algorithm) used in network management. The ABC system is shown to result in fewer call failures than the other methods, while exhibiting many attractive features of distributed control.

Contents

List of Figures

Chapter 1 Load balancing

Introduction

load balancing is a technique to spread work between two or more computers, network links, CPUs, hard drives, or other resources, in order to get optimal resource utilization, maximize throughput, and minimize response time.

Load balancing can be useful when dealing with redundant communications links. For example, a company may have multiple Internet connections ensuring network access even if one of the connections should fail. A failover arrangement would mean that one link is designated for normal use, while the second link is used only if the first one fails. With load balancing, both links can be in use all the time. A device or program decides which of the available links to send packets along, being careful not to send packets along any link if it has failed. The ability to use multiple links simultaneously increases the available bandwidth. Major telecommunications companies have multiple routes through their networks or to external networks. They use more sophisticated load balancing to shift traffic from one path to another to avoid network congestion on any particular link, and sometimes to minimize the cost of transit across external networks or improve network reliability.

The notion of complex collective behaviour emerging from the behaviour of many relatively simple units, and the interactions between them, is fundamental to the field of artificial life. The growing understanding of such systems offers the prospect of creating artificial systems which are controlled by such emergent collective behaviour; in particular, we believe that the exploitation of this concept might lead to completely new approaches for the management of distributed systems, such as load balancing in telecommunications networks.

In such networks, Calls between two points are typically routed through a number of intermediate switching stations, or nodes; in a large network, there are many possible routes for each such call. It is thus possible to relieve actual or potential local congestion by routing calls via parts of the network which have spare capacity. Load balancing is essentially the construction of call-routing schemes which successfully distribute the changing load over the system and minimise lost calls. Of course it is possible to determine the shortest routes from every node to every other node of the network. In this way the average utilisation of nodes will be minimised, but this is not necessarily the ideal way to avoid node congestion, as this has to do with how the traffic on the network is distributed.

Controlling distributed systems like these by means of a single central controller has several disadvantages. The controller usually needs current knowledge about the entire system, necessitating communication links from every part of the system to the controller. These central control mechanisms scale badly, due to the rapid increase of processing and communication overheads with system size. Failure of the controller will often lead to failure of the complete system. There is the additional practical commercial requirement that centrally controlled systems may need to be owned by one single authority. Further, the nature of distributed systems like these is highly dynamic, complex and stochastic, and their behaviour can neither be predicted nor explained by reducing it to a single central controllable factor.

A good decentralized control mechanism will not have the problems mentioned above. The field of artificial life has given us inspiration for such a mechanism that will be completely distributed, and highly adaptive to changes in the network and traffic patterns. This solution makes use of the distributed processing capability already inherently present in the network in the form of network nodes. The distributed nature of such an approach may make the system very robust against failures of individual control entities.

Our approach is inspired by the work of biologists studying social insects, who have uncovered the mechanisms controlling the foraging behaviours of ants[11]. The most important method is the laying and sensing of trails of pheromones - specialised chemical substances which are laid in amounts determined by local circumstances, and which by their local concentration subsequently directly influence an ant's choice of route.

Chapter

2 Ant colony optimization

2.1 Introduction

The notion of complex collective behaviour emerging from the behaviour of many relatively simple units, and the interactions between them, is fundamental to the field of artificial life. The growing understanding of such systems offers the prospect of creating artificial systems which are controlled by such emergent collective behaviour; in particular, we believe that the exploitation of this concept might lead to completely new approaches for the management of distributed systems, such as load balancing in telecommunications networks.

In such networks, Calls between two points are typically routed through a number of intermediate switching stations, or nodes; in a large network, there are many possible routes for each such call. It is thus possible to relieve actual or potential local congestion by routing calls via parts of the network which have spare capacity. Load balancing is essentially the construction of call-routing schemes which successfully distribute the changing load over the system and minimise lost calls. Of course it is possible to determine the shortest routes from every node to every other node of the network. In this way the average utilisation of nodes will be minimised, but this is not necessarily the ideal way to avoid node congestion, as this has to do with how the traffic on the network is distributed.

Controlling distributed systems like these by means of a single central controller has several disadvantages. The controller usually needs current knowledge about the entire system, necessitating communication links from every part of the system to the controller. These central control mechanisms scale badly, due to the rapid increase of processing and communication overheads with system size. Failure of the controller will often lead to failure of the complete system. There is the additional practical commercial requirement that centrally controlled systems may need to be owned by one single authority. Further, the nature of distributed systems like these is highly dynamic, complex and stochastic, and their behaviour can neither be predicted nor explained by reducing it to a single central controllable factor.

A good decentralized control mechanism will not have the problems mentioned above. The field of artificial life has given us inspiration for such a mechanism that will be completely distributed, and highly adaptive to changes in the network and traffic patterns. This solution makes use of the distributed processing capability already inherently present in the network in the form of network nodes. The distributed nature of such an approach may make the system very robust against failures of individual control entities.
Our approach is inspired by the work of biologists studying social insects, who have uncovered the mechanisms controlling the foraging behaviours of ants [11]. The most important method is the laying and sensing of trails of pheromones - specialised chemical substances which are laid in amounts determined by local circumstances, and which by their local concentration subsequently directly influence an ant's choice of route.

2.2 Ants in nature

Individual ants are behaviourally very unsophisticated insects. They have a very limited memory and exhibit individual behaviour that appears to have a large random component. Acting as a collective however, ants manage to perform a variety of complicated tasks with great reliability and consistency. A few examples of collective behaviour that have been observed in several species of ants are ([15]):

- Regulating nest temperatures within limits of 1°C.
- Forming bridges.
- Raiding particular areas for food.

- Building and protecting their nest .
- Sorting brood and food items.
- Cooperating in carrying large items.
- Emigration of a colony .
- Complex patterns of egg and brood care.
- Finding the shortest routes from the nest to a food source.
- Preferentially exploiting the richest available food source.

These behaviours emerge from the interactions between large numbers of individual ants and their environment. In many cases, the principle of stigmergy is used. Stigmergy is a form of indirect communication through the environment. Like other insects, ants typically produce specific actions in response to specific local environmental stimuli, rather than as part of the execution of some central plan. If an ant's action changes the local environment
in a way that affects one of these specific stimuli, this will influence the subsequent actions of ants at that location. The environmental change may take either of two distinct forms. In the first, the physical characteristics may be changed as a result of carrying out some task-related action, such as digging a hole, or adding a ball of mud to a growing structure. The subsequent perception of the changed environment may cause the next ant to enlarge the hole, or deposit its ball of mud on top of the previous ball. In this type of stigmergy, the cumulative effects of these local task-related changes can guide the growth of a complex structure. This type of influence has been called sematectonic (Wilson, 1975). In the second form, the environment is changed by depositing something which makes no direct contribution to the task, but is used solely to influence subsequent behaviour which is task related. This sign-based stigmergy has been highly developed by ants and other exclusively social insects, which use a variety of highly specific volatile hormones, or pheromones, to provide a sophisticated signaling system. Some of the above behaviours have been successfully

simulated with computer models, using both sematectonic stigmergy [14], and sign-based stigmergy [8] and also on robots ([12], [6]).
A type of sign-based stigmergy is used in our network model. It is based on the way ants find short routes from their nest to a food source, and also on the way they select between food sources of different value. The way ants organise these routes has inspired us to investigate a new approach for congestion avoidance in telecommunications networks.

2.2.1 Basic principles of trail laying

Depending on the species, ants may lay pheromone trails when travelling from the nest to food, or from food to the nest, or when travelling in either direction. They also follow these trails with a fidelity which is a function of the trail strength, among other variables. Ants drop pheromones as they walk by stopping briefly and touching their gaster, which carries the pheromone secreting gland, on the ground. The strength of the trail they lay is a function of the rate at which they make deposits, and the amount per deposit. Since pheromones evaporate and diffuse away, the strength of the trail when it is encountered by another ant is a function of the original strength, and the time since the trail was laid. Most trails consist of several superimposed trails from many different ants, which may have been laid at different times; it is the composite trail strength which is sensed by the ants. The principles applied by ants in their search for food are best explained by an example as given in [7]:

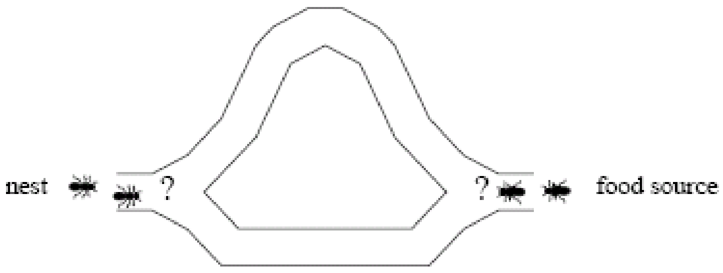

Figure 2.1 Ants have a decision to make

Figure 2 illustrates two possible routes between nest and food-source. Initially, an ant arriving at a T-crossing (choice point), makes a random decision with a probability of 0.5 of turning left or right. Now suppose there are two ants leaving the nest, looking for food, and two ants returning from the food source to the nest. Let the ants be of a type such as *Lasius Niger* which deposits pheromones when travelling both to and from the nest. If one ant from each pair turns left, and the other turns right, after a while a situation occurs like that in Figure 3. The lines on the paths represent the pheromone trails. The ants that chose the shorter branch have arrived at their destination, while the ones that chose the longer branch are still on their way. Ants initially select their way with a 0.5 probability for both branches, as there is no pheromone on the paths yet. If there is pheromone present, there is a higher probability of an ant choosing the path with the higher pheromone concentration, i.e. the path where more ants have traveled recently. If at the moment of the situation in Figure 3 other ants arrive and have to choose between the two paths, they are more likely to choose the shorter path, because that is where the concentration of pheromone is higher. This means that the amount of pheromone on the shorter path is more likely to be reinforced again. In this way, a stronger pheromone trail will arise on the shorter path, and so the path will be selected by an increasing proportion of ants As fewer ants choose the longer path, and the existing pheromone slowly evaporates, the trail on the longer path will weaken and eventually disappear.

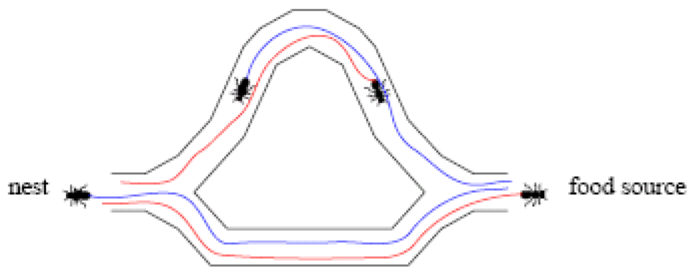

Figure 2.2 simulation several moments later

Although this is essentially self-organisation rather than learning, ants have to cope with a phenomenon that looks very much like overtraining in reinforcement learning techniques. There are two main issues: the blocking problem and the shortcut problem (Sutton, 1990). The blocking problem occurs when a route previously found by the ants is no longer available. It can then take a relatively long time for the ants to find a new route. The shortcut problem occurs when a new, shorter route suddenly becomes available. In this case the new route will not easily be found, because the old trails are so strong that almost all the ants choose them.

2.3 Ant-Based Control (ABC) for network management

How could this trail laying and following behaviour be applied to something like a telecommunications network? And can we overcome the blocking problem and the shortcut problem? This section describes how we implemented an artificial ant population on the network model. Further details may be found in [5].

2.3.1 Pheromone tables

We replaced the routing tables in the network nodes by tables of probabilities, which we will call 'pheromone tables', as the pheromone strengths are represented by these probabilities. Every node has a pheromone table for every possible destination in the network, and each table has an entry for every neighbour. For example, a node with four neighbours in a 30-node network has 29 pheromone tables with four entries each. One could say that an *n*-node network uses *n* different kinds of pheromones. The entries in the tables are the probabilities which influence the ants' selection of the next node on the way to their destination node. Figure 4 shows a possible network configuration and a pheromone table. For example, ants travelling from node 1 to node 3 have a 0.49 probability of choosing node 2 as their next node, and 0.51 of choosing node 4. 'Pheromone laying' is represented by 'updating probabilities'.

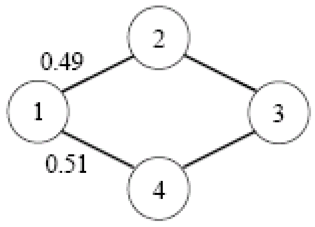

Figure 3 Using ants for network management.

The three pheromone tables for node 1:

		Next node	
		2	4
Destination node	2	0.95	0.05
	3	0.49	0.51
	4	0.05	0.95

At every time step during the simulation, ants can be launched from any node in the network. Each ant has a random destination node. Ants move from node to node, selecting the next node to move to according to the probabilities in the pheromone tables for their destination node. Arriving at a node, they update the probabilities of that node's pheromone table entries corresponding to their *source* node i.e. ants lay the kind of pheromone associated with the node they were launched from. They alter the table to increase the probability pointing to their previous node.

The method used to update the probabilities is quite simple: when an ant arrives at a node, the entry in the pheromone table corresponding to the node from which the ant has just come is increased according to the formula:

$$P_{new} = \frac{P_{old} + \Delta P}{1 + \Delta P}$$

Here *p* is the new probability and Δ*p* is the probability (or pheromone) increase. The other entries in the table of this node are decreased according to:

$$P = \frac{P_{old}}{1 + \Delta P}$$

Since the new values sum to 1, they can again be interpreted as probabilities. Note that a probability can be reduced only by the operation of normalisation following an increase in another cell in the table; since the reduction is achieved by multiplying by a factor less than one, the probability can approach zero if the other cell or cells are increased many times, but will never reach it. For a given value of Dp the absolute and relative increase in probability is much greater for initially small probabilities than for those which are larger. This has the effect of weighting information from ants coming from nodes which are not on the currently preferred route, a feature which may assist in the rapid solution of the shortcut problem.

2.3.2 Ageing and delaying ants

A primary requirement of this work was to find some simple methods of encouraging the ants to find routes which are relatively short, yet which avoid nodes which are heavily congested. Two methods are used. The first is to make Dp, the value used to change the pheromone tables, reduce progressively with the age of the ant. When the ant moves at one node per time step, the age of the ant corresponds to the path length it has traced; this biases the system to respond more strongly to those ants which have moved along shorter trails. The second method, which depends on the first, is to delay ants at nodes that are congested with calls to a degree which increases with the degree of congestion. This delay has two complementary effects:

- it temporarily reduces the flow rate of ants from the congested node to its neighbours, thereby preventing those ants from affecting the pheromone tables which are routing ants to the congested node, and allowing the probabilities for alternative choices to increase rapidly.
- since the ants are older than they otherwise would have been when they finally reach the neighbouring nodes, they have less effect on the pheromone tables.

It is of course possible to achieve the second effect alone by increasing the parameter representing the age of the ant without actually delaying the ant; this essentially reduces the effect on pheromone tables of an ant which has passed through a congested node. This has a biological parallel: in some species of ants, those returning from a richer food source tend to drop more pheromone than those from a poor source [11].In the case of our network simulation, however, the combination of delay and age-related penalty seems to be particularly effective. An added advantage of this formulation is that the manipulation of the parameter relating delay to the degree of congestion can be used to control the relative weighting which the system gives to preferring the shortest route (which maximises spare capacity), as against preferring the least congested route.

2.3.3 How calls are routed

Having explained how ants ‘choose’ their routes through the network, let us consider the calls. Calls operate independently of the ants. To determine the route for a call from a particular node to a destination, the largest probability in the pheromone table for this destination is looked up. The neighbour node corresponding to this probability will be the next node on the route to this destination. The route is valid if the destination is reached, and the call is then placed on the network, unless one of the nodes on the route is congested; in that case the call fails to be placed on the network.

In this way, calls and ants dynamically interact with each other. Newly arriving calls influence the load on nodes, which will influence the ants by means of the delay mechanism. Ants influence the routes represented by the pheromone tables, which in their turn determine the routing of new calls. These relationships are illustrated in Figure 5. One needs to realise that the pheromone table by which an individual ant is influenced, is a different table than the pheromone table that will be updated by this ant. The load on the network at any given time influences which calls can subsequently be placed on the network and which calls will fail; which of course determines the load at a later stage.

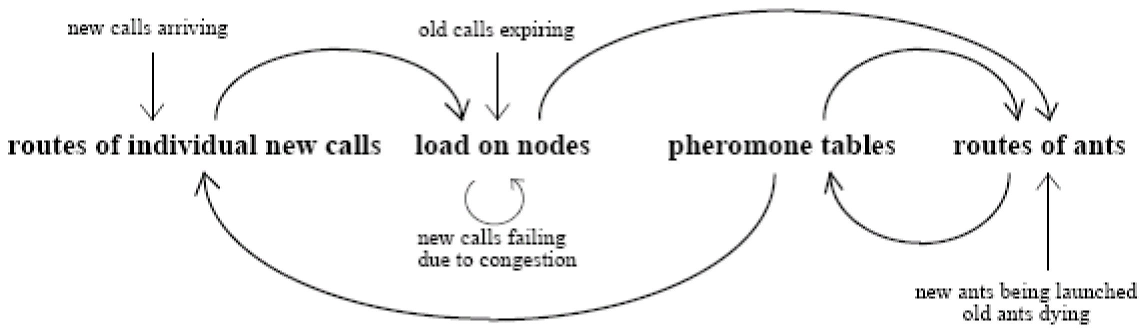

Figure 2.4 Relationship between calls, node utilisation, pheromone tables and ants. An arrow indicates the direction of influence.

2.3.4 General frame work of ant-based control systems

The basic principles for ant-based control (ABC) systems are applicable to a wide variety of problems, and can be characterised as follows:

- Ants are regularly launched with random destinations on every part of the system.
- Ants walk randomly according to probabilities in pheromone tables for their particular destination.

- Ants update the probabilities in the pheromone table for the location they were launched from, by increasing the probability of selection of their previous location by subsequent ants.
- The increase in these probabilities is a decreasing function of the age of the ant, and of the original probability.
- This probability increase could also be a function of penalties or rewards the ant has gathered on its way.
- The ants get delayed on parts of the system that are heavily used.

2.4 Ant Net algorithm

Suppose a data network, with **N** nodes, where *s* denotes a generic source node,when it generates an agent or ant toward a destination *d*. Two types of ants are defined:

1. Forward Ant, denoted $F_{s\to d}$, which will travel from the source node *s* to a destination *d*.
2. Backward Ant, denoted $B_{s\to d}$, that will be generated by a forward ant $F_{s\to d}$ in the destination *d*, and it will come back to *s* following the same path traversed by $F_{s\to d}$, with the purpose of using the information already picked up by $F_{s\to d}$ in order to update routing tables of the visited nodes.

Every ant transports a stack $S_{s\to d}(k)$ of data, where the *k* index refers to the *ke-st* visited node, in a journey, where $S_{s\to d}(0) = s$ and $S_{s\to d}(m) = d$, being *m* the amount ojumps performed by $F_{s\to d}$ for arriving to *d*.

Let *k* be any network node; its routing table will have N entries, one for each possible destination.

Let *j* be one entry of *k* routing table (a possible destination).

Let N_k be set of neighboring nodes of node *k*.

Let P_{ji} be the probability with which an ant or data packet in *k*, jumps to a node *i*, $i \in N_k$ when the destination is *j* ($j \neq k$). Then, for each of the N entries in the node *k* routing table, it will be n_k values of P_{ji} subject to the condition:

$$\sum_{i \in N_k} P_{ji} = 1; j = 1, \ldots, N$$

The following lines show AntNet pseudocode, using the symbols and nomenclature already presented:

```
BEGIN
{Routing Tables Set-Up: For each node $k$ the routing tables are initialized with a
uniform distribution:
                    $P_{ji} = \frac{1}{n_k}, \forall i \in N_k$
DO always (in parallel)
    {
     STEP 1: In regular time intervals, each node $s$ launches an $F_{s \to d}$ ant to a
    randomly chosen destination $d$.
    /*During its trip to $d$, when $F_{s \to d}$ reach a node $k$, ($k \neq d$), it does step 2*/
DO (in parallel, for each $F_{s \to d}$
    {
     STEP 2: $F_{s \to d}$ pushes in its stack $S_{s \to d}$ ($k$) the node $k$ identifier and the
    time elapsed between its launching from $s$ to its arriving to $k$.
    $F_{s \to d}$ selects the next node to visit in two possible ways:
            (a) It draws between $i$ nodes, $i \to N_k$, where each node $i$ has a $P_{di}$
                probability (in the $k$ routing table) to be selected.
               IF the node selected in (a) was already visited
            (b) It draws again, but with the same probability for all
                neighbor nodes $i$, $i \to Nk$. $F_{s \to d}$ jumps to chosen node.
                   IF the selected node was already visited
                          STEP 3: A cycle is detected and $F_{s \to d}$ pops from its
                    *stack* all data related to the cycle nodes, since the
                    optimal path must not have any cycle. $F_{s\to d}$ comes
                    back to step 2 (a).
                END IF
            END IF
        } WHILE jumping node $\neq d$
STEP 4: $F_{s\to d}$ generates another ant, called backward ant $B_{s\to d}$. $F_{s\to d}$ transfers to
        $B_{s\to d}$ its *stack* $S_{s\to d}$ and then dies.
/*$B_{s\to d}$, will follow the same path used by $F_{s\to d}$, but in the opposing
direction, that is, from $d$ to $s$*/

DO (in parallel, for each $B_{s\to d}$ ant)
        {
         /*When $B_{s\to d}$ arrives from a node $f$, $f \in Nk$ to a $k$, it does step 5*/
         STEP 5: $B_{s\to d}$ updates the $k$ routing table and list of trips, for the
         entries regarding to nodes $k'$ between $k$ and $d$ inclusive, according to
         the data carried in $S_{s\to d}$ ($k'$).
        IF $k \neq s$
         $B_{s\to d}$ will jump from $k$ to a node with identifier given by $S_{s\to d}$ ($k$-1)
        END IF
      } WHILE ($k \neq s$)
  }
}
END
```

The routing table and list of trips updating methods for k are described as follows:

1. The k routing table is updated for the entries corresponding to the nodes k' between k and d inclusive. For example, the updating approach for the d node, when $B_{s\to d}$ arrives to k, coming from f, $f \in Nk$ is explained, next:

- A P_{df} probability associated with the node *f* when it wants to update the data corresponding to the *d* node is increased, according to:

$$P_{df} \leftarrow P_{df} + (1 - r').(1 - P_{df})$$

where *r´* is an dimensional measure, indicating how good (small) is the elapsed trip time *T* with regard to what has been observed on average until that instant. Experimentally, *r′* is expressed as:

$$r' = \begin{cases} \dfrac{T}{c\mu} & c \geq 1 \quad \dfrac{T}{c\mu} < 1 \\ 1 & \text{otherwise} \end{cases}$$

where: μ is the arithmetic observed trip time *T* average.
c is a scale factor experimentally chosen like 2 (Dorigo 1997).

More details about r′ and its significance can be found in (Dorigo 1997).

- The other neighboring nodes (*j≠f*) *Pdj* probabilities associated with node *k* are diminished, in order to satisfy equation (1), through the expression:

$$P_{df} \leftarrow P_{df} + (1 - r').P_{df} \qquad \forall j \in N_k, j \neq f$$

2. A list $trip_k(\mu_i, \sigma_i^2)$ of estimate arithmetic mean values μ_i and associated variances σ_i^2 for trip times from node *k* to all nodes *i* (*i≠k*) is also updated. This data structure represents a *memory* of the network state as seen by node *k*. The list trip is updated with information carried by $B_{s\rightarrow d}$ ants in their stack $S_{s\rightarrow d}$. For any node pair source-destination,μ after (*n*+1) samples (*n*>0) is calculated as follows:

$$\mu_{n-1} = \frac{n\mu_X + x_{n+1}}{n+1}$$

where: x_{n+1} trip time T sample n+1,

μ_n arithmetic mean after n trip time samples.

Chapter 3 Dijkstra's Algorithm

3.1 Introduction

Dijkstra's algorithm, conceived by Dutch computer scientist Edsger Dijkstra in 1959, is a graph search algorithm that solves the single-source shortest path problem or a graph with nonnegative edge path costs, producing a shortest path tree.

Suppose you create a knotted web of strings, with each knot corresponding to a node, and the strings corresponding to the edges of the web: the length of each string is proportional to the weight of each edge. Now you compress the web into a small pile without making any knots or tangles in it. You then grab your starting knot and pull straight up. As new knots start to come up with the original, you can measure the straight up-down distance to these knots: this must be the shortest distance from the starting node to the destination node. The acts of "pulling up" and "measuring" must be abstracted for the computer, but the general idea of the algorithm is the same: you have two sets, one of knots that are on the table, and another of knots that are in the air. Every step of the algorithm, you take the closest knot from the table and pull it into the air, and mark it with its length. If any knots are left on the table when you're done, you mark them with the distance infinity.

Or, using a street map, suppose you're marking over the streets (tracing the street with a marker) in a certain order, until you have a route marked in from the starting point to the destination. The order is conceptually simple: from all the street intersections of the already marked routes, find the closest unmarked intersection - closest to the starting point (the "greedy" part). It's the whole marked route to the intersection, plus the street to the new, unmarked intersection. Mark that street to that intersection, draw an arrow with the direction, then repeat. Never mark to any intersection twice. When you get to the destination, follow the arrows backwards. There will be only one path back against the arrows, the shortest one.

3.2 Algorithm

Let's call the node we are starting with an initial node. Let a distance of a node X be the distance from the initial node to it. Dijkstra's algorithm will assign some initial distance values and will try to improve them step-by-step.

1. Assign to every node a distance value. Set it to zero for our initial node and to infinity for all other nodes.
2. Mark all nodes as unvisited. Set initial node as current.
3. For current node, consider all its unvisited neighbours and calculate their distance (from the initial node). For example, if current node (A) has distance of 6, and an edge connecting it with another node (B) is 2, the distance to B through A will be 6+2=8. If this distance is less than the previously recorded distance (infinity in the beginning, zero for the initial node), overwrite the distance.
4. When we are done considering all neighbours of the current node, mark it as visited. A visited node will not be checked ever again; its distance recorded now is final and minimal.
5. Set the unvisited node with the smallest distance (from the initial node) as the next "current node" and continue from step 3.

3.3 Pseudocode

```
function Dijkstra(Graph, source):
    for each vertex v in Graph:
        dist[v] := infinity
        previous[v] := undefined
    dist[source] :=
    Q := the set of all nodes in Graph
    while Q is not empty
        u := vertex in Q with smallest dist[]
        if dist[u] = infinity:
            break
        remove u from Q
        for each neighbor v of u:
            alt := dist[u] + dist_between(u, v)
            if alt < dist[v]:            // Relax (u,v,a)
                dist[v] := alt
                previous[v] := u
    return previous[]
```

3.4 theoretical Comparison with ant colony algorithm

Ant colony algorithm has a number of qualitative and quantitative advantages over dijkstra’s algorithm, as well as some disadvantages:

Consuming network resources. An ant hardly requires any bandwidth on the network: It only holds its age, and its source and destination identifiers . in dijkstra ,there are relatively large tables and therefore require much more bandwidth than ant colony algorithm. congestion dependent ants temporarily reduces traffic at the congested nodes.

Robustness. Malfunctioning in the system might cause a process such as an ant or to crash. If an ant crashes, this will not have a significant effect on the performance of the algorithm at all but if a node fails in djikstra algorithm all the tables have to be modified.

Computational issues. Ants are likely to require more computation on the nodes of the network, due to the extensive use of random generators. Further, with antbased control, nodes need to allocate more space for their pheromone tables than is needed when normal routing tables are used. However, these issues do not affect bandwidth or switching capacity, which is our main concern.

Bidirectional routes. During the simulations of the ant controlled system, the route from one node to another tends most of the time to be the same that in the opposite direction. This is probably due to our mechanism of trail laying, where ants from complementary source and destination nodes mutually reinforce one anothers’ trails. At first sight, this property might seem to be disadvantageous for good load balancing,but we believe that this will only make a significant difference in small networks.

Chapter

4 Experimental Results and Analysis

4.1 Simulation

The 30-node network topology of Figure 1 was chosen because this is a realistic interconnection structure of a possible switch-based network. It is also the same topology as was used in (Appleby & Steward, 1994), and is in fact the structure of the British Synchronous Digital Hierarchy (SDH) network.

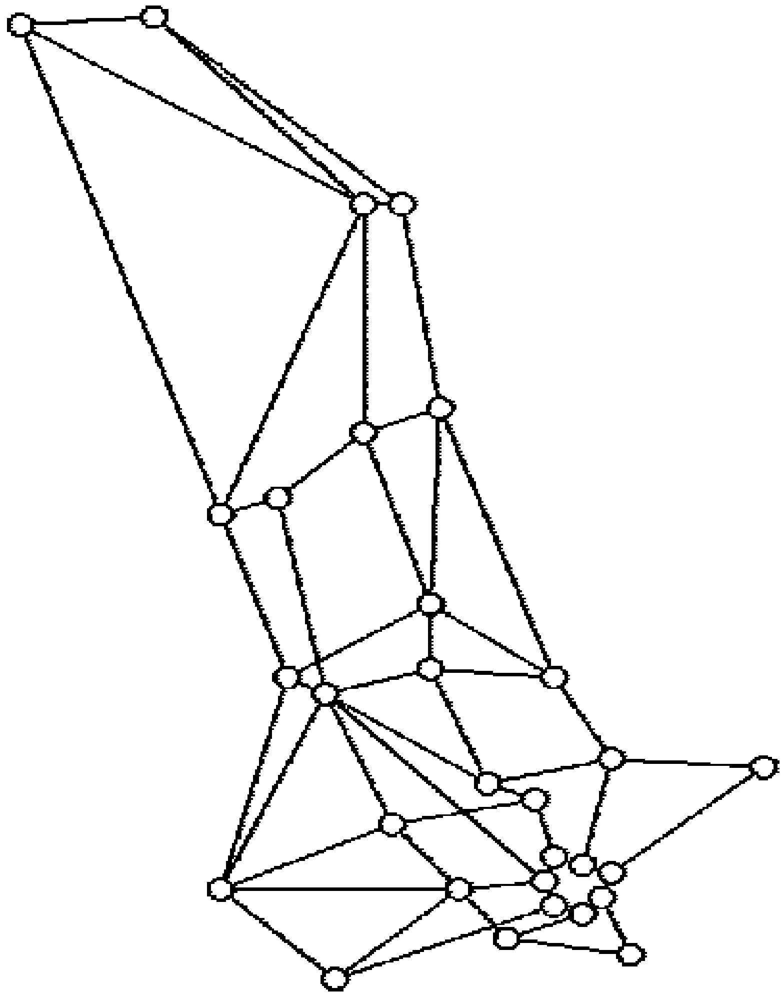

Figure 4.1 The network topology used for simulation.

The number of parameters that can vary in this network model is large. This implies that some arbitrary choices have to be made for testing:

- total no of calls to be made for each simulation = 1000,200,500
- node capacity of each node = 35 calls, so that every call using a node increases the utilization (decreases the spare capacity) of that node by 2.5%.
- during every time step of simulation , an average of 1 call is generated with a average duration of 170 time steps.
- Maximum concurrent calls that can be made in the network is 60.
- Simulation speed = 1000 time steps(ticks) per sec.

4.1.1 List of Classes

- MainForm: The GUI for the application
- DrawPanel: The real-time network simulation
- Global :Contains static variables that are accesses by the application
- Node : Represents a Node and holds an array of PheromoneTable objects for routing
- Call : Represents a call on the network and hold a source and destination node
- Simulation : Represents a completed simulation and is used for creating graphs
- PheromoneTable : A routing table for a Node

Graphs have been plotted using software " **dotnetcharting**".

The network contains 30 Nodes and each Node contains an array of PheromoneTable objects, one for every other Node in the network (29). Every

PheromoneTable contains an array of TableEntries, one for each Node connected to the current Node.

The following diagram represents the relationships between classes in the program.

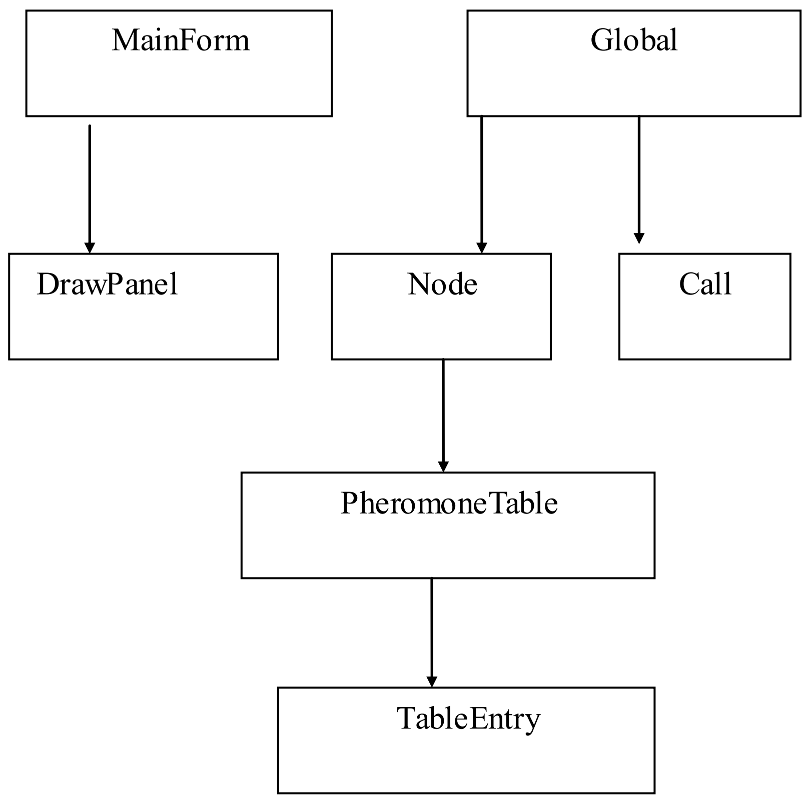

Figure 4.2 Relationship between Classes

4.2 Pheromone table updation

In the beginning , each possible path has an even likelihood of being chosen. For example , lets consider the following node structure. An ant is placed on a network of 4 nodes with the source node of 1 and destination node 2. A chance mechanism is invoked and a path is chosen.

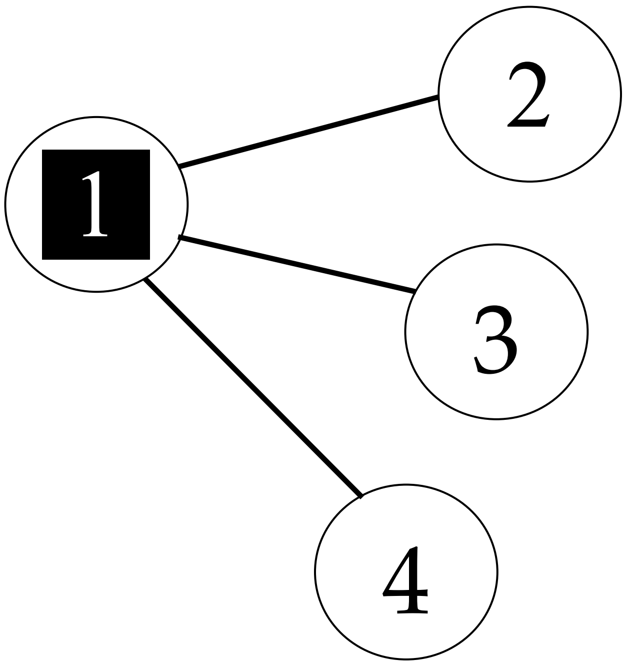

Figure 4.3.1 The network graph

Next node	Percentage chance
2	33.33%
3	33.33%
4	33.33%

Table 4.1 Pheromone table for node 1

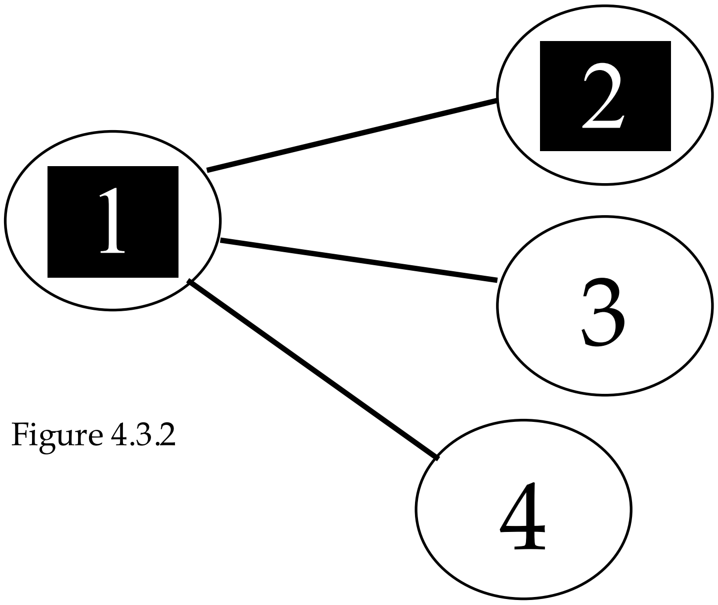

Figure 4.3.2

In this case node 2 has been selected [figure 4.3] and the ant arrives at its source destination. The ant then moves and updates the pheromone tables for the visited nodes with higher (and more mathematically biased) value. This would be calculated for figure 4.2 and table 4.1 in the following way:

- Node 2 was the final destination
- It took 1 hop to get to its destination
- Divide 1 (hop) by 100 : 100%
- Add 100 to the probability value of node 2 (currently 33.3333) : 133.3333
- Add the values of the other nodes to 133.3333 (133.3333+ 33.3333 + 33.3333) : 200 (approximately)
- Calculate the ratio : ratio = 100/200 0.5
- Set the probability of the node to its current value multiplied by the ratio
    - Node 2: 133.3333 * ratio (0.5) = 66.6666%
    - Node 3: 33.3333 * ratio (0.5) = 16.6666%
    - Node 4: 33.3333 * ratio (0.5) = 16.6666%
- Node 2 (66.6666%) + Node 3 (16.6666%) + Node 4 (16.6666%) = 99.9999%

The system isn't 100% accurate as the total will never add up to exactly 100% but it will be close enough to allow accuracy within the level required.

The following table shows the updated pheromone table of node 1 .

Next node	Percentage chance
2	66.66%
3	16.66%
4	16.66%

Tabe 4.2

4.3 Results

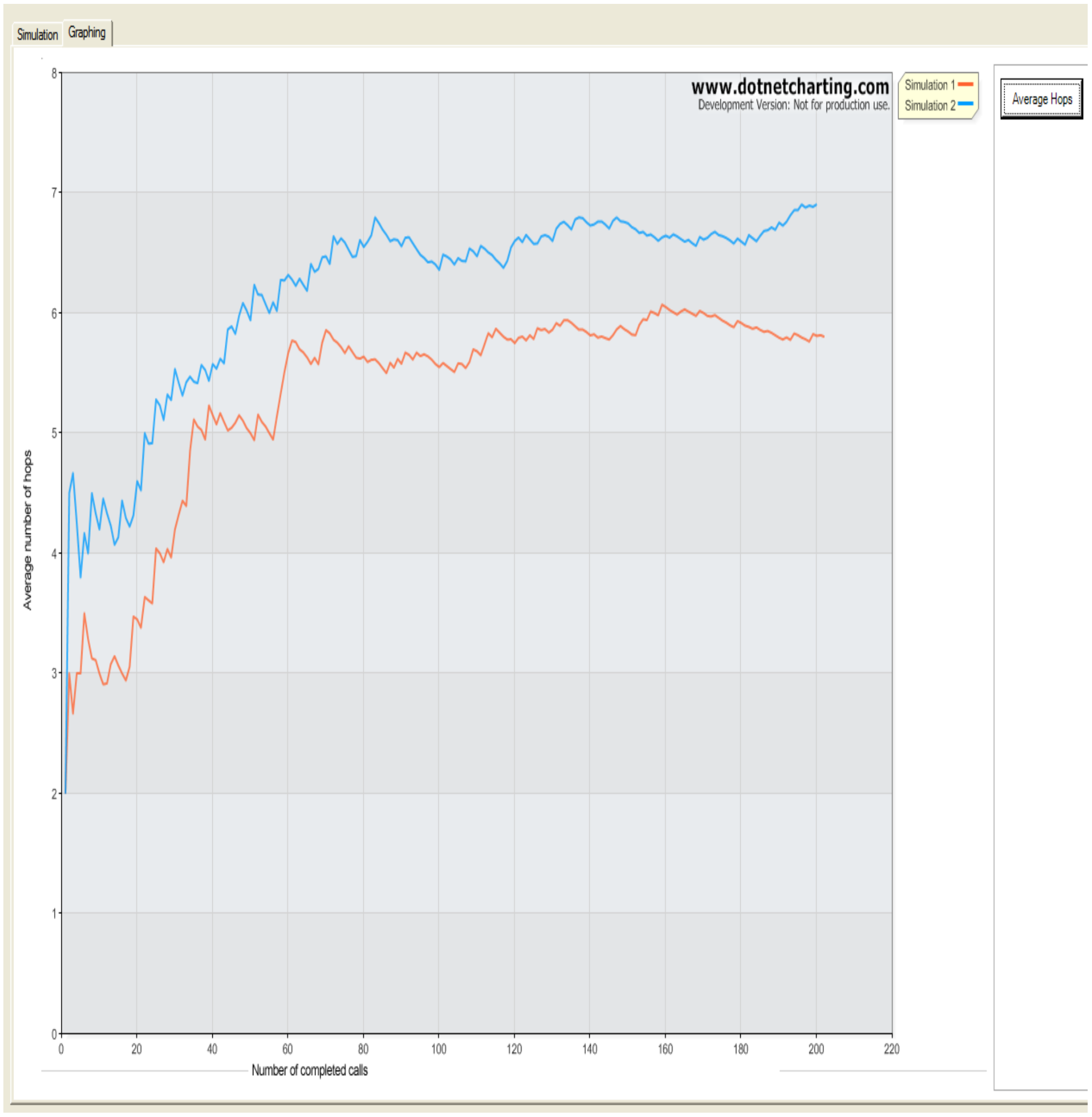

Figure 4.4 simulation for 200 calls [orange-antnet simulation, blue-dijkstra]

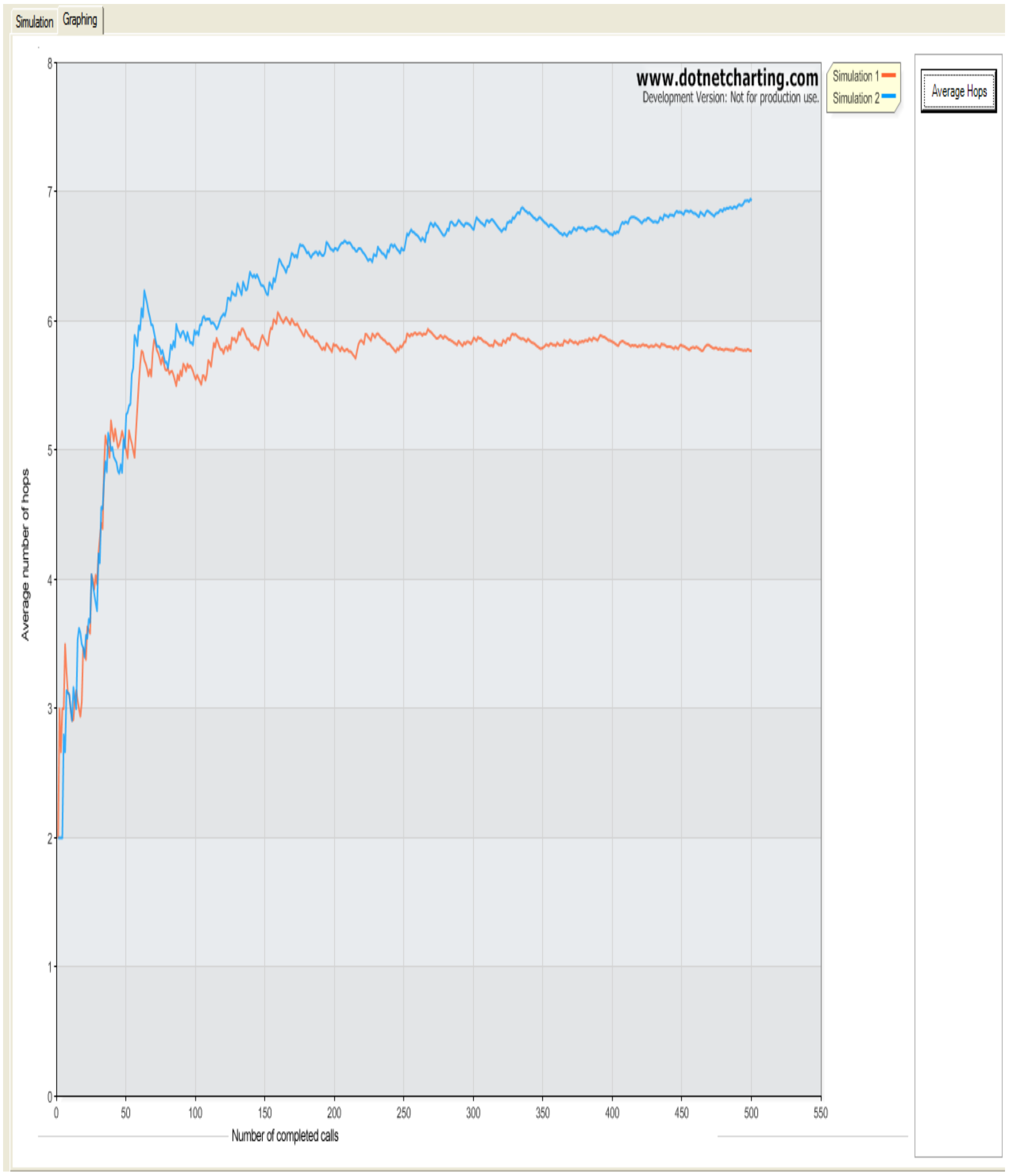

Figure 4.5 simulation for 500 calls [orange-antnet simulation,blue-dijkstra]

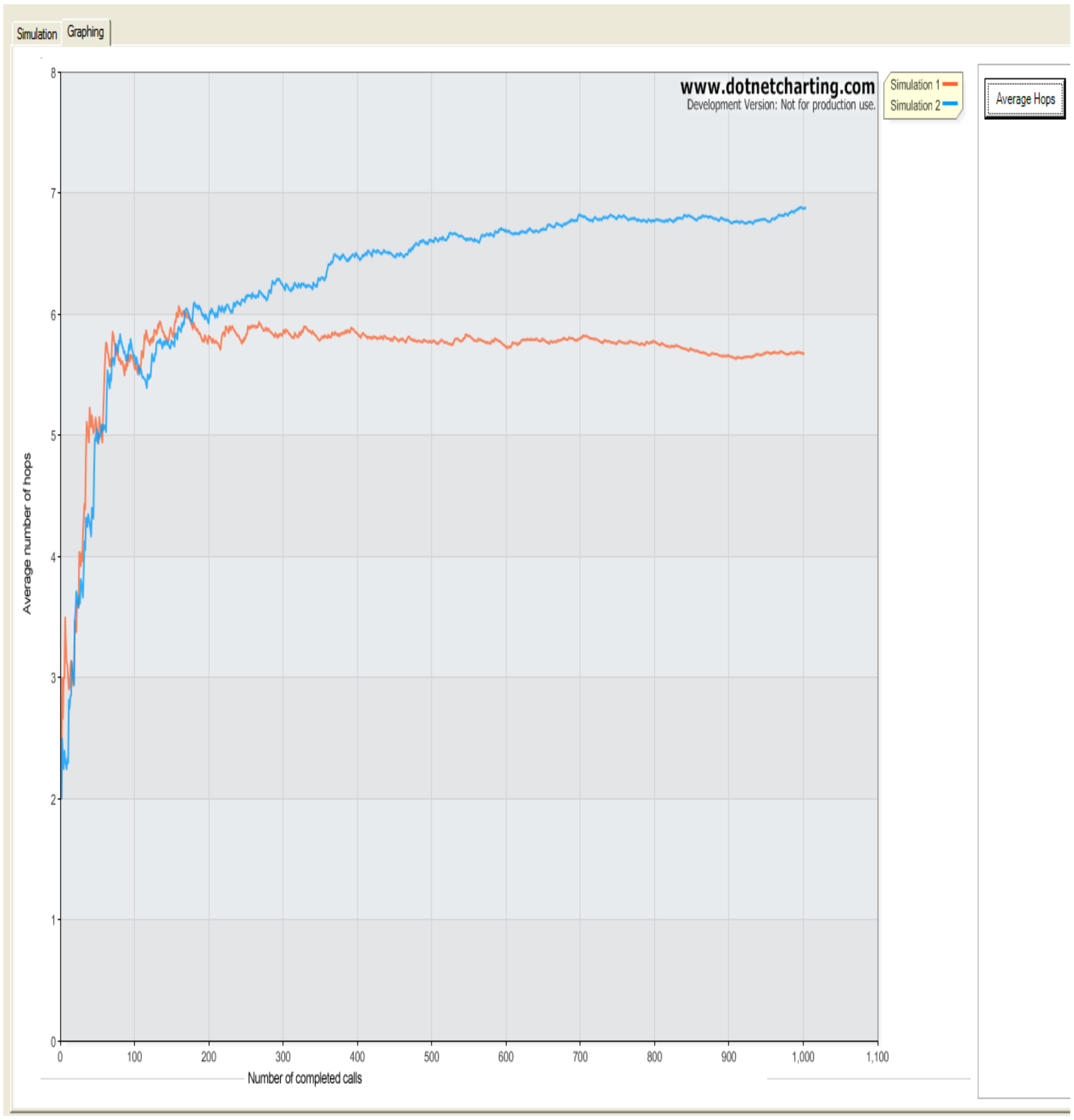

Figure 4.6 simulation for 1000 calls [orange-antnet simulation,blue-dijkstra]

4.3.1 Results Analysis

As per the graph we observe that average no of hops in each simulation are less in case of ant colony based algorithm than in case of dijkstra. For the initial calls there is not much difference between the both graphs , this is because the pheromone tables of each node in case of ant colony optimization technique are not normalized for initial calls made on the network. But as more calls are placed on the network the pheromone tables get normalized and there is not much variation in the pheromone tables . therefore the graph , in case of ant colony based algorithm, remains almost saturated. Hence from the results we inferred that ant colony based algorithms perform better than shortest fixed route algorithms.

Chapter 5 Conclusion and Future Work

5.1 Conclusion

As we have seen that that present telecommunication networks suffer from network congestion which causes calls, put on the network , to fail . Better load sharing and routing algorithms are required to minimize the effect of congestion. In this thesis, we have simulated two algorithms ,ant colony algorithm and dijkstra algorithm for load balancing in telecommunication network. Performance of both algorithms is studied with respect to the average number of hops taken to complete a fixed no of calls in the network as shown in the graph .we found out that the average no of hops are less in case of ant colony based method as compared to the dijkstra. Hence ant colony optimization technique performs better that fixed-shortest path route algorithms.

5.2 Future Work

We believe that ABC systems can be used to solve a large variety of optimisation problems, eg:

- distributing loads of interconnected processors on parallel machines and managing interprocessor communication for complex programs;
- material flow in production environments;
- optimal routing on integrated circuit-boards;
- organising public transport schemes.

Much investigation of the basic principles of the ant algorithm remains to be done. So far, our experiments have not yet enabled us to make statements that are sufficiently supported by statistics about the influence of the number of ants used in the simulations. We also do not know exactly how variations in the ants' influence on the pheromones affect the system, or about the effects of the size of the delays imposed on the ants. Most choices in this work are based on a relatively small number of experiments. It would also be useful to investigate the performance of the algorithm on extremely large or 25 very small networks; the large networks will tell us about scaling, and the small neworks might assist our understanding of the basic processes which make the algorithm work. Another intriguing possibility is to use 'probabilistic routing' of calls. Here routes of calls, or perhaps a proportion of calls, would not be chosen according to the largest probabilities in the pheromone tables, but randomly according to these probabilities. A mechanism that is assumed not to be used by natural ants, but could be useful here, is laying 'anti-pheromone'. One could let ants directly decrease probabilities in the pheromone tables in particular circumstances, rather than increase them.The pheromone tables do not only represent the best routes, but also contain information about the relative merits of alternative possibilities if something goes wrong. Our simulations have so far been confined to examining the use of this information in dealing with node congestion; sudden node (or link) failure and restoration also needs to be simulated to examine the abilities of the ants to deal with these contingencies. The ability to cope with the insertion of new nodes and links during network extension is also a topic of interest. Extending ant-like algorithms to situations like telecoms networks which are not found in nature will also increase our understanding of the abstract and general abilities of such algorithms over and above those applications found in nature. We hope that this increase of knowledge will in turn assist and inform biologists studying social insects.

References

[1] B. Barán, M Almirón, E. Chaparro. 'Ant Distributed System for Solving the Traveling Salesman Problem'. pp. 779-789. Vol. 2^{15} th Informatic Latinoamerican Conference-CLEI,. Paraguay (1999).

[2] S. Bak, J. Cobb, E. Leiss. 'Load Balancing Routing via Randomization'. pp. 999-1010. Vol. 2^{15} th Informatic Latino american Conference-CLEI. Paraguay (1999).

[3] M. Dorigo, V. Maniezzo, A. Colorni. 'The Ant System: Optimization by a colony of cooperating agents'. pp. 1-13. Vol. 26-Part B. IEEE Transactions on Systems, Man, and Cybernetics,. Number 1 (1996).

[4] M. Dorigo, G. Di Caro. 'AntNet. Distributed Stigmergetic Control for Communications Networks'. pp. 317-365 Journal of Artificial Intelligence Research. Number 9 (1998).

[5] [Schoonderwoerd et al 97] R. Schoonderwoerd, O. Holland, J. Bruten. 'Ant-like agents for load balancing in telecommunications networks'. Hewlett-Packard Laboratories, Bristol-England (1997).

[6] Beckers, R., Holland, O.E., & Deneubourg, J.L. (1994). From local actions to global tasks: Stigmergy and Collective Robotics. In R.A. Brooks, & P. Maes (Eds.) *Artificial Life IV*, Cambridge, MIT Press. p. 181-189.

[7] Beckers, R., Deneubourg, J.L., & Goss, S. (1992).Trails and U-turns in the Selection of a Path by the Ant Lasius Niger. In *J. theor. Biol. 159*, 397-415.

[8] Stickland, T.R., Tofts, C.M.N., & Franks, N.R. (1992). A path choice algorithm for ants. In *Naturwissenschaften 79,* 567-572.

[9] Goss. S., Beckers, R., Deneubourg, J.L., Aron, S., & Pasteels, J.M. (1990). How trail-laying and trail following can solve foraging problems for ant colonies. In R.N. Hughes (Ed.) *NATO AS1 Series, Vol. G20 Behavioral mechanisms of food selection*, Springer Verlag.

[10] Deneubourg, J.L., Goss, S., Franks, N., Sendova-Franks, A., Detrain C., & Chrétien, L. (1990). The dynamics of collective sorting robot-like ants and ant-like robots. In J.-A. Meyer & S. Wilson (Eds.), *From Animals to Animats: Proceedings of the first international conference on simulation of adaptive behavior*. Cambridge, MIT Press.

[11] Beckers, Deneubourg & Goss, 1992; Deneubourg & Goss, 1989; Goss et.al., 1990; Franks, 1989.

[12] Beckers, Holland & Deneubourg, 1994; Russell 1995; Deveza et.al., 1994.

[13] A. Shankar, C. Alaettinoglu, I. Matta. 'Performance Comparison of Routing Protocols using MaRS. Distance Vector versus Link-State'. Technical Report, Maryland-USA (1992)

[14] Modeling the collective building of complex architectures in social insects with lattice swarms, Theraulaz and Bonabeau, Journal of Theoretical Biology, 1995.

[15] Bert Hölldobler, E.O. Wilson: "Journey to the Ants: A Story of Scientific Exploration, *1994, ISBN 0-674-48525-4.*