uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Dia Elghobary

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.A.Sc. (Electrical Engineering)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

AntNet Based Routing Algorithms for Resource Constrained Networks

TITRE DE LA THÈSE / TITLE OF THESIS

Prof. El Saddik

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Prof. A. Boukerche

Prof. P. Liu

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

# AntNet Based Routing Algorithms for Resource Constrained Networks

By

Dia Elghobary

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements for the degree of
Master of Applied Science
In
Electrical and Computer Engineering

Ottawa-Carleton Institute of Electrical and Computer Engineering
School of Information Technology and Engineering

Faculty of Engineering
University of Ottawa

*Your file Votre référence*
*ISBN: 978-0-494-48452-4*
*Our file Notre référence*
*ISBN: 978-0-494-48452-4*

# Abstract

Research in the area of metaheuristics and routing in telecommunication networks have been of great interest in the past several years, especially in wireless networks where routing differs greatly in than in traditional networks. Due to resource constraints, the dynamic nature of networks, and unconventional destination specification, typical routing algorithms fair poorly. Ant Colony Optimization (ACO), a common metaheuristic has inspired routing algorithms in such networks. AntNet, a popular one, has shown promising performance results. However, AntNet has shown that it may not be the most efficient. In this thesis, we will examine, AntNet, suggest two alternative algorithms based on AntNet, and introduce a fourth algorithm based on a hybrid AntNet and genetic operator algorithm. The performance data provided by the tests show that the two improved algorithms and the fourth hybrid one introduce improved performance. Finally, based on the experience with the simulated algorithms, recommendations for future work are provided.

# Acknowledgements

I would like to express my sincere gratitude to my thesis supervisor Dr. Abdulmotaleb El Saddik, for giving me an opportunity to pursue my graduate studies and for his continuous support and guidance.

I would like to also thank my parents and siblings for their encouragement, love and support. I could not make it with out them.

# Table of Contents

# List of Figures

# List of Tables

# CHAPTER 1

# INTRODUCTION

## 1.1 Objective and Motivation

Routing is defined as the process of determining a path of delivery from a source to a destination in a network [Yun04]. Traditionally, routing in common networks is accomplished using traditional algorithms such as OSPF. While such algorithms may be sufficient when conditions are ideal, they suffer greatly in resource constrained topologies [Johnson95].

Routing with traditional algorithms becomes especially challenging in wireless ad-hoc networks. Although mobile hosts could act as individual routers, routing still remains especially difficult due to: memory constraints, limited computational power and battery life of mobile hosts, constant topology changes, and asymmetric connectivity, to mention a few reasons [Johnson95] [Zhang04].

The research in this thesis is motivated by real-world problems: how can we make efficient use of meager resources in wireless ad-hoc networks?

Metaheuristic based algorithms, such as AntNet, best suit such conditions. AntNet outperforms many conventional algorithms in solving routing problems [Cheng03]. The

reason it is widely applied and considered is due to its lack of central control mechanism and distributed computation. Even so, AntNet is not considered efficient enough and in many cases it performs quite poorly, especially in large networks where it has been observed that it converges quite slowly [Cheng03].

Our research objective is to design and simulate algorithms that introduce performance improvements to AntNet so that we can make efficient use of scarce bandwidth and minimize power consumption in resource strained networks. The objective is to also realize an algorithm that possesses rapid convergence in route discovery.

## 1.2 The Research Problem

AntNet is an algorithm that is based on Ant Colony Optimization (ACO) metaheuristic for routing in telecommunications networks. In this algorithm, a set of artificial ants work concurrently towards finding optimal routing paths. Artificial ants iteratively work on finding a solution to the routing problem and communicate with other artificial ants by way of informing them of possible routes to the destination by laying pheromone. This phenomenon is also known as stigmergy. AntNet has been compared to conventional routing algorithm and in many cases outperforms them.

However, AntNet has inherent deficiencies that have plagued its wide scale adoption. There are several reasons for that.

Firstly, AntNet is slow when converging towards the destination. This especially gets worse the larger and denser the network is.

Second, the overall low success rate and poor performance in resource constrained configurations.

And last but not least, although algorithms contributed by [Zhang04] and others are a positive step towards improving AntNet, none so far have presented an all inclusive algorithm that performs well in all performance aspects and in the same time scalable.

## 1.3 Thesis Contributions

This thesis contains a number of contributions, including:

- Analysis of ant behaviour and the concept of stigmergy
- Analysis of ACO and genetic algorithms
- Proposal of three algorithms that are based on AntNet [DiCaro98], namely Improved AntNet, Pharaoh Routing and Hybrid AntNet/Genetic algorithm.
- Implementation and simulation of proposed algorithms using popular multi-agent simulation tool NetLogo
- Tests and performance measurements of the proposed algorithms and comparison with AntNet based on varying common network parameters.

## 1.4 Organization of the Thesis

We have stated our objective and motivation, research problem and contributions. The remainder of the thesis is organized as follows:

- **Chapter 2** provides the background knowledge of ACO and genetic algorithms. The behaviour of the ant in nature is discussed first followed by an introduction to metaheuristics and ACO. AntNet, a popular ACO algorithm for routing in telecommunications networks is introduced followed by an introduction to genetic algorithms.
- **Chapter 3** introduces the three proposed AntNet based algorithms: Improved AntNet, Pharaoh Routing and Hybrid AntNet/Genetic algorithm. The rationale behind each algorithm is given followed by a brief theoretical evaluation.
- In **Chapter 4**, first the implementation basics of NetLogo are presented followed by the implementation details of the three proposed algorithms.

- **Chapter 5** covers the particulars of all experimentation and data collection done on the first two algorithms, (Improved AntNet and Pharaoh Routing) which are compared to AntNet according to a set of varying network parameters.
- **Chapter 6** evaluates the performance of the Hybrid AntNet/Genetic algorithm by comparing it to AntNet, Improved AntNet and Pharaoh Routing based on a baseline network configuration chosen in Chapter 5.
- **Chapter 7** summarizes the results and concludes the thesis and provides suggestions for potential future research work.

# CHAPTER 2

# BACKGROUND

## 2.1 Ants and Technology

Even with the increased computation power and technological advancements of our day, scientists have been looking for alternate ways to deal with problems that are ever increasing in complexity. Scientists have been looking for unconventional ways to tackle traditional problems such as the *Traveling Salesman Problem* [Tarasewich02] [Dorigo97].

Scientists have been looking at insect behavior for a source of inspiration for new approaches. Specifically, scientists have been looking at how ants forage together for food. Individually ants are very simple in their behavior, but together they form a complex system that is very resilient and flexible [Tarasewich02].

Ants communicate indirectly by way of stigmergy where one ant behaves according to the actions taken by a previous ant. [Dorigo06] defines stigmergy as "an indirect, non-symbolic form of communication mediated by the environment: insects exchange information by modifying their environment". The basic way ants use to communicate is by laying a chemical called pheromone. Pheromone attracts other ants and can guide

them to the source of food. In cases where there are multiple paths marked by pheromone, ants can pick the path with the highest concentration [Dorigo97]

Individual ants move randomly till a source of food is discovered. As they return to their nests, they lay down a trail of pheromone that leads to the food. Other ants are attracted to this pheromone trail and follow it to the source of the food. Those ants lay their own deposit of pheromone on their way back to the nest as well.

Initially, multiple paths may be discovered to the food. Ants taking the shorter paths will return to their nests first, and in the process lay more pheromone. Other ants will be attracted to these paths with higher concentration of pheromone. Eventually, shorter paths will remain as they will be traversed by far more ants than longer. This is clearly illustrated in figure 1 [Dorigo97].

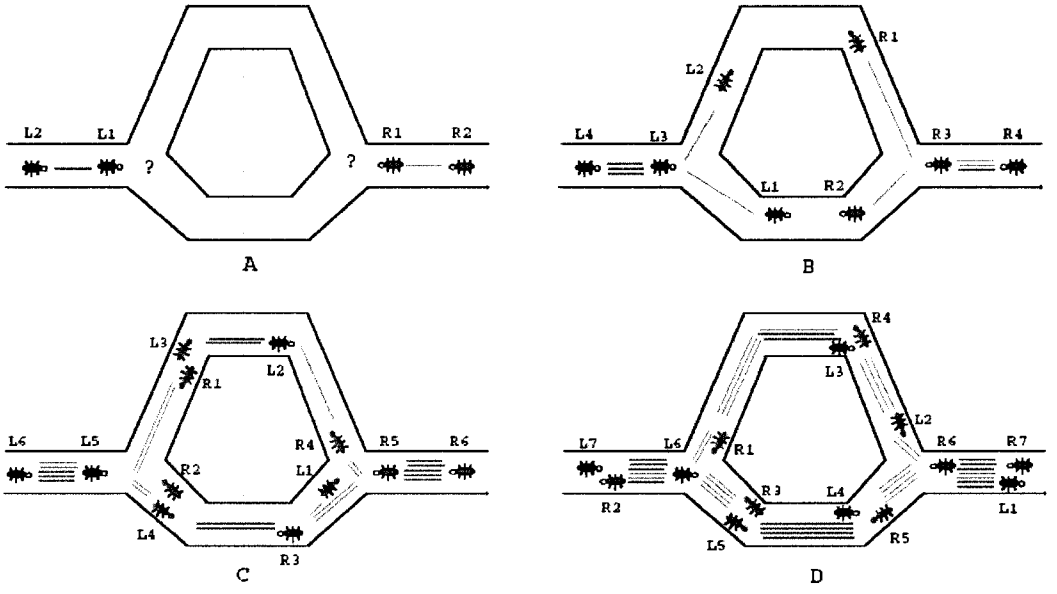


**Figure 1 How ants find the shortest path**

In figure 2, a snapshot is shown from a NetLogo simulation of a colony of ants foraging for food. The colony's nest is located in the middle of the screen while three sources of food are placed at different distances from the colony's nest. From the figure it is noticeable that the source of food directly to the right hand side is depleted first as it is

closer to the nest. More ants are traversing that path making the trail of pheromone to that source of food stronger than the trails to the other two sources that are placed further away from the nest.

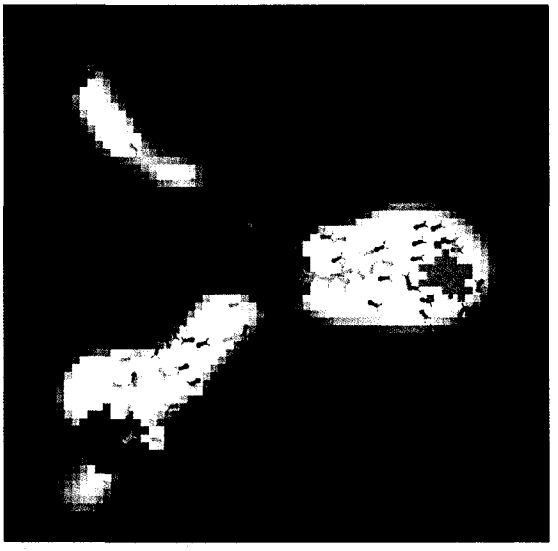

**Figure 2 NetLogo ant behavior simulation**

The behavior of ants described above has influenced meta-heuristics such as the Ant Colony Optimization (ACO) that is used to solve complex problems which will be discussed further in the following section [Tarasewich02]. The idea is for simulated ants to work together towards solving common problems. Artificial ants differ in their behavior than real ants. Of the differences [Blum05]:

- Real ants move asynchronously, while simulated ants are synchronized
- A Simulated ants' way of laying pheromone is by updating a local routing table
- The evaluation of the behavior of real ants is based on the relative goodness of paths discovered. Artificial ants evaluate their solution based on a quality measure that is domain specific and accordingly determines the amount of pheromone deposited.

A classical problem is the Traveling Salesman Problem. The idea is that ants cooperate and share information together towards building a solution to the problem. Ants work

iteratively towards adding more cities to partial solution using pheromones deposited by other ants [Dorigo97].

Ants' behavior has also worked its way into solving problems relating to efficient ways of doing business. As an example Southwest Airlines was unable to find a way to efficiently use its cargo space on its planes. By looking at how ants forage for food, they established a system that has cut transfer rates by 80% and reduced work loads at stations by 20% [Bonabeau01].

Ant behavior heuristics have also been used to minimize the cost of vehicles delivering goods to a series of customers from a central location. They have also been used to schedule efficient use of factory equipment. Phone companies have also used routing methods based on Ant behavior to route phone calls through circuit switched networks. Research is also being done on other Ant based routing methods to route traffic in packet switched networks [Bonabeau01] [Tarasewich02].

## 2.2 Metaheuristics: An Introduction

The term *metaheuristic* is made up of two Greek words: *meta* which means "higher level" and *heuristic* which means "to find" [Blum03].

Ant colony optimization, genetic algorithms, tabu searching, hill climbing and simulated annealing are just many of the metaheuristics used in recent days to take on combinatorial optimization problems. They can be described as high level procedures that coordinate simple heuristics to find a solution to these combinatorial optimization problems [Ribeiro07], some of which will be discussed in the next section. These metaheuristics and others not mentioned here are the most effective ways to solve a wide selection of problem.

Furthermore, metaheuristics can be defined as "an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring

and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions" [Osman96] [Merkle02].

Some of the properties which characterize metaheuristics are outlined in [Blum03]:

- Metaheuristics employ strategies to find optimal solutions
- Metaheuristics are non-deterministic and are approximate
- Metaheuristics are abstract
- Metaheuristics are not problem specific and can make use of domain-specific heuristics controlled by the higher-level strategy
- Metaheuristics make use of memory of previous searches to guide the search

## 2.3 Ant Colony Optimization

As mentioned before, Ant Colony Optimization (ACO) is a metaheuristic inspired by the behavior of ants as they forage for food. Although ant behavior had been considered previously for other metaheuristics, ACO was formalized at the hands of Dorigo and his co-workers in 1999 [Blum05].

ACO can be described graphically in the figure 3 [Blum05]:

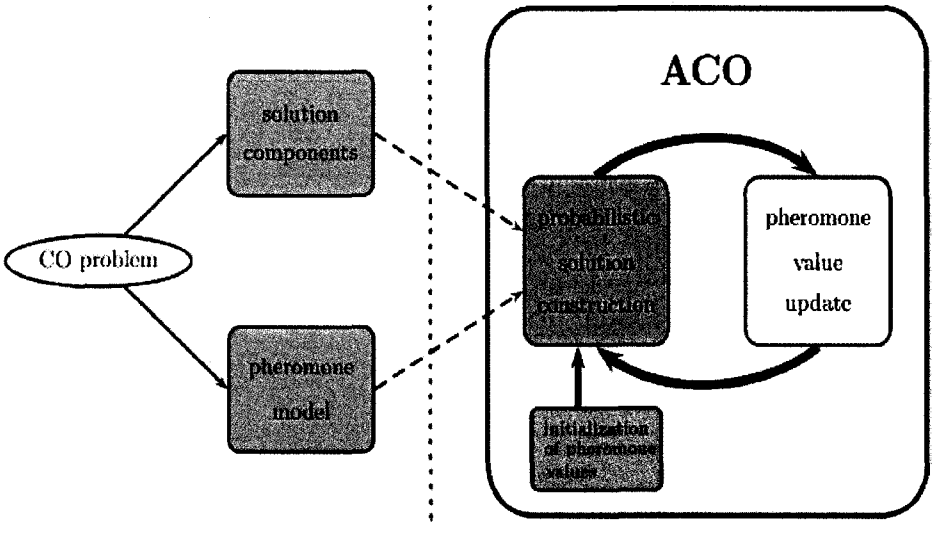


**Figure 3 The ACO metaheuristic**

Given a certain combinatorial optimization (CO) problem, a set of solution components are defined that includes all possible solutions for the CO problem. Next a pheromone model is defined which includes all pheromone reward values for the possible solutions as described in [Dorigo07].

The ACO algorithm follows these simple steps [Blum05] [Dorigo07]:

```
InitializePheromoneTrail()
While termination conditions are not met do
      SolutionConstruction()
      UpdatePheromone()
End while
```

`InitializePheromoneTrail()` initially assigns equal pheromone values to all solutions in the solution component.

`SolutionConstruction()` constructs solutions iteratively from the set of solution components. Each solution is selected iteratively according to a probabilistic rule. The solutions are evaluated according to a heuristic that is specific to the problem at hand which corresponds to a pheromone reward value. For example in the Traveling Salesmen Problem, each solution component is the set of links that connect the current node to all possible attached nodes. Each solution constructed is weighed according to its length or latency from the start node.

`UpdatePheromone()` is the function that deposits artificial pheromones on the good solutions and removes or evaporates pheromones from the bad ones.

ACO is explained in much greater detail in [Dorigo05]. It is also explained in [Sun04] as applied to the Traveling Salesman Problem.

## 2.5 AntNet

AntNet is an ACO based algorithm for routing in mobile communications networks. In this algorithm, a set of artificial ants work concurrently towards finding optimal routing paths. As described by the function of ACO, artificial ants iteratively work on finding a solution to the routing problem and communicate with other artificial ants by way of stigmergy [DiCaro].

The details of the early inception of the AntNet algorithm, namely AntNet1.0, are provided in [Baran00] and [Sosa00]. AntNet2.0 is described in [Sosa0], and it is this version which is commonly known as AntNet.

The AntNet algorithm functions and characteristics can be described as follows [DiCaro98] [Dhillon07] [Zhang04]:

- At regular intervals mobile agents, or forward ants, are launched towards the destination node
- As forward ants travel towards the destination node, they keep track of the path they take as well as traffic conditions. Every node visited is pushed onto a stack in the forward ant's memory
- The node the forward ant selects to move forward to is selected according to a link probability distribution
- If the forward ant visits a node it had previously visited, a cycle is detected. The cycle is removed from the stack. If the cycle is longer than half of the total route, the forward ant is destroyed.
- When the destination node is found, the destination node generates another mobile agent, also known as a backward ant, copies all the information from the forward ant and destroys it.
- The backwards ant takes the same route as the forward ant but in the backwards direction towards the source node.
- At each node on the way, the backward ants updates the link probabilities using

the following equation [Zhang04]:

$$p_m \leftarrow p_m + r(1 - p_m), p_n \leftarrow p_n - rp_n, n \in N, n \neq m$$

Where m is the node the backward is coming from m Є N where N is the set of neighbor nodes.

The value of r Є [0, 1], can be chosen in two ways:

- $r$ = constant value. All paths regardless of their goodness are rewarded equally. The result will be that ants moving along longer paths will have the same effect on routing tables as the ants moving along shorter ones. While the performance of this reward scheme may be good, the heuristic of the length of the path has to be used to determine the reward value [DiCaro98].
- $r$ can determined according to the trip time or the number of hops to the destination node. $r$ can be expressed as follows:

$$r \leftarrow k_1\left(\frac{C_{inf}}{C}\right) + k_2\left(\frac{C_{sup} - C_{inf}}{(C_{sup} - C_{inf}) + (C - C_{inf})}\right)$$

Where:

$$C_{inf} = \min(W)$$

$$C_{sup} = \mu + z(\sigma/\sqrt{M})$$

$$k_1 + k_2 = 1$$

And the mean and variance are calculated as such:

$$\mu \leftarrow \mu + \eta(C - \mu), \sigma^2 \leftarrow \sigma^2 + \eta((C - \mu)^2 - \sigma^2)$$

$C$ is the cost of the current path either expressed in time or number of hops. $W$ is a sliding window of size $M$ that hold last $M$ number of paths while $\eta$ is calculated using the formula Min(5/$M$, 1).

- The launch interval is updated to increase the launch interval if the path has a relatively low cost and reduce it otherwise

### 2.5.1 Summary Evaluation of AntNet

AntNet has been evaluated against many shortest routing algorithms such as OSPF and others mostly in simulations that induce congestion and link failure scenarios.

AntNet's performance has been compared against Dijkstra's shortest path algorithm in those situations. AntNet has displayed good performance in smaller random networks but quickly degrades as the network size and link density increases [Dhillon00].

Compared with OSPF, [Strobbe05] has shown that AntNet is more adaptive to routing situations such as link failures and congestion and can quickly adapt in contrast with OSPF. Results have also shown that AntNet is far superior in terms in success rate [DiCaro98].

### 2.5.2 Improvements Introduced to AntNet

Many have attempted changes and improvement to AntNet to try and achieve better overall performance. Some researchers have introduced the idea of adding popular destinations to routing tables at intermediary nodes which has increased the success rate of AntNet [Yun04].

Researchers have also suggested setting an upper limit to the age of the forward ant. Suggestions have been made in [Tekiner] to kill forward ants that that have an age larger than twice the number of nodes in the network. It also suggests putting an upper and lower bound on the value of $r$ to limit the effect of fluctuations in a network.

Further, in [Zhang04], forward ants are given sensory abilities. They are able to sense the geographic location of the destination dramatically increasing success rate and energy efficiency of the algorithm. Another technique suggested in [Zhang04] is making use of broadcast channels in networks. When a forward ant is launched towards the destination,

it informs its neighbors to find the destination and its neighbors in turn tell their neighbors till the destination is found and a trail of pheromone is left towards to the destination. This has improved the latency quite a bit.

## 2.6 Another Metaheuristic: Genetic Algorithm

Genetic algorithm is another metaheuristic used for solving combinatorial optimization problems. They draw their inspiration from the biological genetic processes of live organisms.

Similar to ACO, genetic algorithms initially create a population of solutions, also known as chromosomes, and determine a way of evaluating the goodness of each solution, for example the path length between two nodes. The most 'fit' solutions are then reproduced through genetic operators and a new set of solutions is created. The 'unfit' solutions are removed and the fit solutions are allowed to reproduce once more. This allows the good characteristics in the fit solutions to be mixed with others and to be extended over several generations [Beasley93].

[Ghoshray95] and [Mitchell96] explain some of the basic genetic operators used in the reproduction cycle such as selection, mutation and crossover.

*Selection* is the process of selecting fit chromosomes and removing others.

*Crossover* is swapping of data between two parent chromosomes along a selected point in both parents resulting in two children which share data with both their parents. This kind of crossover operator is known as single-point crossover.

Other types of crossover operators exist such as multi-point crossover where multiple points are selected on both parents. Multi-point crossover is generally better than single-point crossover. Another sort of crossover is the uniform crossover where children are created by copying corresponding bits from one parent or the other according to a

randomly generated mask [Bull93]. Figure 4 demonstrates the crossover operation [Crossover].

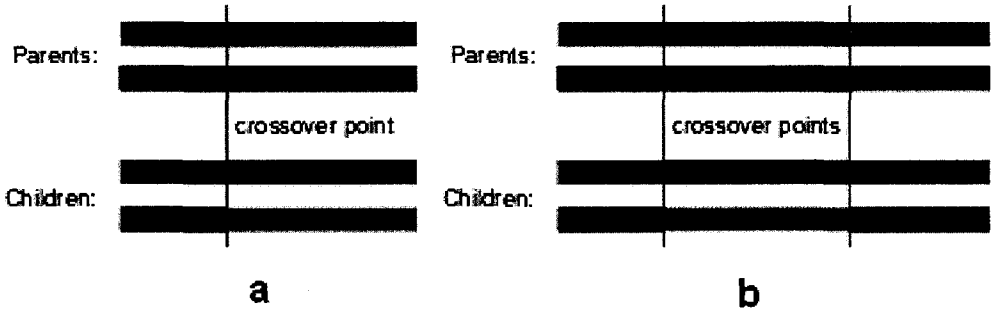


**Figure 4 Crossover operation**
**a) Single-Point Crossover b) Multi-Point Crossover**

*Mutation* is an operator that randomly flips bits in the parent chromosome resulting in a modified child.

Applications of genetic algorithms have extended to include path planning for mobile robots. Genetic operations such as mutation and crossover are being utilized to help mobile robots find near-optimal paths [Hu04].

Genetic algorithms have been compared against ACO algorithms and it has been found that they are indeed similar in many aspects. It has been shown that they perform just as well also as ACO algorithms in solving the Traveling Salesman Problem which lends significance to crossover and mutation operators as being effective in searching for promising solutions [Gomez04].

Further, algorithms have been formulated by combining genetic algorithms with other metaheuristics. Some worth noting is Ant System with Genetic Algorithm (ASGA) [White98], Genetic Ant Routing Algorithm (GARA) [Cheng03] and a genetic algorithm combined with simulated annealing to help generate back-up routing tables in telecommunication networks [He00].

Ant System with Genetic algorithm (ASGA) functions primarily as a genetic algorithm. Pheromone and cost parameters are generated randomly and assigned to the first generation of agents. Agents move forward using Ant System in solving the problem and their paths are reinforced using pheromone. Another generation is created of pheromone and cost parameters randomly. The previous generation and the newly generated one are sampled and manipulated using crossover and mutation operators to form the next generation. ASGA has been compared to Dijkstra's shortest path algorithm and has had favorable results [White98].

Genetic Ant Routing Algorithm (GARA), on the other hand, performs single-point crossover operations after every iteration of AntNet on two of the discovered paths. The two paths are chosen according to a probability distribution and the resulting path is reinforced using a special reinforcement ant. It has been found through simulations that GARA outperforms AntNet in many aspects such as success rate, latency and convergence speed [Cheng03].

# CHAPTER 3

# PROPOSED ALGORITHMS

In this section, three AntNet based algorithms for routing in telecommunication networks are proposed. The algorithms are to be mainly used in route discovery within networks that are dynamic. A Source node wishing to transfer information to a single destination invokes these routing algorithms in order to locate the destination and discover a path to it. These algorithms are based on the AntNet algorithm discussed in the previous chapter and are referred to here as: Improved AntNet, Pharaoh Routing, and Hybrid AntNet/Genetic algorithms.

## 3.1 Improved AntNet

Improved AntNet is based entirely on AntNet, described in the previous chapter. Inspiration for this algorithm came from the fact that AntNet has a very poor success rate in comparison with the improved algorithms presented in [Zhang04]. Also, it has been stated several times that AntNet converges quite slowly especially in larger networks [Chang03]. What single modification to AntNet can increase the success rate and reduce convergence time together?

Forward ants in this algorithm are given forward sensing capabilities. They no longer imitate ants that belong in natural habitat, but they are now intelligent ants with sensors. They are able, from any node in the grid network, sense whether connected nodes have

already been previously traversed in their journey towards the destination node or not. The behavior of backwards ants and the mechanism of how link probability distributions are updated in this algorithm are unchanged.

Figure 5 depicts an example of a forward ant in a grid network searching for the destination node. The purple paths mark the paths taken so far. The nodes visited at that point in the journey are also marked with an X. The red arrows in the figure point to the possible next nodes the forward ant will select to move to. The forward sensing capability prevents it from moving back towards nodes that have been previously visited.

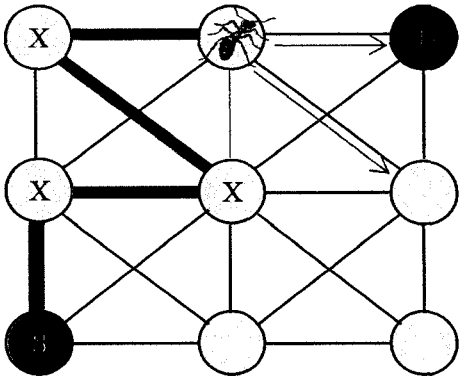


**Figure 5 Forward sensing by a forward ant**

The algorithm of Improved AntNet is described as follows:

1. At some interval that may vary with time, forward ants are created at the source node and sent towards the destination. They move in parallel but independent of each other in finding the destination.
2. Forward ants select the least cost path joining the source and destination. The next node to visit is selected according to a link probability distribution. At the start, each link share equal probability.
3. Forward ants maintain a taboo list of nodes already visited. Before moving forward towards a node, the forward ant checks if that node exists in the taboo list. If it does,

the forward ant selects the next link to an unvisited node with the least probability distribution.

4. If a forward ant reaches a node from which all neighboring nodes exist in the taboo list, it dies.
5. Once the destination is reached, a backward ant is created and sent back along the same path to the source node updating the link probability distribution at every hop exactly as in AntNet. Once it reaches the source node, it dies.

With giving forward ants this forward sensing capability, we are allowing more forward ants to pass through as many nodes as possible in search of the destination node. With this capability, just about every node in the grid network is visited in search of the destination. Although this may cause an increase in power consumption, it is most certainly guaranteed this algorithm yield a higher success rate than AntNet.

Forward sensing also eliminates the possibility loops are generated by forward ants as ants will never revisit nodes they have already traversed. This helps in reducing convergence time of the algorithm.

The functional flow chart of the forward ant in the Improved AntNet algorithm is shown in the figure 6.

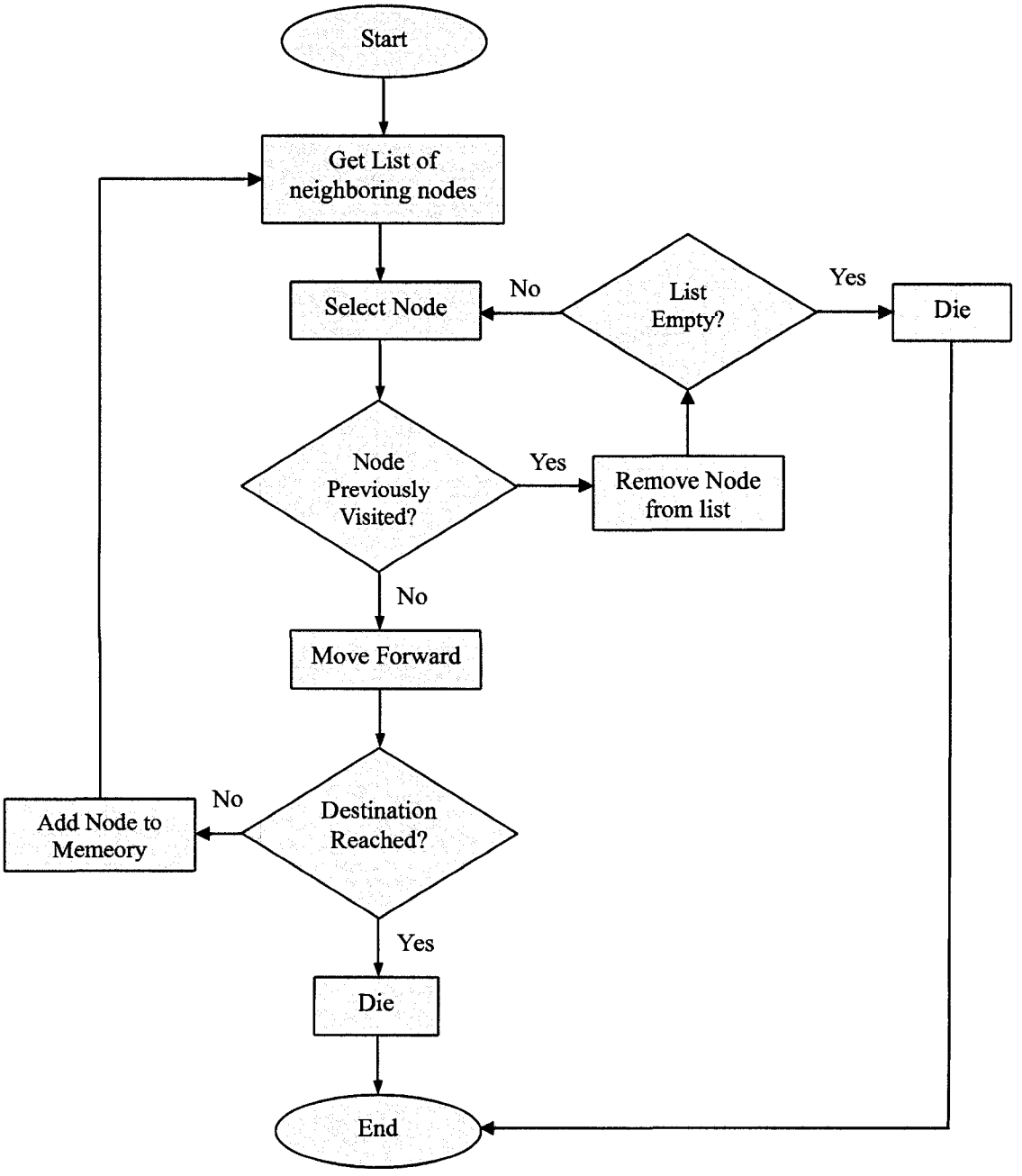


**Figure 6 Improved AntNet forward ant flow chart**

The flow chart illustrates the life of a forward ant in Improved AntNet from creation till death. More details on the individual instructions and actions are provided in the following chapter where the pseudo-code of the entire algorithm will be discussed.

## 3.2 Pharaoh Routing

In ant colonies, pheromones are used to attract foragers to trails that lead to food. Recently, researchers at the University of Sheffield have discovered that not only do Pharaoh ants (Monomorium pharaonis) colonies lay pheromones that lead to food but also lay negative pheromones on entry points to paths that may lead away from the food to repel other foragers from those paths [Robinson05]. These findings have been further reinforced in [Robinson07]. It has been shown that this phenomenon increases the effectiveness of foraging. Similarly, using such element in the area of network routing can increase the efficiency of route discovery methods and decrease convergence times.

Motivated by these findings, the Pharaoh Routing algorithm that is based on this phenomenon is considered. Similar to Improved AntNet, the Pharaoh Routing algorithm is based on AntNet. Moreover, forward ants' in this algorithm imitate the behavior of the Pharaoh ant. In Pharaoh Routing algorithm, whenever a loop is discovered by a forward ant, negative pheromone is placed at the entry point of that loop such that other forward ants do no waste time traversing those routes.

Figure 7 illustrates the way Pharaoh ants work in marking loops or paths that may lead further away from the destination. The paths marked in purple are the loops ants are to avoid. One is an actual loop and the other is a dead end node to avoid, both previously traversed by a forward Pharaoh ant.

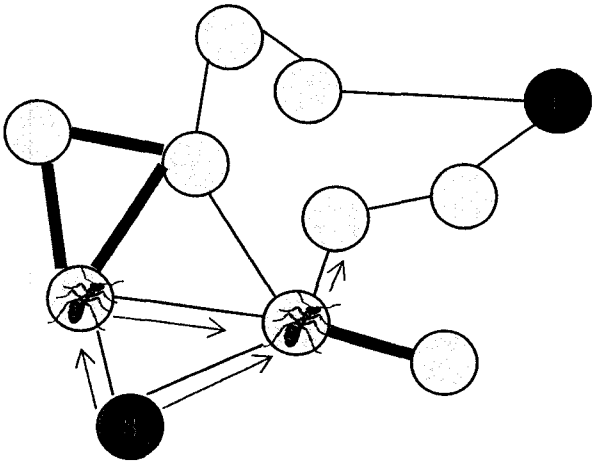

**Figure 7 Pharaoh ant example**

The Pharaoh Routing algorithm is described as follows:

1. Similar to AntNet and Improved AntNet, forward ants are created at the source node and sent towards the destination node at some interval.
2. Forward ants select the least cost path joining the source and destination. The next node to visit is selected according to a link probability distribution. At the start, each link share equal probability.
3. Forward ants maintain a taboo list of nodes already visited. If a node is visited that exists in the taboo list, a loop is discovered. If the loop is longer than half the path taken by the ant, the ant dies. Otherwise, the loop is removed from the forward ants' memory and the path is marked with a negative pheromone such that other forward ants avoid taking that route.
4. Once the destination is reached, a backward ant is created and sent back along the same path to the source node updating the link probability distribution at every hop exactly as in AntNet. Once it reaches the source node, it dies.

Having forward ants behave similarly to Pharaoh ants increases the effectiveness of the foraging behavior in grid networks. Unrewarding routes are marked such that other forward ants do not waste time going through them in their journey towards the destination. More forward ants are finding the destination in less time. As a result it is expected that the most noticeable improvements in performance of this algorithm is the increased success rate and reduced energy consumption.

The functional flow chart of the forward ant in the Pharaoh routing algorithm is in shown in figure 8.

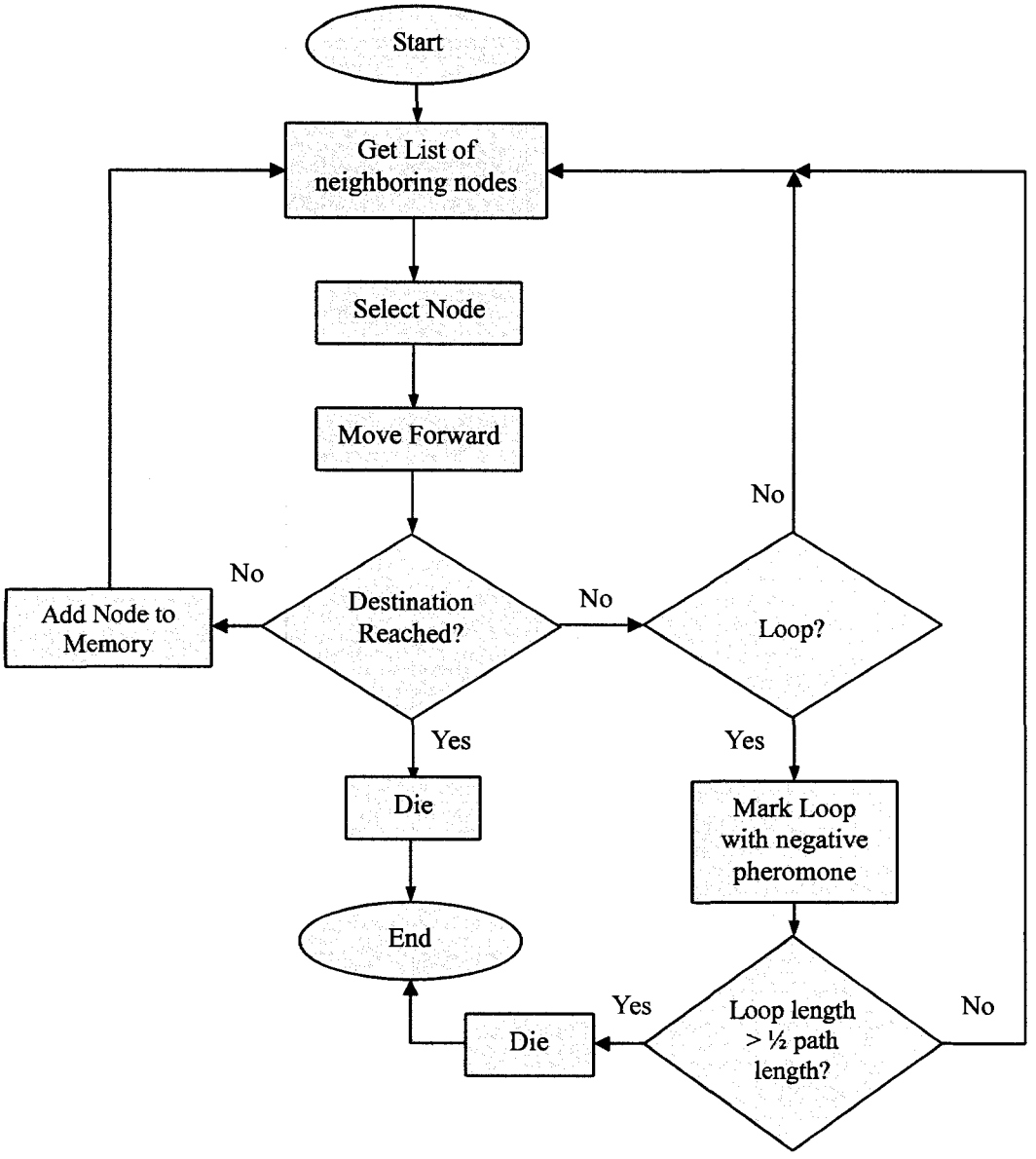


**Figure 8 Pharaoh Routing forward ant flow chart**

## 3. 3 Hybrid AntNet/Genetic Algorithm

Research has been done in multi-agent metaheuristics and frameworks have been presented for the development of hybrid algorithms [Milano04]. These frameworks ease the work done for constructing algorithms that are based on multiple metaheuristics. An example of a hybrid algorithm that has been developed is Coor2Met. It brings together two metaheuristics: genetic algorithm and a tabu search [Cadenas07]. Another example is the Genetic Ant Routing Algorithm (GAR) that was presented in the previous chapter.

Individually, it has been observed that AntNet and genetic algorithms both are good performing algorithms in the area of routing. This was an incentive to try and use the good characteristics of genetic algorithms to minimize the impact of the shortcomings of AntNet and improve the overall performance.

Further after introducing Improved AntNet and Pharaoh Routing, we now introduce an algorithm that is a cross of two metaheuristic methods. This algorithm is part AntNet and part genetic algorithm. Forward ants behave as they do in AntNet, as well as backward ants. Periodically, genetic operators are invoked on paths discovered by forward ants and when it is deemed that a shorter path can be achieved as a result, a reinforce ant is launched along that resultant path.

To be specific, the genetic operator used by the Hybrid AntNet/Genetic algorithm is single-point operation on paths. The single-point crossover operation is done by selecting a common node between two paths. All nodes beyond that single node in either paths are swapped resulting in two child paths. The figure 9 illustrates a simple example of a single-point crossover operation on two paths.

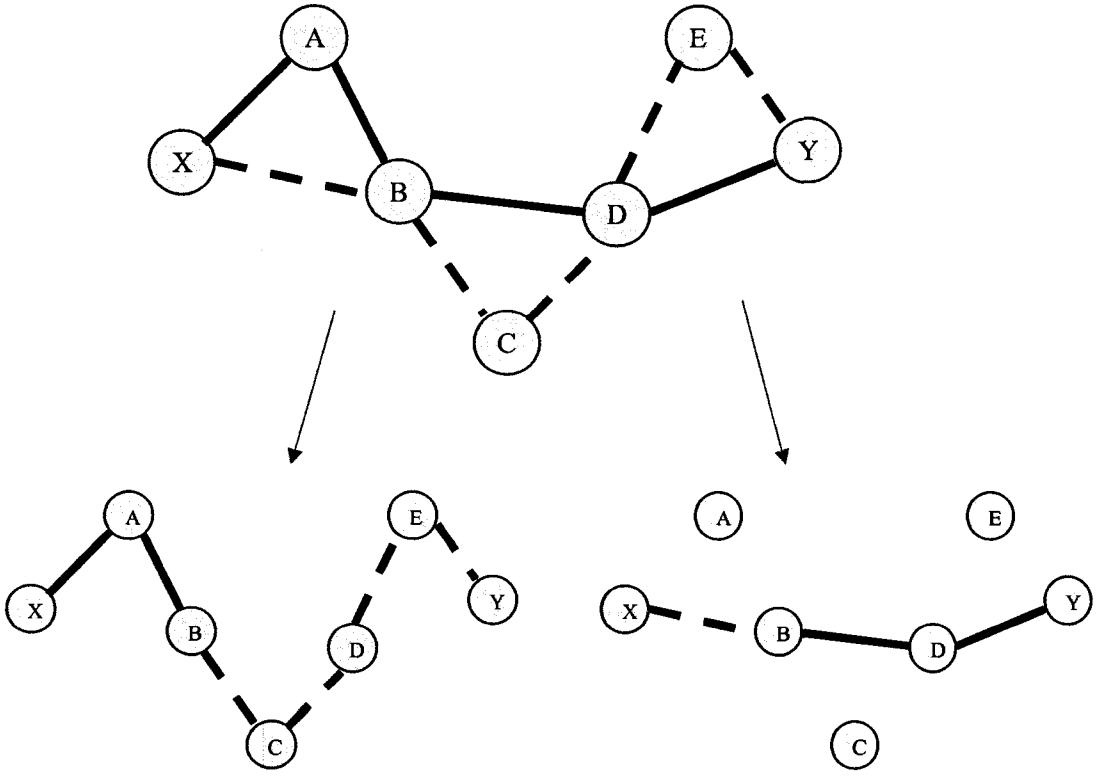


**Figure 9 Single-point crossover example**

In the figure, two paths exist, the path denoted by the solid line and another by the dotted line. Both paths lead from points *X* to *Y*. We can see both paths intersect at points *B* and *D*. Point *B* was selected as the point where the data in both paths would be swapped. The resultant is two child paths *X-A-B-C-D-E-Y* and *X-B-D-Y* displayed above. Path *X-B-D-Y*, the shorter of the two paths is the one the algorithm selects.

The Hybrid AntNet/Genetic algorithm is described as follows:

1. As in AntNet, forward ants are created at the source node and sent towards the destination node at some interval.

2. Forward ants select the next node to visit selected according to a link probability distribution. Forward ants remember the nodes it has visited and removes loops when necessary.
3. Once the destination is reached, a backward ant is created and sent back along the same path the forward ant took to the source node updating the link probability distribution at every hop exactly as in AntNet. Once it reaches the source node, it dies.
4. The source node maintains a list of paths updated by backwards ants. Every $n$ paths the source node selects two shortest paths in that list and performs single-point crossover operation on them resulting in two child paths. The shortest of the two child paths is chosen and the list of paths discovered at the source node is cleared.
5. If that child path is shorter than the current shortest path in AntNet, it is reinforced $x$ times using a special reinforcement ant. The reinforcement ant moves along the child path towards the destination and returns to the source in the opposite direction updating the probabilities at each node as in AntNet.

In hybrid AntNet/Genetic algorithm, the forward ant behaves exactly as in AntNet described in the previous chapter. For that reason we will not reiterate. Instead, the flow chart in figure 10 will describe the functions performed at the source node including the genetic single-point crossover operation mentioned earlier.

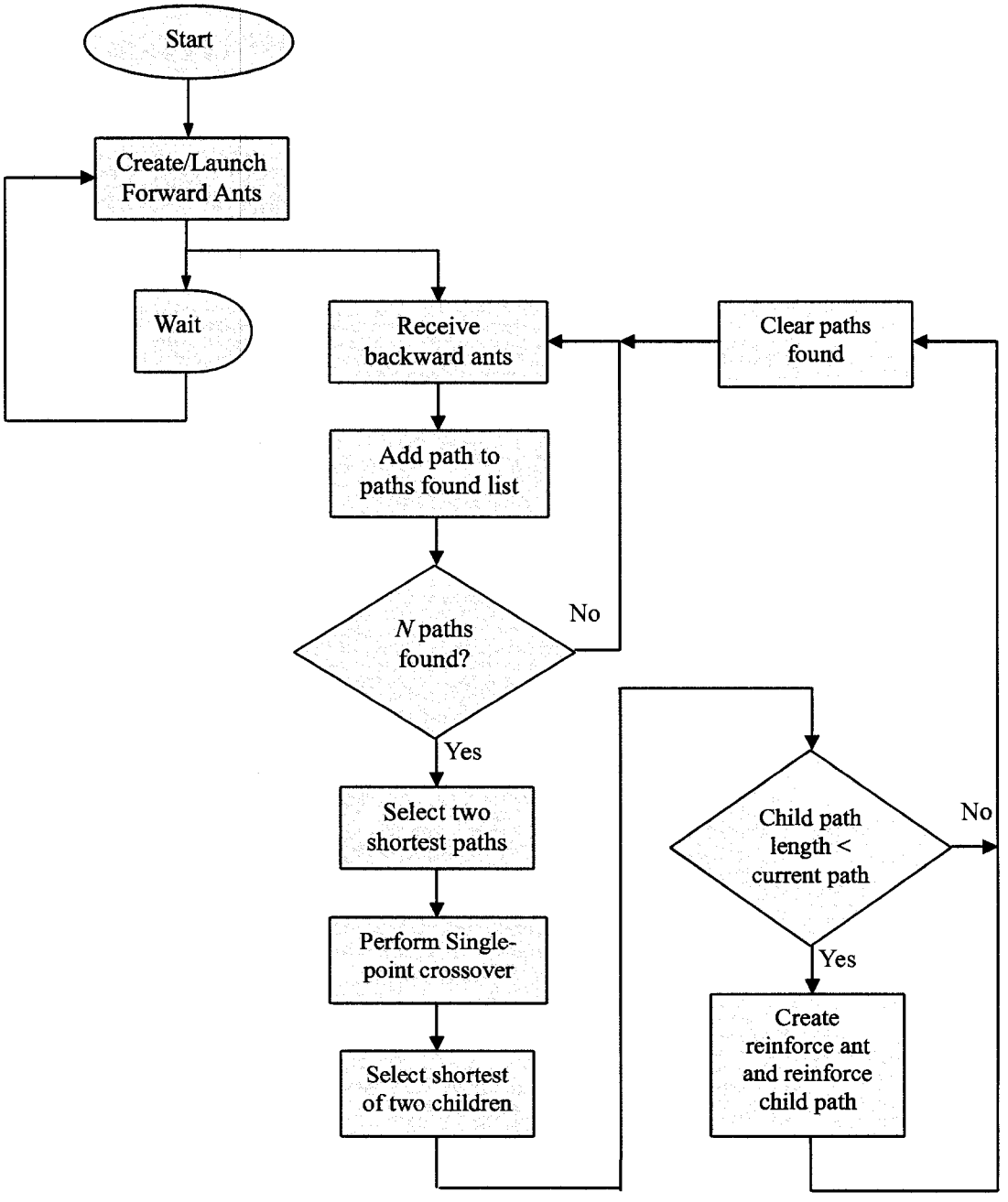


**Figure 10 Hybrid AntNet/Genetic source node operation flow chart**

# CHAPTER 4

# IMPLEMENTATION

## 4.1 Implementation Basic

### 4.1.1 NetLogo

NetLogo is a modeling tool written in Java that allows it's users to simulate natural phenomena. It offers an integrated multi agent environment that is easy to learn and use. The programming language allows for flexibility for modeling any social or natural science behavior in domains such as biology, chemistry, physics, and economics. NetLogo was authored by Uri Wilensky, director of Northwestern University's Center for Connected Learning and Computer-Based Modeling [NetLogo].

NetLogo is particularly a good fit for designing complex simulations. Developers through procedures give instructions to many agents, or in our case ants, all operating independently and concurrently. It is widely used in academia and due to its popularity and ease of use; it was chosen to simulate the ant routing algorithms proposed in this thesis.

The NetLogo development environment interface offers many features that make creating models easy. The interface contains an area for viewing models in 2D or 3D. There is a command center for executing commands during simulations. There is an area around the model viewing window that can be used for interface builders such as: sliders, buttons,

monitors and plots, all of which we make use of in the. Sliders and buttons were made use of for controlling many aspects as displayed on the left hand side of the picture. Plots and monitors were used as well to gather data and information that was later exported to spreadsheets.

Several important NetLogo programming features were utilized in the implementation of the algorithms. Some of the more important ones worth mentioning:

**Procedure:**

There are Commands and Reporters in Netlogo. Commands are built in actions that agents perform. Reporters compute and return values. Procedures are commands and reporters you define just like C-style functions.

**Agents and Agentsets:**

NetLogo simulations are constructed using turtles, patches and the observer. The observer can be considered to be "the world" with patches and turtles placed in it. Turtles have coordinates and can move around within the world. Patches have coordinates but are stationary. We utilize only turtle agents in our simulation. The following sections will demonstrate how turtle agents were used to derive breeds of ants. As for Agentsets, as the name indicates they are sets of agents. They can represent turtles or patches but not both. Agentsets can also be constructed to represent subsets of turtles or patches.

**Breeds**

NetLogo allows for the creation of custom turtle-like breeds. They inherit all the primitives the turtles have, along with new breed only primitives. They also inherit all the properties of agentsets. Simulators can go head and create their own breeds and specify how they are supposed to behave.

In the simulations in this thesis, ant, node, and link breeds were used. The figure 11 is an example the ant breed used in the implementation.

```
breed [ ants ant ]

ants-own[
   node current-node
   node-path
   taboo-list
   destination-reached
   current-cost
  ]
```

**Figure 11 Ant breed in NetLogo**

## 4.1.2 Environment Design

The environment used to test the performance of the algorithms was modeled using NetLogo's graphic design tool. Through NetLogo, network parameters were varied in order to study its effect on the overall performance of each algorithm. The simulation was run on an *n*-dimensional node grid with number of nodes equal to *n x n* where the source and destination nodes are on either corners of the network as displayed in the figure 12. The green node indicates the source node and the blue indicates the destination node.

The below network parameters were altered in the simulation experiments:

- **Number of Nodes**: The number of nodes in the simulation is controlled by altering the value of the *n*
- **Node Connectivity**: The number of links between each node is controlled by altering the value of the radius around which each node establishes its connections with other nodes. The nodes are placed at equal distance on the horizontal and vertical plane in the simulation. The radius value is multiples of that distance between two nodes. All nodes in the simulation share the same node connectivity value.

Figure 12 is an example of a grid network with n = 3 and *r = 2*. All nodes are connected to nodes within distance = *2 * d*.

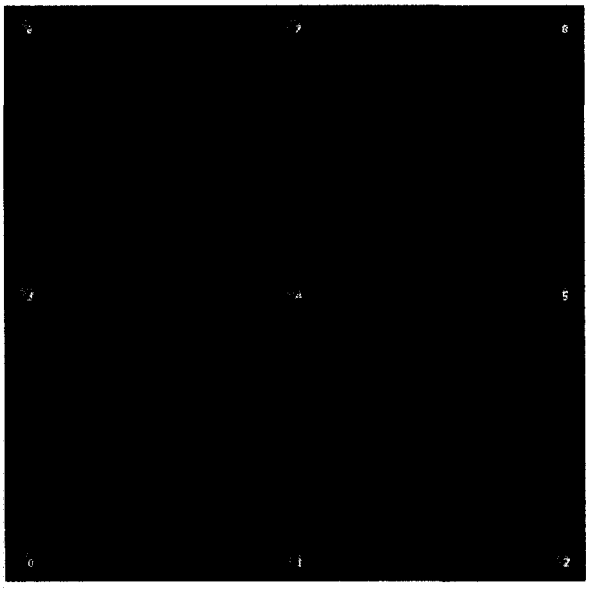

**Figure 12 Example grid network simulation - n = 3 and r =2**

In order to conduct the simulations more efficiently, the environment was modeled using the NetLogo's graphical user interface tools. Figure 13 displays the environment used for simulating the grid network.

The user interface consists of two parts: the black canvas on the right hand side inside which the grid network is simulated and the controls on the left hand side through which the simulation is controlled and monitored.

The slider at the top of the black canvas is used to speed up the execution of the simulation. This is done to speed up the simulation execution. The controls on the left hand side include a slider to adjust the number of ants to be launched every launch interval. The *setup* button is used to redraw a new grid network and clear all plots and monitors between simulation runs. The two *go* buttons are used to launch the simulation. The *go* button with the green circular arrows runs the procedure *go* in an infinite loop till interrupted by the user by depressing the same button again. The other *go* button executes the same procedure once only, used mostly for debug purposes. *Send-data* is used to export the data in the plots to excel spreadsheets once the simulation has completed.

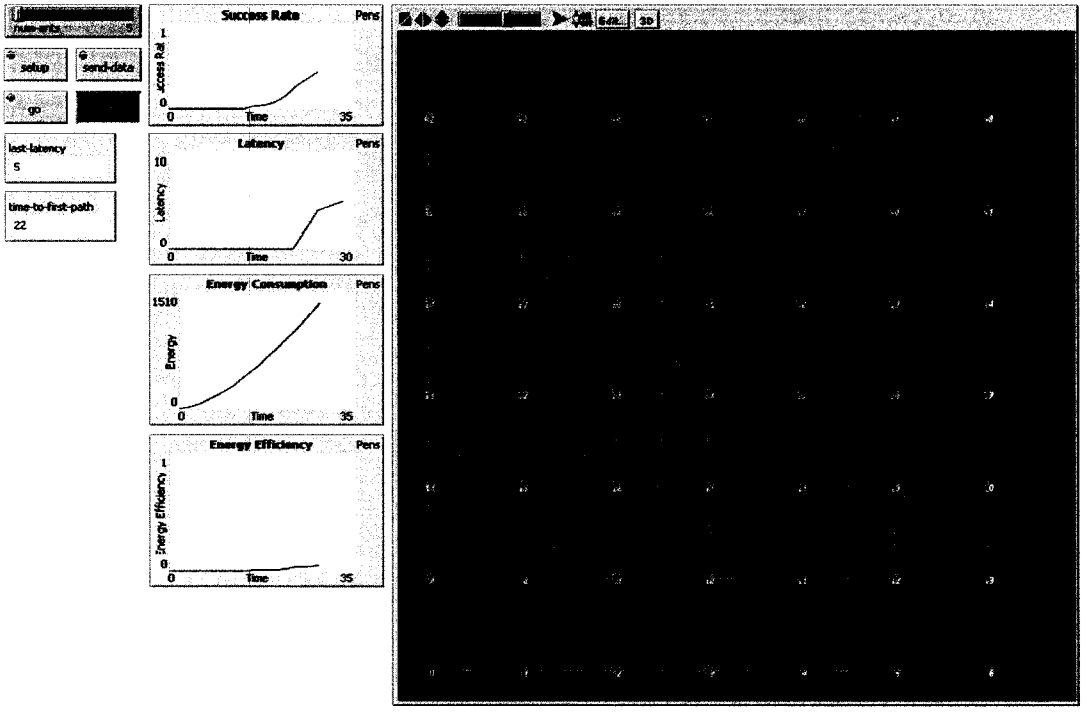


**Figure 13 NetLogo GUI used for simulation**

# 4.2 Implementation Details

## 4.2.1 Improved AntNet

In summary, the forward ant agent in Improved AntNet does the following for the duration of its life:

1. Select a prospective next node
2. Verify it is not in the taboo list
3. Move towards the selected node pushing it in it's memory

Below is the pseudo-code of the forward ant agent:

```
Foreach ForwardAnt
{
      Add CurrentNode to NodePathList;
      NextNode := SelectNode(Destination, RoutingTable);
      TempRoutingTable := RoutingTable;
      While (NextNode is member of TabooList && TempRoutingTable is not
```

```
        empty)
        {
                Remove NextNode from TempRoutingTable;
                If (TempRoutingTable is Empty)
                        ForwardAnt Die;
                NextNode := SelectNode(Destination, TempRoutingTable);
        }
        MoveTowards(NextNode);
        CurrentNode := NextNode;
        If(CurrentNode == destination)
        {
                Create BackwardsAnt and copy NodePathList
                ForwardAnt Die;
        }
}
```

The backwards ant agent performs the following for the duration of its life:

1. Pop the next node from the node list inherited from the forward ants memory
2. Move towards the selected node
3. Update the routing table

The pseudo code for the backwards ant agent is as follows:

```
Foreach BackwardsAnt
{
        NextNode := Last Node in NodePathList;
        NodePathList := NodePathList but last Node;
        MoveTowards(NextNode);
        UpdateRoutingTable(CurrentNode, NextNode);
        CurrentNode := NextNode;
        If (CurrentNode == SourceNode)
                BackwardsAnt Die;
}
```

The particulars on `SelectNode()` and `UpdateRoutingTable()` are not mentioned as they have been explained in great detail in the previous chapter as part of the introduction to AntNet. For details on how it was implemented in NetLogo consult the implementation of AntNet in appendix A.

At the source node, the following pseudo code briefly describes what takes place:

```
Every LaunchInterval
{
      CreateForwardAnts();
      LaunchForwardAnts(Destination);
}
```

As mentioned in the implementation basics section, NetLogo allows for concurrent executions of procedures across agents. All the mentioned above segments for each agent set is executed at the same time and in parallel.

### 4.2.2 Pharaoh Routing

In summary, the Forward ant agent in Pharaoh Routing does the following for the duration of its life:

1. Select a prospective next node
2. Move towards the selected node pushing it in it's memory
3. Verify that the node visited is not in taboo list
4. if loop exists, remove loop from ant memory and mark loop with negative pheromone

The pseudo-code of the forward ant agent in Pharaoh Routing follows:

```
Foreach ForwardAnt
{
      Add CurrentNode to NodePathList;
      NextNode := SelectNode(Destination, RoutingTable);
```

```
        TempRoutingTable := RoutingTable;
        while (NextNode is marked by negative pheromone)
        {
              Remove NextNode from TempRoutingTable;
              if (TempRoutingTable is Empty)
                    ForwardAnt Die;
              NextNode := SelectNode(Destination, TempRoutingTable);
        }
        MoveTowards(NextNode);
        CurrentNode := NextNode;
        if (CurrentNode == destination)
        {
              Create BackwardsAnt and copy NodePathList
              ForwardAnt Die;
        }
        if (CurrentNode is member of NodePathList)
        {
              LayNegativePheromone(CurrentNode, NodePathList);
              RemoveLoop(CurrentNode, NodePathList);
        }
}
```

`LayNegativePheromone()` procedure will move the ant across the loop laying a colored marker (negative pheromone) that other ants will look for and avoid in their search of the destination.

In `RemoveLoop() procedure`, given a list of nodes and the current node which is the starting and ending point of the loop, will remove the loop by creating a sub-list ranging between the first node in the list till the first occurrence of CurrentNode.

The functions of the backwards ant and source node in the Pharaoh Routing are identical to that of Improved AntNet and will not be mentioned again for the sake of avoiding redundancy. Their function is described in the previous section.

### 4.2.3 Hybrid AntNet/Genetic

As mentioned previously when introducing the Hybrid AntNet/Genetic algorithm, this algorithm is largely identical to AntNet. Forward ants search for the destination and backwards ants update routing tables as they normally would.

The only difference in this algorithm is the backward ant agent maintains a copy of the path taken from destination back to the source. The source node extracts that information as it receives backward ants. What the source node in this algorithm does with this information is the difference between this algorithm and others.

The source node performs the following for the duration of the simulation:

1. Maintain a list of paths to the destination
2. Every *n* number of paths recorded, perform single-point crossover on the two shortest paths
3. Select the shortest of the two children paths that result from the crossover
4. If the length of the chosen child path is less than the current shortest path discovered by AntNet, reinforce that child path.

The pseudo-code below highlights what takes place at the source node. As usual, every time interval, forward ants are created and dispatched towards the destination:

```
Every LaunchInterval
{
      CreateForwardAnts();
      LaunchForwardAnts(Destination);
}
```

The below occurs concurrently as the above segment and is halted when the simulation ends:

```
Repeat
```

```
{
      ReceiveBackwardAnts();
      Add NodePathList to DiscoveredPaths;
      if (length of DiscoveredPaths > n)
      {
            Select two shortest paths from DiscoveredPaths;
            Perform single-point crossover on both paths;
            ChildPath := shortest of two children;

            if (length of ChildPath < current AntNet latency)
                  Reinforce(ChildPath, x);
            Clear DiscoveredPaths;
      }
}
```

The `Reinforce()` procedure will create a special type of ant that will traverse the given path to the destination and reinforce it with pheromone on the way back to the source node. This is repeated `x` number of times.

# CHAPTER 5

# PERFORMANCE ANALYSIS

After presenting Improved AntNet and Pharaoh Routing in the previous two chapters, this chapter will feature the experimentation results on them. The data collected in these experiments will be presented along with a comparison with AntNet. The findings in this chapter will used to draw conclusions on the performance of the two algorithms. Using NetLogo, a simulation environment was developed to simulate a grid network. Using the built in features of NetLogo, the number of nodes in the network was varied as well as the number of links between them to study the effects of each variable on the performance of each algorithm. The environments used do not present real life scenarios as they consider only ideal conditions. It is assumed that all the networks simulated are absent of any other traffic data and that the source and destination nodes remain static.

## 5.1 Metrics

- **Latency:**

The latency is defined as the time it takes for an ant to go from the source node to the destination node. In the simulations presented here, it is assumed that the inter-node latency is uniform across the grid network; therefore the latency is expressed as the number of hops in each route as opposed to the actual time it takes to traverse the route. Example in Figure 14, the route marked in red is of latency equal to 2 hops.

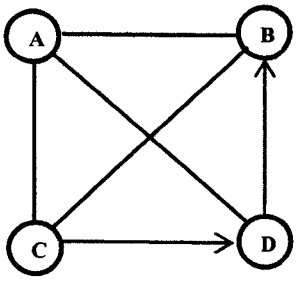


**Figure 14 Latency Example**

- **Success Rate**

The success rate of individual algorithms is expressed in terms of the ratio of the total number of ants that arrive at the destination node versus the total number of ants dispatched from the source node.

- **Path Discovery Time**

The path discovery time is a metric that is mainly ignored in literature but considered and highlighted in this thesis. It is also referred to as convergence time. It is important that it be known how long is needed in each algorithm before the source node is aware of a route to the destination node and is able to begin transmitting data. As more paths are discovered during the algorithm process, the source node can always adjust the route of its data path if a shorter route is discovered. The path discovery time is expressed in ticks or number of cycle units which is a NetLogo counter that represents execution time [NetLogo].

- **Energy Consumption**

The energy consumption is defined as the total amount of energy consumed in the network. In our simulation, to avoid complexity and for proof of concept, it is assumed that the amount of energy consumed in any transmission between any two nodes is equal. Therefore, the total amount of energy consumed is equivalent to the total number of ants sent through the grid network.

- **Energy Efficiency**

The energy efficiency is the ratio of the total number of ants that arrive at the destination versus the total amount of energy consumed in the grid network.

## 5.2 Data Collection

### 5.2.1 Baseline Experiment Results on the AntNet Based Algorithms

This section describes the testing methodology performed on AntNet, Improved AntNet and the Pharaoh algorithms and the results collected in a standard configuration.

In preparing the data for every metric, 10 samples were taken and the mean was calculated. In order to reduce the size of data sets collected and minimize the fluctuation of the data within it, measurements were made less frequently. For that reason, a measurement was made every 4 ticks. Time to path discovery is one measurement value made every simulation run.

To begin, the initial testing is done on a grid network of size $n = 7$ *and* $r = 3$ values. Figure 15, illustrates a snapshot of an ongoing simulation.

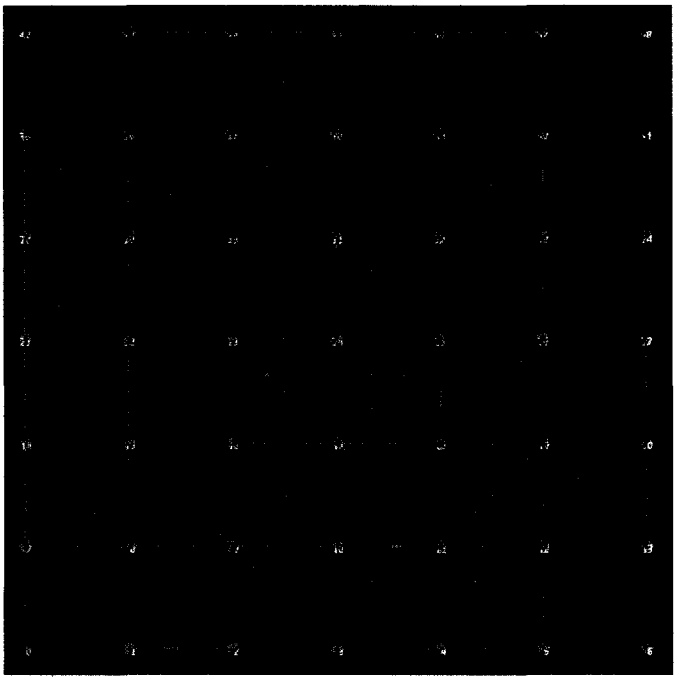

**Figure 15 Simulation snapshot in a grid network n = 7 / r =3**

Below are the charts and tables for each metric comparing each of the three algorithms together:

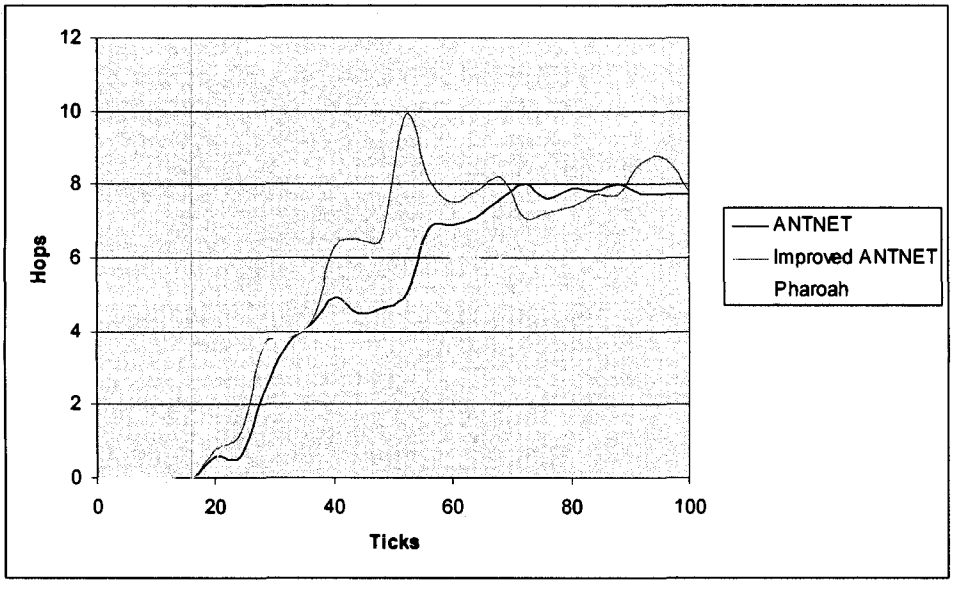


**Figure 16 Latency n = 7 / r = 3**

Final latency values were averaged out for each sample to establish a baseline to compare the three algorithms in situations where the source and destination nodes are constant over extended periods of time. The results can be found in table 1. The units of the results shown are number of hops:

| | |
|---|---|
| AntNet | 7.4 |
| Improved AntNet | 8.2 |
| Pharaoh | 6.1 |

**Table 1 Final Latency n = 7 / r = 3**

Standard deviations of the final latency for each algorithm are 3.17, 2.4, and 0.88 respectively. Pharaoh Routing's low Final Latency standard deviation shows that the readings are clustered closely around the mean which gives indication of a higher reliability and a sense of consistency of the algorithms performance in this metric. As for AntNet and Improved AntNet, the relatively higher standard deviation value is attributed to the initial random searching of forward ants in search of the destination node.

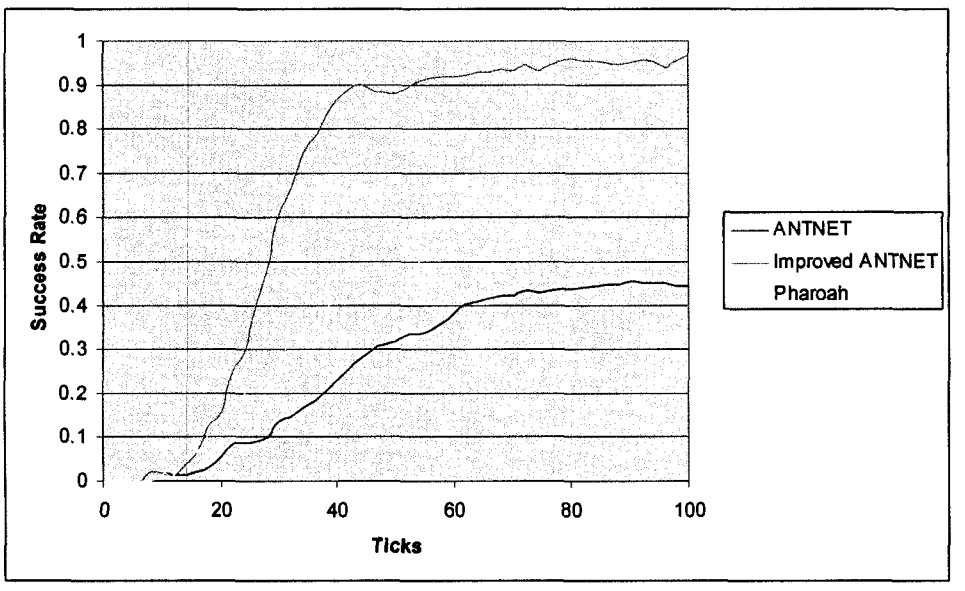


**Figure 17 Success Rate n = 7 / r = 3**

Improved AntNet has a higher success rate while maintaining comparable latency value as that of AntNet and Pharaoh Routing. The positive increase in the success rate of Improved AntNet in this test run can be tied to forward sensing capabilities introduced to

forward ants. The capability to sense ahead of moving forward whether the node was previously visited or not in forward ants improves performance in two ways:

- Such feature allows for far greater exploration in the grid network for the destination node removing the possibility that a forward ant will revisit a previously explored node.
- Removal of the possibility that loops may exist, reducing the number of ants that die significantly, thus improving success rates.
- The only case where ants may die is if they reach a node in the network from which they cannot move on to another node they have not already visited.

Pharaoh Routing achieves higher success rate than AntNet because of forward ants to mark paths that end up in loops with a negative pheromone preventing other forward ants from venturing into paths that lead away from the destination node and into unwanted loops.

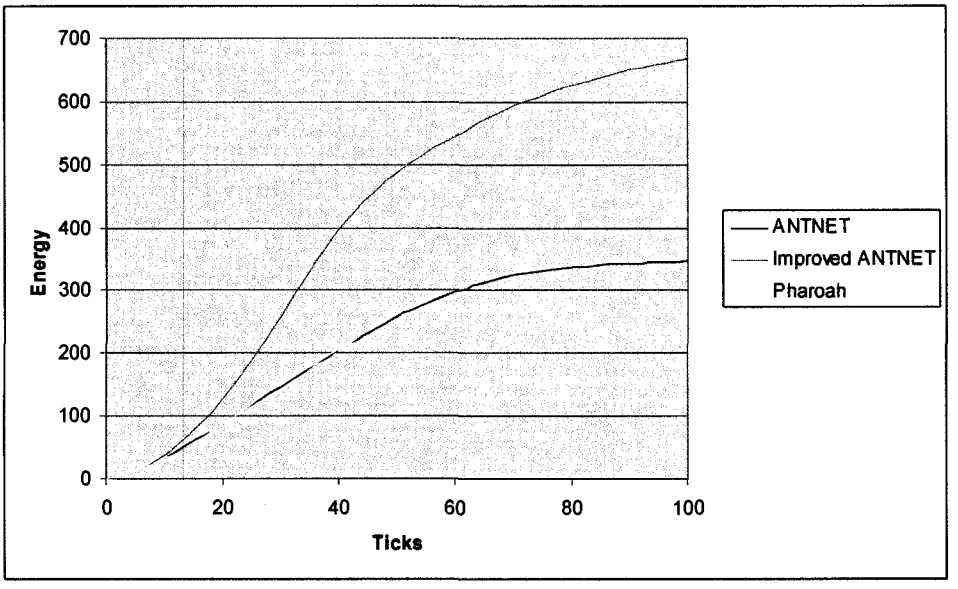


**Figure 18 Energy Consumption n = 7 / r = 3**

Even though Improved AntNet has a higher success rate, it consumes far greater energy than the others which may not be acceptable in networks where energy resources may be scarce.

Pharaoh Routing consumes the least amount of energy of all three algorithms with time. The reason this can be explained as follows: Less and less ants are spending time going through paths that lead away from the destination node and into unwanted loops so far less energy is being consumed.

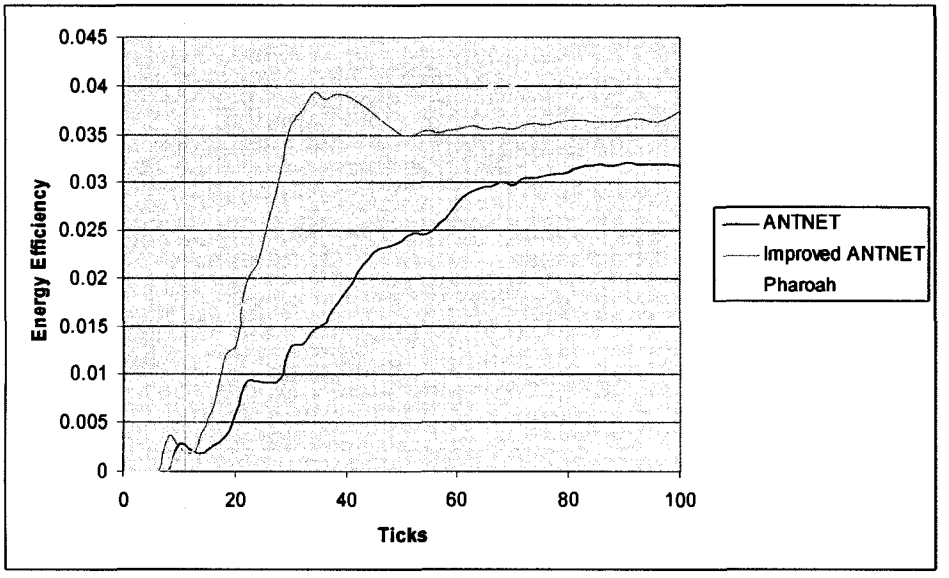


**Figure 19 Energy Efficiency n = 7 / r = 3**

Table 2 illustrates the time to path discovery values observed. The unit of the result shown is number of ticks:

| | |
|---|---|
| AntNet | 37.9 |
| Improved AntNet | 29.1 |
| Pharaoh | 31 |

**Table 2 Time to Path Discovery n = 7 / r = 3**

The standard deviations observed for the time to path discovery values for the three algorithms are 18.25, 6.37 and 9.97 respectively. These values indicate to what degree the individual readings of the data are dispersed around the mean. They can also be an indication of accuracy. AntNet and Pharaoh Routing have slightly more disperse time values than Improved AntNet. This may well be an indication of how accurate and deterministic Improved AntNet could be in this regard.

Looking at the measurements results, we can draw the following conclusions regarding

our improvements introduced in Improved AntNet and Pharaoh Routing algorithms:

1. Improved AntNet has the highest success rate
2. Improved AntNet finds the destination node quicker
3. Pharaoh Routing has the least latency
4. Pharaoh Routing consumes the least energy
5. Pharaoh Routing is the most energy efficient.

Of the three routing algorithms, it can be said that Pharaoh Routing is the better option in real life scenarios. Typical networks have limited energy resources, so the need for a routing algorithm with higher success rates and energy efficiency is great. With that being said, Pharaoh Routing still maintains comparable latency values and finds the destination node second fastest after Improved AntNet.

### 5.2.2 Varying Network Parameters Experiments

This section describes the testing done on AntNet, Improved AntNet and the Pharaoh Routing algorithms in different network configurations. Comparison on how the algorithms perform against the baseline chosen earlier is also described.

As in the previous section, every metric is made up of the mean of 10 samples. In order to reduce the size of data sets collected and minimize the fluctuation of the data within it, measurements were made less frequently. For that reason, a measurement was made every 4 ticks. Time to path discovery is one measurement value made every simulation run.

The testing in the first configuration is done on a grid network of size $n = 7$ *and* $r = 2$ values. We have reduced the connectivity of the network by reducing the value of variable $r$. This considerably reduces the number of links that exist amongst the 49 nodes from 452 links to 212. The Figure 20 illustrates a simulation snapshot.

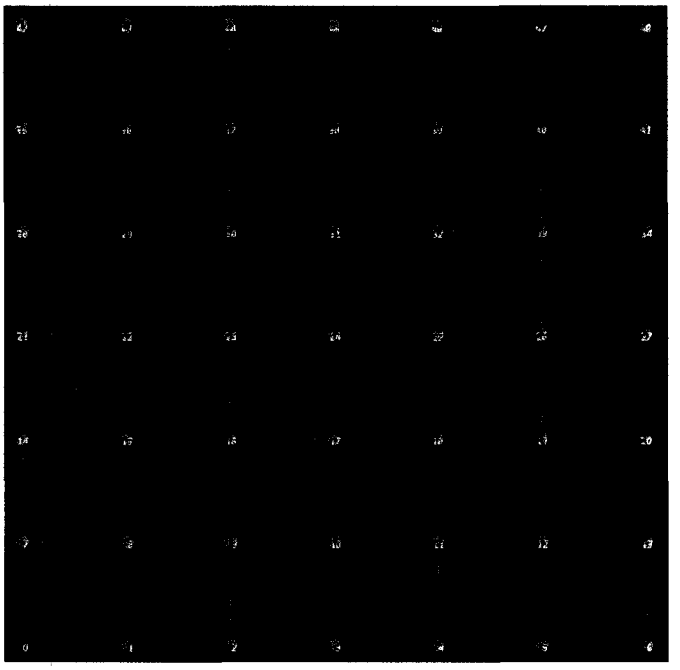

**Figure 20 Simulation snapshot in a grid network n = 7 / r = 2**

Below are the charts and tables for each metric comparing each of the three algorithms together:

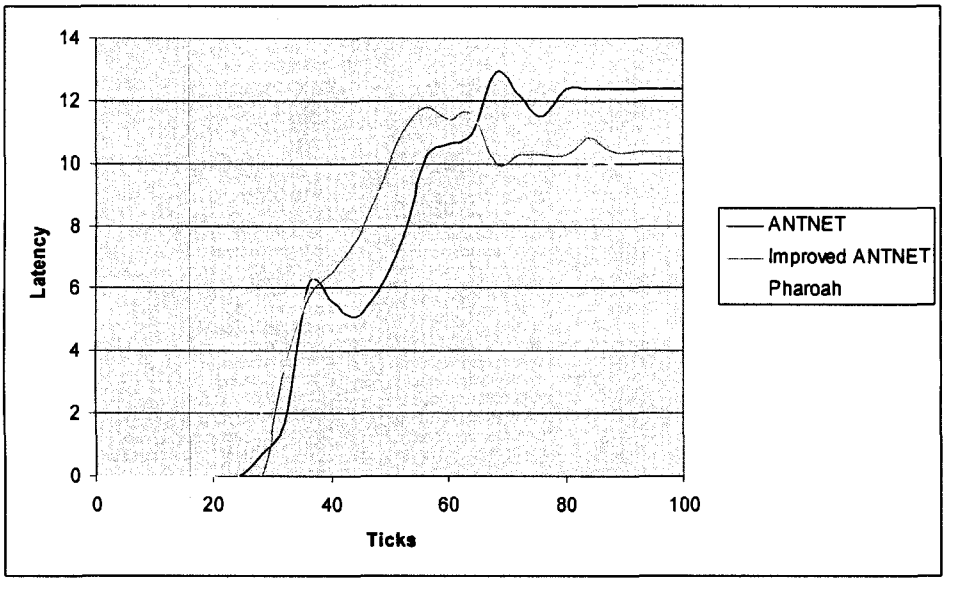


**Figure 21 Latency n = 7 / r = 2**

Table 3 illustrates the final latency values observed. The units of the results shown are number of hops:

| | |
|---|---|
| AntNet | 12 |
| Improved AntNet | 10.1 |
| Pharaoh | 9.4 |

**Table 3 Final Latency n = 7 / r = 2**

With decreased network connectivity, AntNet suffers the greatest in terms of latency. This considerable increase in latency is again due to the random nature of the AntNet algorithm. With decreased connectivity, forward ants initially have a hard time finding the destination node. Improved AntNet and Pharaoh Routing display comparable latency values across time with Pharaoh Routing having a slightly lower final latency value.

Standard deviation values of the final latency are 2.79, 2.51, and 1.07. This once again, shows that Pharaoh Routing is favored in this aspect.

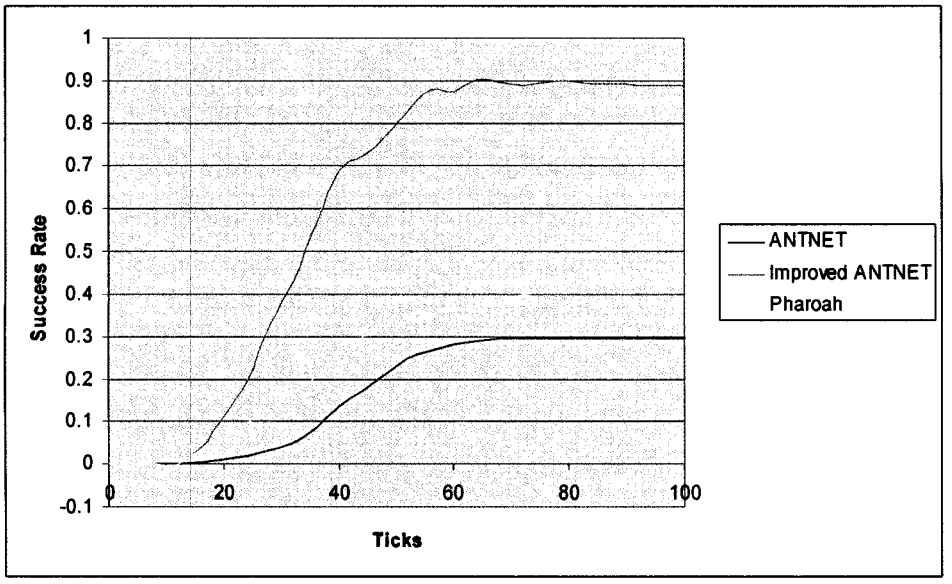


**Figure 22 Success Rate n = 7 / r = 2**

Again, Improved AntNet shows that it can achieve higher success rate than the remaining algorithms even with the decreased connectivity. This is attributed to the forward sensing capabilities of forward ants.

AntNet and Pharaoh Routing's success rates are reduced slightly in this scenario but not significantly with Pharaoh Routing maintaining a higher success rate throughout the graph.

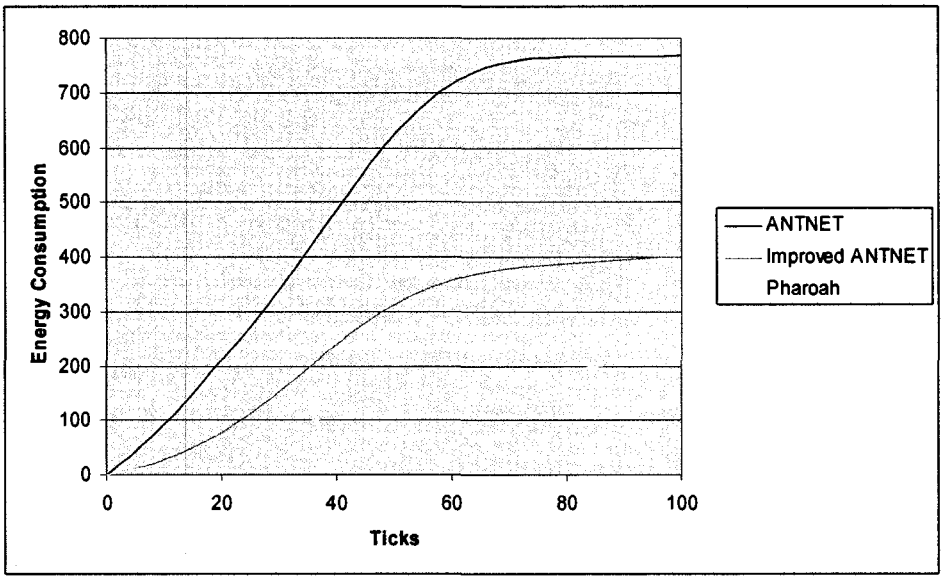


**Figure 23 Energy Consumption n = 7 / r = 2**

With the decreased connectivity, AntNet consumes much greater energy than the other algorithms. Due to the random walking of forward ants in search of the destination combined with the delayed convergence, AntNet consumes greater energy.

It should be noted that Improved AntNet consumes far less energy in poor connectivity conditions. It is noticed that its energy consumption reaches the range of 600-700 transmissions, while in this case its energy consumption barely reaches the 400 transmissions mark of the graph. In decreased connectivity conditions and with the forward sensing feature of forward ants, it is likely that the algorithm will find the destination node much quicker than AntNet, thus consuming less energy.

Pharaoh Routing still maintains lowest power consumption. This is to be expected due to the fact that forward ants will avoid paths that have been marked to lead to unrewarding paths.

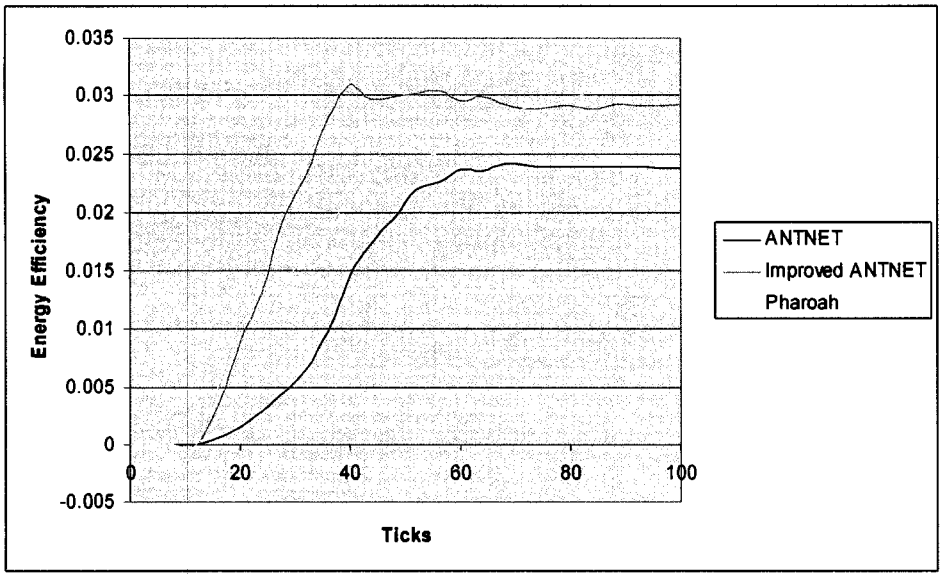


**Figure 24 Energy Efficiency n = 7 / r = 2**

As expected with AntNet consuming far greater energy than the remaining algorithms and with having a reduced success rate that it would have the lowest energy efficiency of the three algorithms.

Table 4 illustrates the time to path discovery values observed. The result is shown in tick units:

| | |
|---|---|
| AntNet | 39.3 |
| Improved AntNet | 37 |
| Pharaoh | 36.5 |

**Table 4 Time to Path Discovery n = 7 / r = 2**

The standard deviations observed for the time to path discovery values for the three algorithms are 9.07, 6.67 and 9.78 respectively. Pharaoh and Improved AntNet have very close time to path discovery values but Improved AntNet has the smallest standard deviation. In this case Improved AntNet has the least standard deviation indicating that it is the most consistent of the three and is a sign of its accuracy and predictability.

**Second configuration and simulation results**

The testing in the second configuration is done on a grid network of size $n = 5$ *and* $r = 3$ values. The number of nodes of the network is reduced from 49 to 25 but the network is kept highly connected with the number of links between the nodes being equal to 188. Figure 14 depicts a snapshot of the simulation during a run in this configuration.

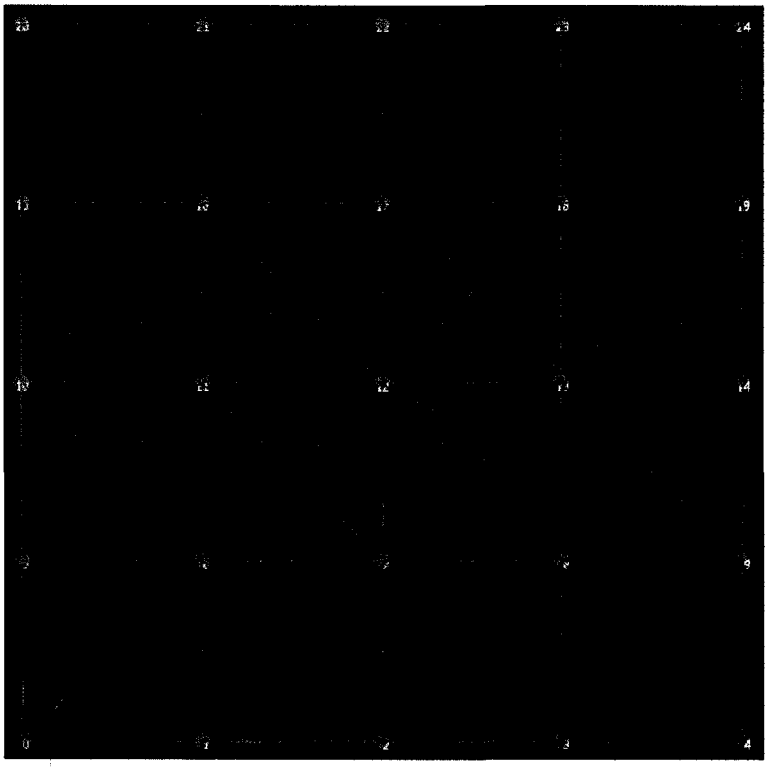

**Figure 25 Simulation snapshot in a grid network n = 5 / r = 3**

Below are the charts and tables for each metric comparing each of the three algorithms together:

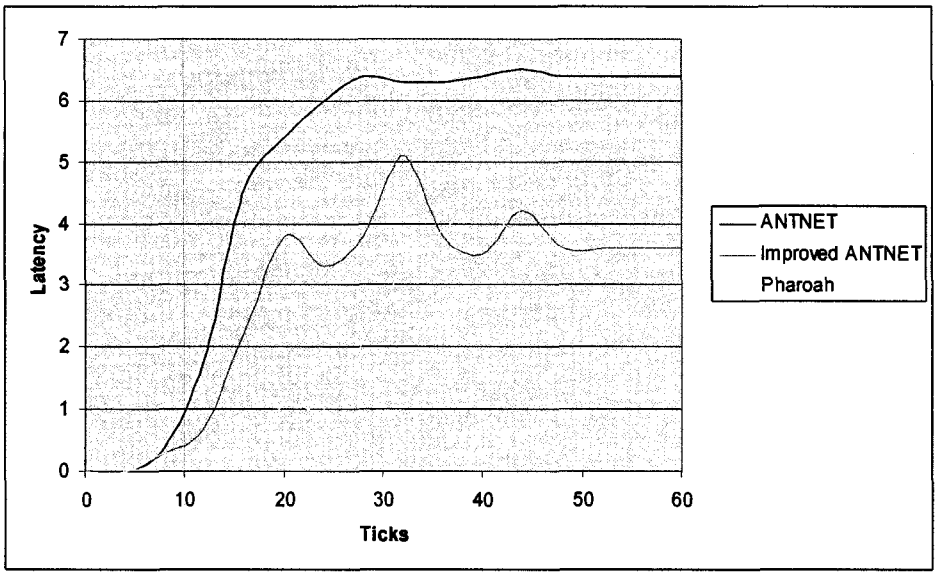


**Figure 26 Latency n = 5 / r = 3**

Table 5 illustrates final latency values observed. The units of the results shown are number of hops:

| | |
|---|---|
| AntNet | 6.4 |
| Improved AntNet | 3.6 |
| Pharaoh | 3.8 |

**Table 5 Final Latency n = 5 / r = 3**

In a smaller well connected network, Pharaoh Routing has a final latency that is comparable to that of Improved AntNet. It should be noted that Pharaoh Routing latency also has lower values for the first half of the simulation.

AntNet still has a higher latency values than the other algorithms due to the random searching of forward ants for the destination node.

Standard deviation values for the final latency are 2.37, 0.84, and 0.79. Lower standard deviations for final latency values of both Improved AntNet and Pharaoh Routing indicate that the values are closer to the mean and that the algorithms themselves are more reliable in terms of their latency.

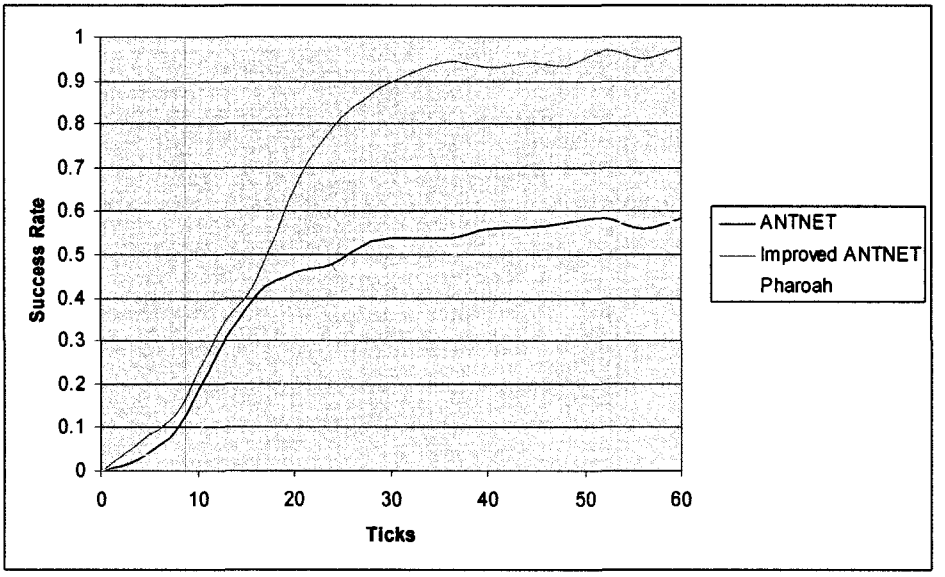


**Figure 27 Success Rate n = 5 / r = 3**

As expected, due to the forward sensing feature given to forward ants and constant avoidance of previously visited nodes, Improved AntNet has the highest success rates amongst all three algorithms.

AntNet displays a high success rate throughout the simulation than Pharaoh Routing. As the simulation progresses Pharaoh Routing success rate catches up with that of AntNet and eventually becomes very close in value. This increased performance of AntNet is due to the reduced number of nodes, higher connectivity and closer proximity of the destination to the source node.

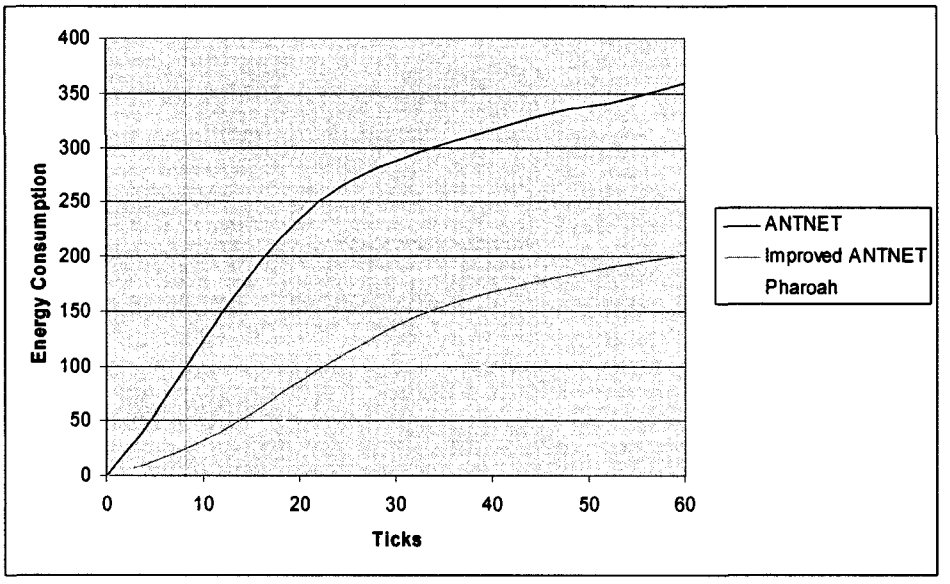


**Figure 28 Energy Consumption n = 5 / r = 3**

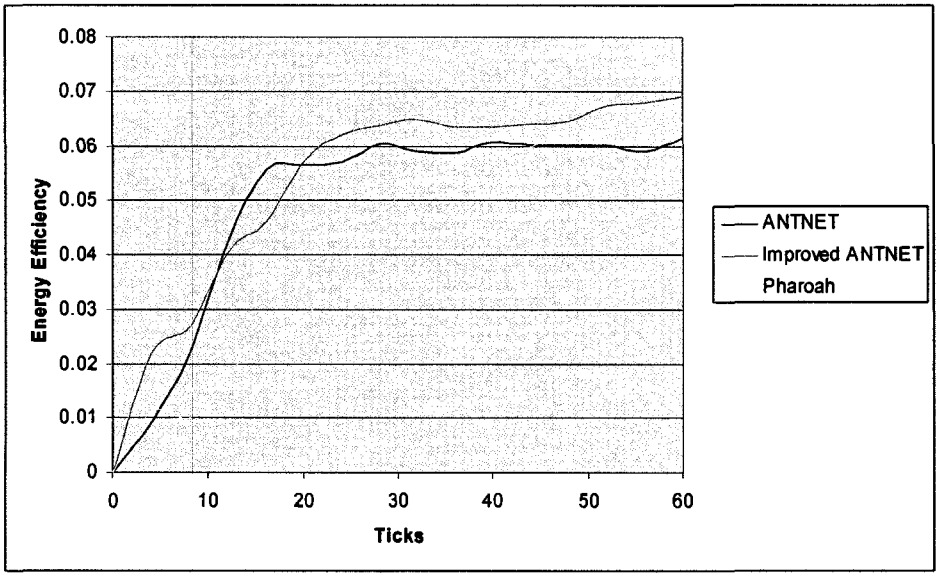


**Figure 29 Energy Efficiency n = 5 / r = 3**

With the increased latency and lowered success rate of AntNet in this configuration, it is expected that it consumes more energy than the other three algorithms. Pharaoh Routing as in previous configurations consumes the least amount of energy.

Unlike in previous configurations, all three algorithms display comparable energy

efficiency with time. Pharaoh Routing shows that it has lower energy efficiency early in the simulation due to the delayed finding of the destination node.

Table 6 shows the time to path discovery values observed. The units are in number of ticks:

| | |
|---|---|
| AntNet | 12 |
| Improved AntNet | 18.5 |
| Pharaoh | 25.2 |

**Table 6 Time to Path Discovery n = 7 / r = 2**

The standard deviations observed for the time to path discovery values for the three algorithms are 3.37, 7.85 and 8.22 respectively.

AntNet finds the destination node the quickest in this configuration. Factors such as close proximity of the source and destination nodes could be the leading cause in how low this value is. Unlike previously, Improved AntNet and Pharaoh Routing take longer to find the destination node.

**Third Configuration**

The testing in the third configuration is done on a grid network of size $n = 5$ *and* $r = 2$ values. Further to reducing the number of nodes from 49 from the previous section we now reduce the connectivity by reducing the value of r to 2. This reduces the total number of links considerably between nodes from 188 to 102. Figure 19 depicts what the simulation would typically look like during a run in this configuration.

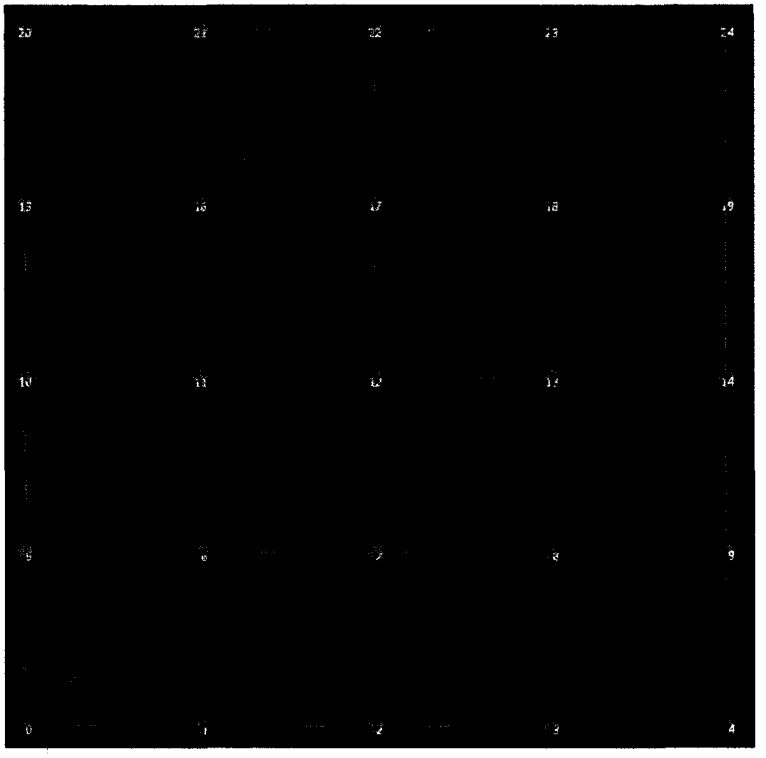

**Figure 30 Simulation snapshot in a grid network n = 5 / r = 2**

Below are all the charts and tables for each metric comparing each of the three algorithms together:

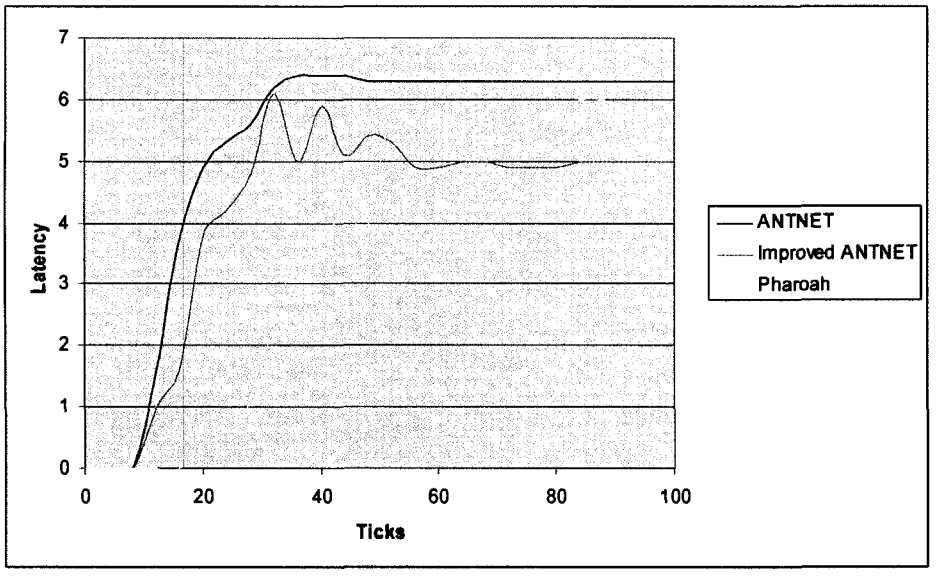


**Figure 31 Latency n = 5 / r = 2**

Table 7 illustrates final latency values observed. The units of the results shown are number of hops:

| | |
|---|---|
| AntNet | 6.3 |
| Improved AntNet | 5 |
| Pharaoh | 5.8 |

**Table 7 Final Latency n = 5 / r = 2**

The three algorithms show very close final latency values with AntNet having the highest latency of the three throughout the simulation. Pharaoh Routing has lower latency early in the simulation but its curve quickly catches up to Improved AntNet and surpasses it although not by much.

The standard deviation for the final latency of the three algorithms is 1.34, 0.94 and 1.23 respectively. Lower standard deviations show that the data points are clustered around the mean. With the comparable final latency values of each of the algorithms, it is an indication that similar results should be expected in such configuration.

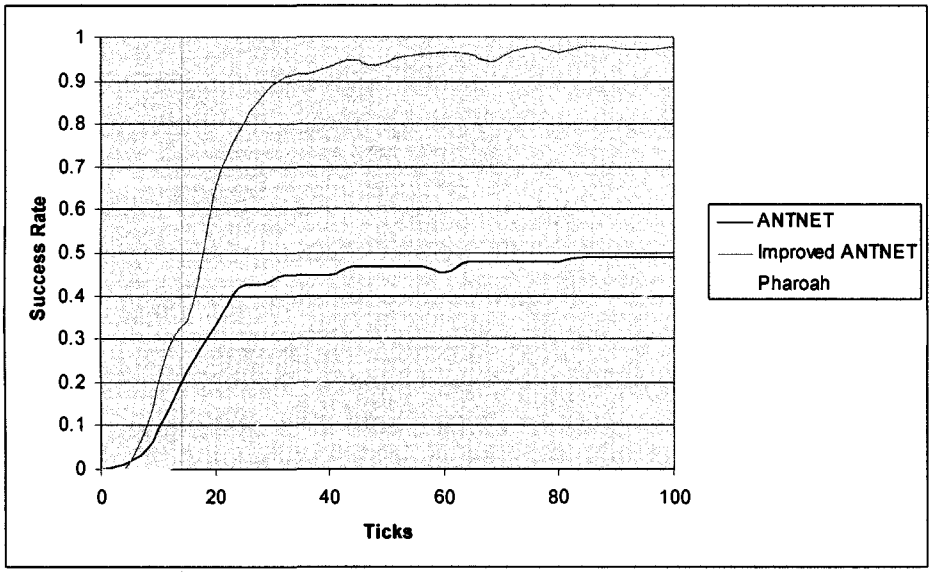


**Figure 32 Success Rate n = 5 / r = 2**

Improved AntNet as previously seen in other configurations has the highest success rate. AntNet also performs better than Pharaoh Routing. The decrease in the success rate of

Pharaoh Routing is due to the limited connectivity and the probably marking of possible paths to the destination as paths that lead to unwanted loops.

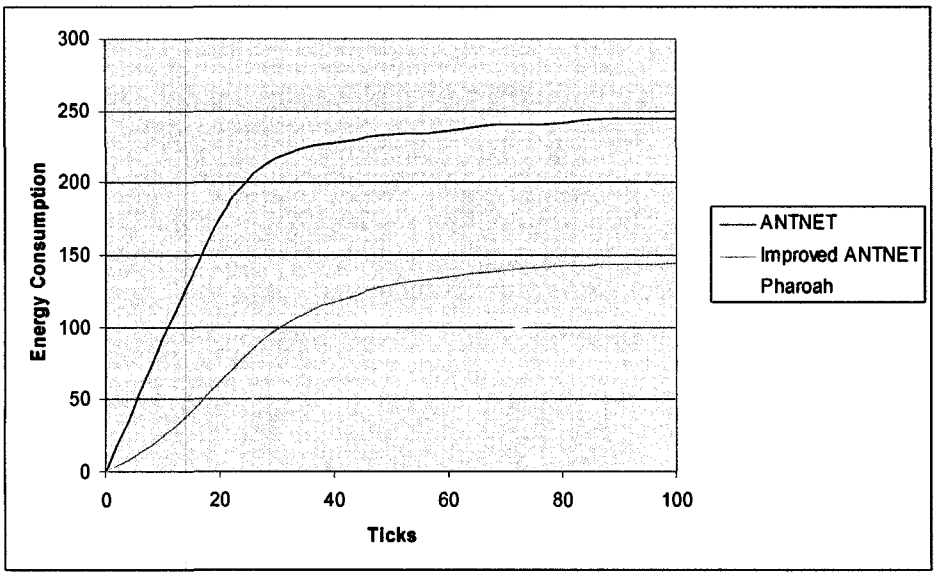


**Figure 33 Energy Consumption n = 5 / r = 2**

Even with the decreased success rate of Pharaoh Routing, it still consumes less energy. Improved AntNet follows with the next least energy consumption followed by AntNet that consumes the most.

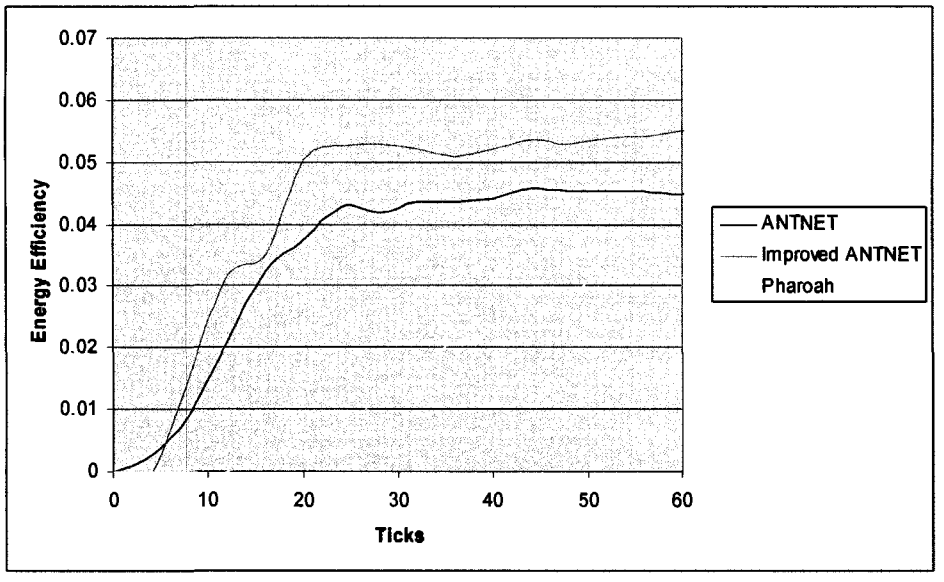


**Figure 34 Energy Efficiency n = 5 / r = 2**

Improved AntNet is the most energy efficient in this configuration while Pharaoh Routing is the least efficient.

Table 8 shows the time to path discovery values observed. The units are in number of ticks:

| | |
|---|---|
| AntNet | 16 |
| Improved AntNet | 19.7 |
| Pharaoh | 36.9 |

**Table 8 Time to Path Discovery n = 7 / r = 2**

The standard deviations observed for the time to path discovery values for the three algorithms are 6.62, 6.65 and 10.52 respectively.

AntNet finds the destination node the quickest in this configuration. Factors such as close proximity of the source and destination nodes and decreased number of connections that lead to the destination could be the leading cause in how low this value is. Improved AntNet has the second lowest time to path discovery value while Pharaoh Routing suffers the greatest with low connectivity.

Even with its low energy consumption, Pharaoh Routing has a low success rate, the lowest energy efficiency due to these factors:

- Increased delay in finding the destination node
- Early random wandering in a small network with limited connectivity may cause possible paths to the destination to be marked as part of undesired loops.

# CHAPTER 6

# HYRBID ANT/GENETIC ROUTING ALGORITHM COMPARISON

After presenting the simulation results achieved using AntNet, Improved AntNet and Pharaoh Routing in the previous chapter, the results of the proposed Hybrid AntNet/Genetic algorithm will be presented in this chapter and will be compared with the remaining algorithms. Again, we are using NetLogo to simulate the testing environment. It is assumed that all the networks simulated are absent of any other traffic data and that the source and destination nodes remain static.

## 6.1 Environment Design

This section describes the testing done on the proposed hybrid AntNet/Genetic algorithm and the results collected based on the standard baseline configuration outlined in the previous chapter and its comparison with the results of the other proposed algorithms.

In preparing the data for every metric, 10 samples were taken and the mean was calculated. In order to reduce the size of data sets collected, and minimize the fluctuation of the data per set, measurements were made less frequently. For that reason, a measurement was made every 4 ticks. Time to path discovery is one measurement value made every simulation run.

## 6.2 Experimental Data

Below are the tables and charts containing the data collected using the Hybrid AntNet/Genetic algorithm on the baseline configuration and the of performance comparison with AntNet, Improved AntNet and Pharaoh Routing.

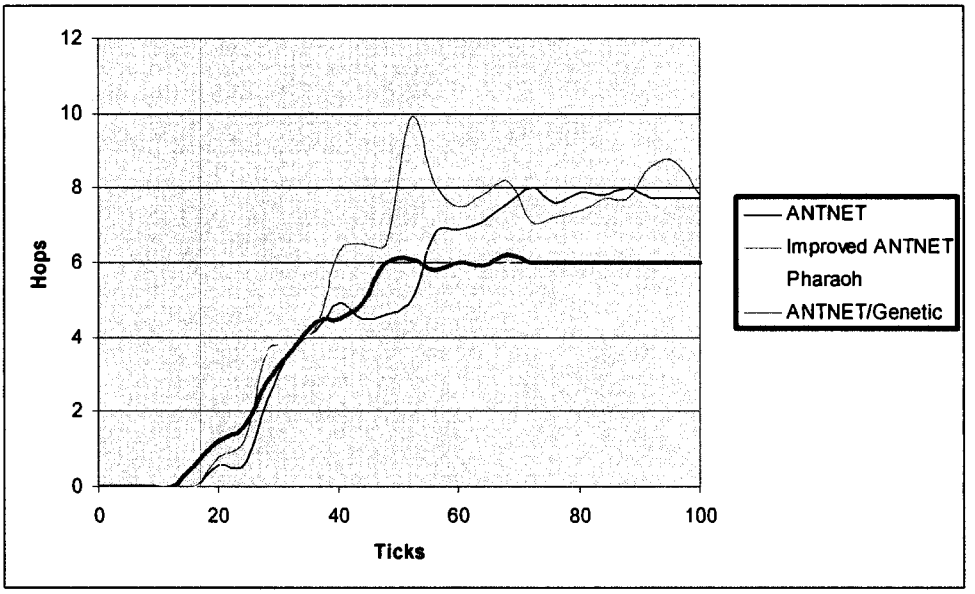


**Figure 35 Hybrid AntNet/Genetic Latency Comparison**

| | |
|---|---|
| AntNet | 7.4 |
| Improved AntNet | 8.2 |
| Pharaoh | 6.1 |
| **AntNet/Genetic** | **6** |

**Table 9 Hybrid AntNet/Genetic Final Latency Comparison**

The hybrid AntNet/Genetic algorithm has the least final latency value. Observing the curve on the latency graph, it follows the Pharaoh Routing curve quite closely but with slightly lesser latency values.

The standard deviations of the final latency for all four algorithms are 3.17, 2.4, and 0.88 and 1.05 respectively. Along with Pharaoh Routing, the hybrid AntNet/Genetic algorithm has a low standard deviation value indicating the samples collected are closely clustered around the mean. As mentioned before, this gives a sense of how reliable the algorithm is

in this aspect. It also provides a sense of predictability of the algorithms final latency values.

This improved latency is expected. The significant improvement in the hybrid AntNet/Genetic algorithm can be explained as follows: if a shorter path is found as a result of the genetic crossover operations on a subset of paths found by the AntNet algorithm, that path is enforced *n* times with a trail of pheromone. This ensures that future forward ants traverse that path moving forward maintaining lower latency values. This also prevents forward ants that concurrently discover other paths that may or may not be longer than the one discovered due to genetic crossover operation from taking precedence.

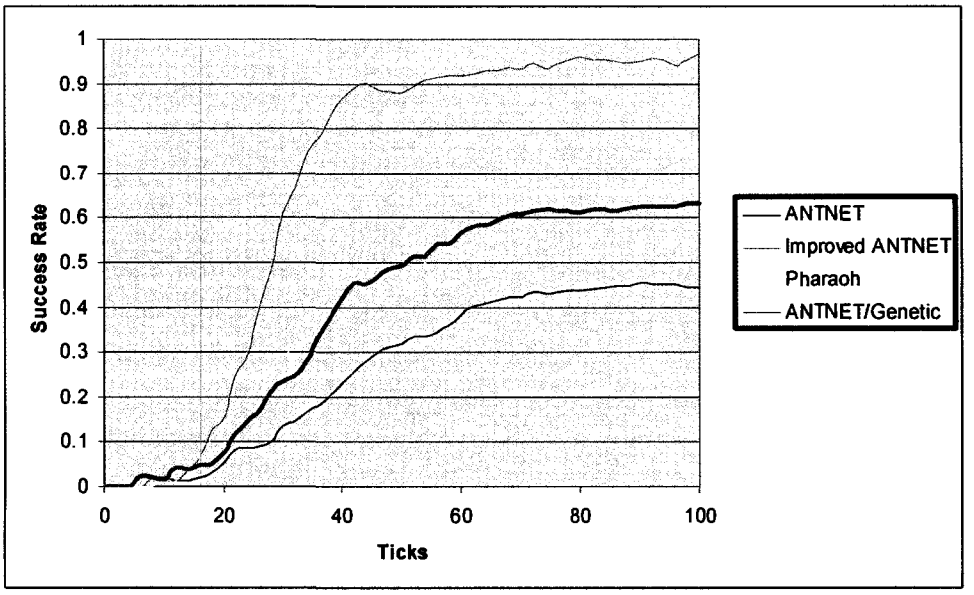


**Figure 36 Hybrid AntNet/Genetic Sucees Rate Comparison**

The hybrid AntNet/Genetic algorithm has a success rate that is higher than AntNet and Pharaoh Routing but not as good as Improved AntNet. As we notice on the graph in the figure above, hybrid AntNet/Genetic algorithm follows closely the Pharaoh Routing curve and outperforming AntNet early on the simulation. Close to the midway point, it surpasses the Pharaoh Routing success rate curve and stabilizes significantly above it.

The reasoning behind the fact that the success rate of hybrid AntNet/Genetic algorithm has a lower curve through the first half and then increases significantly at the midway point can be explained as follows: Early on in the simulation, the algorithm behaves exactly the same as AntNet till a shorter paths are found due to genetic crossover operations and subsequently, these paths are enforced. Following that point, more forward ants traverse a shorter path towards the destination node, increasing the success rate of the algorithm.

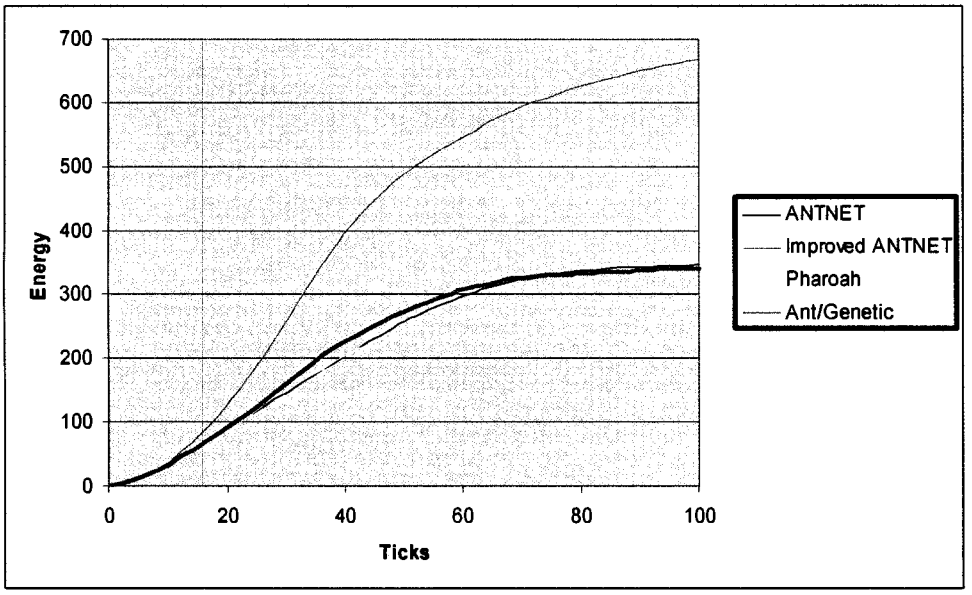


**Figure 37 Hybrid AntNet/Genetic Energy Consumption Comparison**

The hybrid AntNet/Genetic algorithm consumes amount of energy comparable to AntNet but less than Improved AntNet and more than Pharaoh Routing. It is expected that it consumes similar amounts to AntNet as the algorithm behaves primarily as AntNet does. Only when shorter paths are discovered due to genetic crossover operations do AntNet operations halt and the destination node enforces the newly discovered path.

This measurement neglects the amount of energy consumed at the source destination in performing the genetic crossover operations on the paths found. It only takes into account energy consumed in transmissions throughout the grid network. It should be noted that on the other hand, this measurement does take into account the transmissions used to enforce paths discovered due to genetic crossover.

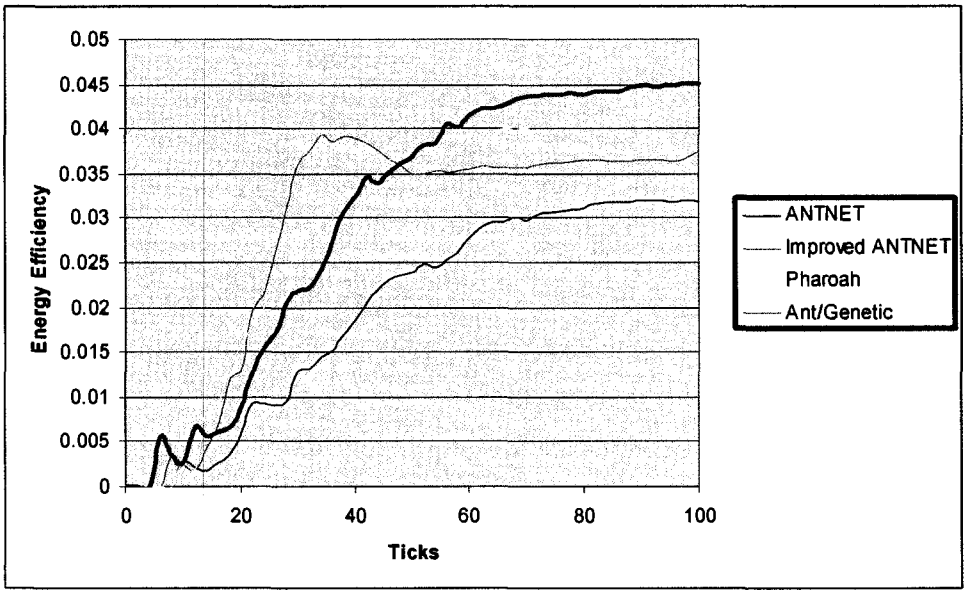


**Figure 38 Hybrid AntNet/Genetic Energy Efficiency Comparison**

The hybrid AntNet/Genetic algorithm can be considered to be the most efficient of all four algorithms. Its energy efficiency is slightly lower than Improved AntNet and Pharaoh Routing early in the simulation but surpasses both curves with time.

The increased energy efficiency is expected with the increased success rate while maintaining similar energy consumption performance to AntNet. This is unlike in Improved AntNet, where energy consumption is increased significantly to achieve a higher success rate while maintaining lower energy efficiency.

| | |
|---|---|
| AntNet | 37.9 |
| Improved AntNet | 29.1 |
| Pharaoh | 31 |
| **AntNet/Genetic** | **32.1** |

**Table 10 Time to Path Discovery Comparison**

The hybrid AntNet/Genetic algorithm does not possess the lowest time to path discovery value but still has still shown significant performance improvements. The standard deviations of each algorithm are as follows: 18.25, 6.37, 9.97 and 11.01. The high standard deviation value for the hybrid AntNet/Genetic algorithm is an indication that this algorithm again behaves primarily as AntNet.

# CHAPTER 7

# CONCLUSION

## 7.1 Conclusion

This thesis has presented three AntNet based algorithms. It has been established by reviewing the literature and via experimentation done in this thesis that AntNet does not perform well overall. Each algorithm has introduced improvement to the AntNet algorithm in one way or another. Based on the performance evaluation of each algorithm in Chapters 6 and 7, the initial assessment is as follows:

In Improved AntNet, by giving the forward ant forward sensing capabilities the following has improved:

- Increased success rate – more forward ants are reaching the destination
- Increased convergence time – forward ants are finding the destination quicker

The tradeoffs for these improvements is increased energy consumption and reduced energy efficiency

In Pharaoh Routing, by imitating the behavior of an that lives in natural habitat, one that lays negative trail pheromone to repel other ants from unrewarding paths along with positive pheromone that leads to the destination, an algorithm has been created with the

following characteristics:

- Lowered latency – least number of hops to the destination
- Least energy consumption – least number of packet transmissions overall
- And most energy efficiency – least packets arrived to packets transmitted ratio

Studying the effect of varying network parameters such as the number of nodes and the degree of connectivity on all the algorithms and comparing them with AntNet resulted in the following findings:

Large network/poor connectivity:

- Improved AntNet has the highest success rate
- Pharaoh Routing has the least latency, consumes lesser energy and has comparable energy efficiency to Improved AntNet
- Both algorithms converge on to the destination with comparable times but quicker than AntNet

Small network/well connected:

- Improved AntNet and Pharaoh routing have comparable latencies and energy efficiency
- Improved AntNet has the highest success rate
- Pharaoh Routing consumes the least energy
- AntNet converges quicker than Improved AntNet and Pharaoh Routing

Small network/poor connectivity:

- Improved AntNet and Pharaoh Routing have the similar latency to AntNet
- Improved AntNet has the highest success rate and highest energy efficiency
- Pharaoh Routing has the least success rate and poor energy efficiency but consumes the least energy
- AntNet converges quicker than both algorithms with Pharaoh Routing having the worst convergence time.

Following the comparison of Improved AntNet and Pharaoh Routing with AntNet in different configurations, the Hybrid AntNet/Genetic algorithm is introduced. From, the testing done on this algorithm and comparison with the other three algorithms in chapter 6, the following was established regarding the algorithm:

- Has the least latency with very little difference with Pharaoh Routing
- Higher success rate than AntNet and Pharaoh Routing but less than Improved AntNet
- Consumes almost the same amount of energy as AntNet
- The most energy efficient of all algorithms
- Faster convergence speeds than AntNet

Furthermore, table 11 compares the features of the algorithms presented in this thesis with those introduced in [Zhang04] and [Chang03]:

| | |
|---|---|
| Improved ANTNET | Use of forward sensing to avoid previously traversed nodes to improve success rate |
| Pharaoh Routing | Introducing negative pheromones to be used along side positive ones to improve efficiency |
| Sensor Driven Cost-Aware Routing [Zhang04] | Using metrics such as distance of current node from destination |
| Flooded Forward Ant Routing [Zhang04] | Broadcast forward ants towards destination and stop when a good enough path is found |
| Flooded Piggybacked Ant Routing [Zhang04] | Combining forward ants with data ants to improve the success rate in dynamic networks |
| Hybrid AntNet/Genetic algorithm | Hybrid AntNet with single point crossover operations on discovered paths and reinforcement of paths according to their goodness |
| Genetic Ant Routing algorithm [Chang03] | A list of paths discovered by Ants is maintained and evolved using genetic operations and several copies of favorable paths are kept to be evolved further. |

Based on the experience with the implemented algorithms, the following conclusions can be drawn:

- The proposed Improved AntNet always maintains the highest success rate but at a cost of consuming the most energy and therefore it is not ideal for resource limited networks
- The proposed Pharaoh Routing seems ideal for most situations as it always consumes the least energy while maintaining high energy efficiency except in smaller networks where it suffers from poor success rate and slow convergence
- AntNet is only ideal to use in smaller poorly connected networks for the reason that is converged faster than Improved AntNet and Pharaoh while maintaining similar latency values
- Hybrid AntNet/Genetic algorithm is viable alternate that provides improvements in success rate, latency and convergence speed over AntNet while being the most energy efficient of all algorithms presented in this thesis.

## 7.2 Future Work

The future work can be summarized as follows:

- Comparing Hybrid AntNet/Genetic algorithm with the remaining algorithms in various network configurations
- The addition of traffic simulation and studying its effect on the performance of each algorithm
- Focusing on the route maintenance aspect of routing in telecommunication networks
- Further compare the performance of the proposed algorithms in this thesis with the ones presented in [Zhang04]
- Comparing the performance of hybrid Genetic/AntNet algorithm with Genetic Ant Routing Algorithm (GARA) introduced in [Chang03]

# References


[Baran00] B. Baran and R. Sosa, "A New Approach for AntNet Routing", *Computer Communications and Networks*, pages 303-208, 2000

[Beasley93] D. Beasley et al., "An Overview of Genetic Algorithms: Part 1, Fundamentals", *Inter-University Committee on Computing,* No. 15(2), pages 58-69, 1993

[Blum03] C. Blum & A. Roli, "Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison", *ACM Computing Surveys*, Vol. 35, No. 3, , pp. 268–308, September 2003.

[Blum05] C. Blum, "Ant colony optimization: Introduction and recent trends", Elsevier Physics of Life Review, No. 2, pages 253-273, 2005

[Bonabeau01] E. Bonabeau & C. Meyer, "Swarm Intelligence: A Whole New Way to Think About Business", *Harvard Business Review*, May 2001

[Bull93] D. Bull et al., "An Overview of Genetic Algorithms: Part 1, Fundamentals", *Inter-University Committee on Computing,* No. 15(4), pages 170-181, 1993

[Cadenas07] J.M. Cadenas et al., A Cooperative System of Metaheuristics, *Seventh International Conference on Hybrid Intelligent Systems*, 2007

[Cheng03] X. Cheng and Y. Hou, "A Study of Genetic Ant Routing Algorithm", *Proceedings of the Second International Conference on Machine Learning and Cybernetics*, November 2003

[Crossover] Crossover – Genetic Algorithms, http://en.wikipedia.org/wiki/Crossover_%28genetic_algorithm%29

[Dhillon07] S.S. Dhillon and P. Van Mieghem, "Performance Analysis of the AntNet algorithm", *Elsevier Computer Networks*, No. 51, pages 2104-2125, 2007

[DiCaro] G. Di Caro and M. Dorigo, "AntNet: A Mobile Agents Approach to Adaptive Routing", *Iridia Technical Report 97-12*

[DiCaro98] G. Di Caro and M. Dorigo, "AntNet: Distributed Stigmergetic Control for Communications Networks", *Journal of Artificial Intelligence Research*, No. 9, pages 317-365, 1998

[Dorigo97] M. Dorigo & L. M. Gambardella, "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem", *IEEE Transactions on Evolutionary Computation*, Vol.1, No.1, 1997

[Dorigo05] M. Dorigo & C Blum, "Ant Colony Optimization theory: A Survey", *Elsevier Theoretical Computer Science*, No 344, pages 243 – 278, 2005

[Dorigo06] Marco Dorigo, Mauro Birattari, and Thomas Stutzle, "Ant Colony Optimization: Artificial Ants as a Computational Intelligence Technique", *IRIDIA – Technical Report Series,* September 2006

[Dorigo07] M. Dorigo & K. Socha, "An Introduction to Ant Colony Optimization", *Approximation Algorithms and Metaheuristics*, CRC Press, 2007

[Ghoshray95] S. Ghoshray and K. K. Yen, "More Efficient Genetic Algorithm for Solving Optimization Problems", *IEEE International Conference on Systems, Man and Cybernetics*, Vol. 5, page(s): 4515-4520, 1995

[He00] L. He and N. Mort, "Hybrid Genetic Algorithms for Telecommunications Network Back-up Routing", *BT Technology Journal,* Vol. 18, Issue 4, pages 42-50, 2000

[Hu04] Y. Hu and S.X. Yang, "A Knowledge Based Genetic A1gorith:m for Path Planning of a Mobile Robot", *Proceedings of the 2004 IEEE international Conference on Robotics & Automation*, April 2004

[Johnson95] D.B. Johnson, "Routing in Ad Hoc Networks of Mobiles Hosts", Mobile Computing Systems and Applications, page 158-163, 1995

[Merkle02] D. Merkle et al., "Ant Colony Optimization for Resource-Constrained Project Scheduling", *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 4, August 2002

[Milano04] M. Milano and A. Roli, "MAGMA: A Multiagent Architecture for Metaheuristics", *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 33, No. 2, April 2004

[Mitchell96] M. Mitchell, "An introduction to Genetic Algorithms", *MIT Press*, Published 1996

[NetLogo] NetLogo User Manual, http://ccl.northwestern.edu/netlogo/docs/

[Osman96] I.H. Osman & G. Laporte, "Metaheuristics: A bibliography, *Annals of Operations Research 63*, pages 513 – 623, 1996

[Ribeiro07] C. Ribeiro et al., "Metaheuristics for optimization problems in computer communications", *Computer Communications*, Vol 3, No. 4, Pages: 656-669, 2007

[Robinson05] E. J.H. Robinson et al., "Insect Communication: "No Entry" Signal in Ant Foraging", *Nature*, Vol. 438, page 442, November 2005

[Robinson07] E. J.H. Robinson et al., "No Entry Signal in Ant Foraging (Hymenotera: Formicidae): New Insights from an Agent-Based Model", *Myrmecological News*, No. 10, September 2007

[Sosa00] B. Baran and R. Sosa, "AntNet: Routing Algorithm for Data Networks based on Mobile Agents", *Argentine Symposium on Artificial Intelligence*, 2000

[Strobbe05] M. Strobbe et al., "Implementation and evaluation of AntNet, a distributed shortest-path algorithm", *Advanced Industrial Conference on Telecommunications*, 2005

[Sun04] J. Sun, S. Xiong & F. Guo, "A New Pheromone Updating Strategy in Ant Colony Optimization", *Third International Conference on Machine Learning and Cybernetics*, 2004

[Tarasewich02] P. Tarasewich & P. R. McMullen, "Swarm Intelligence: Power in Numbers", *Communications of the ACM,* Vol. 45, No. 8, August 2002

[Tekiner] F. Tekiner et al., "The Antnet Routing Algorithm – A Modified Version", *Argentine Symposium on Artificial Intelligence*

[White98] T. White et al, "ASGA: Improving the Ant System by Integration with Genetic Algorithms", *Proceedings of the Third Annual Conference on Genetic Programming*, 1998

[Yun04] H. Yun and A. N. Zincir-Heywood, "Intelligent Ants for Adaptive Network Routing", *Second Annual Conference on Communication Networks and Services Research*, 2004

[Zhang04] Y. Zhang et al., "Improvements on Ant Routing for Sensor Networks", *ANTS 2004*, pp. 154–165, 2004