Routing with Swarm Intelligence

Tony White

SCE Technical Report SCE-97-15

Abstract

This technical report describes how biologically inspired agents can be used to solve control and management problems in Telecommunications. These agents, inspired by the foraging behavior of ants, exhibit the desirable characteristics of simplicity of action and interaction. The collection of agents, or swarm system, deals only with local knowledge and exhibits a form of distributed control with agent communication effected through the environment. In the technical report we explore the application of ant-like agents to the problem of routing in circuit switched telecommunication networks.

1. Introduction

Routing in telecommunication networks has mainly been static to date. With the advent of new technologies and, in particular, high capacity networks, much more flexibility will be required. Distance learning, video on demand, world wide information services such as WWW servers and many other services will create dynamic network load which should be taken care of by new algorithms. This is a totally new area of research with many economic and technical implications. The purpose of this report is to explore what swarm intelligence might offer to solve these dynamic and fundamentally distributed optimization problems.

Swarm intelligence research originates from the work on the emergence of collective behaviours of real ants. Ants have a small amount of cognitive capability, limited individual capabilities, and react instinctively in a probabilistic way to their perception of their immediate environment. They can, for example, find the shortest path between the nest and a food source by laying down on their way back from the food source a trail of an attracting substance, called pheromone. Ants wandering randomly around the nest can then be attracted by this trail and this will rapidly lead them to the food source. By laying down a pheromone trail of different density depending on the quality of the food source the have found, the colony becomes able to discriminate between food sources of different kinds and qualities.

The idea of using swarm intelligence as a new computational paradigm for solving engineering problems is quite recent. Marco Dorigo and colleagues were the first ones to propose adapting these ideas to the Travelling Salesman Problem [Dorigo et al. 92]. Many other applications since then have been proposed: graph partitioning [Kuntz et al. 94], data clustering [Faieta et al. 94], job shop scheduling [Colorni et al. 94], robotics [Beni 91], etc. See [Bonabeau et al. 94] for an overview.

The advantages of swarm intelligence are twofold. Firstly, it offers intrinsically distributed algorithms that can use parallel computation quite easily. Secondly, these algorithms show a high level of robustness to change by allowing the solution to dynamically adapt itself to global changes by letting the agents (the ants) self-adapt to the associated local changes.

2. Swarm Intelligence: Background

This section brings together information regarding the main sources of research in Swarm Intelligence. The most complete set of work has been done by Marco Dorigo, and this has influenced our work considerably. In addition, work undertaken at the Santa Fe Institute and Télécom Bretagne is briefly presented.

The Ant System

The Ant System is a general-purpose heuristic algorithm, which can be used to solve diverse combinatorial optimization problems. This work has been lead by Marco Dorigo at the Politecnico di Milano, Italy. For an in-depth technical description of the algorithm, the reader is referred to [Dorigo et al. 92].

Motivations for the Ant System

The Ant System (AS) has the following desirable characteristics:

1. It is versatile, in that it can be applied to similar versions of the same problem. For example, there is a straightforward extension from the travelling salesman problem (TSP) to the asymmetric travelling salesman problem (ATSP).

2. It is robust and general purpose. With minimal modifications, it can be applied to other combinatorial optimization problems such as the job shop scheduling problem (JSP) and the quadratic assignment problem (QAP).

3. It is a population-based heuristic. As such, it allows the exploitation of positive feedback as a search mechanism, as described in a later section. Consequently, it makes the system amenable to parallel implementations.

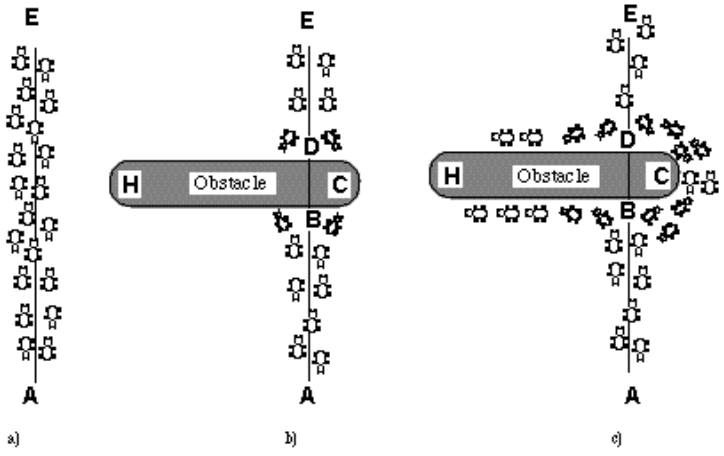
These desirable properties are mitigated by the fact that, for some applications, the Ant System can be outperformed by more specialized, domain specific algorithms. Many heuristics share this problem, examples being the much-researched problem solving techniques of simulated annealing (SA), and tabu search (TS). Nevertheless, as is the case with SA and TS, the AS represents a heuristic that can be applied to problems which are similar to a classical problem, such as TSP, but are sufficiently different as to make the application of a domain specific algorithm impossible. The ATSP is an example of such a problem.

The AS is an example of a distributed search technique. Search activities are distributed over ant-like agents, i.e. entities with very simple basic capabilities. These agents, in a metaphorical and highly stylized way, mimic the behaviour of real ants. In fact, research on the behaviour of real ants has greatly inspired the AS. The research inspiration for the AS arises from the work of ethologists, who attempted to understand how almost blind animals like ants could manage to establish shortest route paths from their colony to feeding sources and back.

Research found that the mechanism used for the communication of information among individuals regarding paths, and used to make routing decisions, consists of the sensing of pheromone trails. A moving ant lays some varying quantities pheromone on the ground, thus marking the path by a continuous trail of the substance. While an isolated ant apparently moves at random, an ant encountering a trail laid by an ant of its own species can sense it and decide to follow it with high probability. In this way, the trail is further reinforced with more pheromone. The collective behaviour of many ants attempting to find a path that emerges is in the form of an autocatalytic process where, the more the ants follow a trail, the more attractive that trail becomes for being followed. The process forms a positive feedback loop, where the probability with which an ant chooses a path will increase with the number of ants that previously chose the same path.

The following paragraphs are a modified explanation of path-emergence as provided by [Dorigo et al. 96].

> "Consider, for example, the scenario shown in Figure 1. Ants are walking along a path, for example from food source A to the nest E, and vice versa, see Figure 1a. Suddenly an obstacle appears and the path is cut off. Therefore, at position B the ants walking from A to E (or at position D those walking in the opposite direction) have to decide whether to turn right or left (Figure 1b). The choice is influenced by the intensity of the pheromone trails left by preceding ants. A higher level of pheromone on the right path gives an ant a stronger stimulus and thus a higher probability to turn right. The first ant that reaches point B (or D) has no preference for left or right branches of the path because there is no pheromone on either path. Because path BCD is shorter than BHD, the first ant following it will reach D before the first ant following path BHD (Figure 1c).



How ants find shortest paths
a) Ants follow a path between points A and E.
b) An obstacle is interposed; ants can choose to go around it following one of the two different paths with equal probability.
c) On the shorter path more pheromone is laid down.

Figure 1: Shortest Path Emergence

The result is that an ant returning from E to D will find a stronger trail on path DCB, caused by the half of all the ants, by chance, deciding to approach the obstacle via DCBA and by the already arrived ones coming via BCD. Therefore, they will prefer path DCB to path DHB. Consequently, the number of ants following path BCD per unit of time will be greater than the number of ants following BHD. Hence, the quantity of pheromone on the shorter path will grow more quickly than on the longer one. Therefore, the probability with which any single ant chooses a particular path is quickly biased towards the shorter one. The result is that, very quickly, all ants will choose the shorter path.

The algorithms that we are going to define in the following sections are models derived from the study of artificial ant colonies. Therefore, we call the system the Ant System (AS) and the algorithms we introduce, ant algorithms. As we are not interested in simulation of ant colonies, but in the use of artificial ant colonies as an optimization tool, our system will have some major differences with a real (natural) one:

- Artificial ants have some memory.
- They are not completely blind.
- They live in an environment where time is discrete.

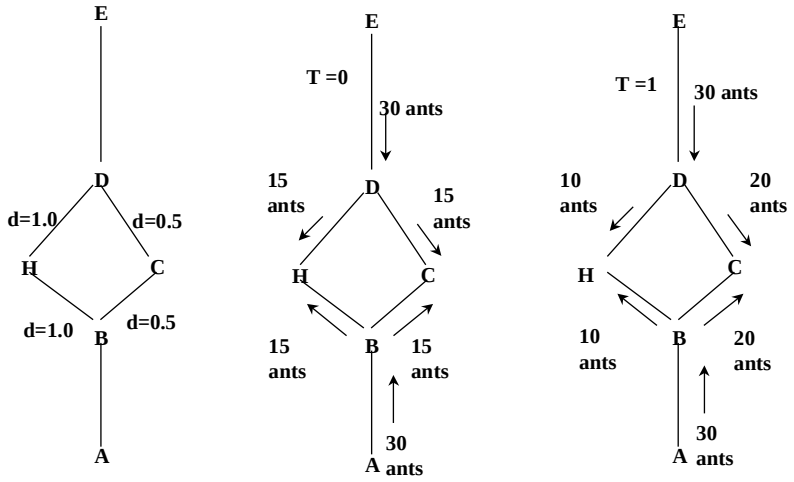


Figure 2: Pheromone trails

Nevertheless, we believe that the ant colony metaphor can be useful to explain our model. Consider the graph of Figure 2a, which is a possible AS interpretation of the situation of Figure 1b. In order to fix the AS ideas more concrete, suppose that the distances between D and H, between B and H, and between B and D—via C—are equal to unity. Let C be

positioned half the way between D and B (see Figure 2a). Now let us consider what happens at regular discrete points in time: t= 0, 1, 2, … and so on. Suppose that 30 new ants come to B from A, and 30 to D from E at each time unit, that each ant walks at a speed of one per time unit. Further, suppose that, while walking, an ant lays down at time t a pheromone trail of intensity one, which, to make the example simpler, evaporates completely and instantaneously in the middle of the successive time interval (t+1, t+2).

At t=0 there is no trail yet, but 30 ants are in B and 30 in D. Their choice about which way to go is completely random. Therefore, on average, 15 ants from each node will go towards H and 15 towards C (Figure 2b).

At t=1 the 30 new ants that come to B from A find a trail of intensity 15 on the path that leads to H, laid by the 15 ants that went that way from B and a trail of intensity 30 on the path to C. This trail has developed as the sum of the trail laid by the 15 ants that went that way from B and by the 15 ants that reached B coming from D via C (Figure 2c). The probability of choosing a path is now biased, so that the expected number of ants going toward C will be the double of those going toward H: 20 versus 10 respectively. The same is true for the new 30 ants in D that came from E.

This process continues until all of the ants will eventually choose the shortest path.

The idea is that, if at a given point an ant has to choose among different paths, those which were heavily chosen by preceding ants (that is, those with a high pheromone level) are chosen with higher probability. Furthermore, high pheromone levels are synonymous with short paths."

Applications of the Ant System

Most work on the Ant System has been applied to the Travelling Salesman Problem[1]. This is a classical problem and is often used for the assessment of heuristics as it is NP-Complete. Dorigo shows that the Ant System can be applied to the TSP, and other problems, which can be expressed in graph-partitioning terms. Much more detail on this work is presented in [Dorigo et al. 92, Dorigo et al. 96].

Dorigo's conclusions on the results of the Ant System on the TSP are:

1. Within the range of parameter optimality, the algorithm always finds very good solutions for all of the tested problems.

2. The algorithm quickly finds good solutions, while not exhibiting stagnation behaviour - the ants continue to search for new, possibly better tours.

[1] This, however, is changing and several potential applications are being researched.

3. With increasing problem size, the sensitivity of the parameter values to the problem dimension has been found acceptable[2].

The work on the TSP is directly transferable to the asymmetric TSP. This problem is significantly more difficult than the TSP. Typically the TSP can be solved for graphs with several thousand nodes, while the ATSP is only solved optimally in cases where there are several dozen nodes.

Application to the ATSP required no modifications to the basic AS algorithm. The results of applying the algorithm to this problem were very close to (within 3.3%) the known optimal tour, and were found within acceptable time.

The Ant System has also been applied to the Quadratic Assignment Problem (QAP) [Maniezzo et al. 94] and the Job Shop Scheduling Problem (JSP) [Colorni et al. 94]. These applications required very few changes to the algorithm, as long as the problem could be stated in an appropriate graph notation.

Ant System Conclusions

The Ant System is a new search methodology based on a distributed autocatalytic process and its application to the solution of a classical optimization problem. The general idea underlying the Ant System search paradigm is that of a population of agents each guided by an autocatalytic process directed by a greedy force. If an agent were to search alone, the autocatalytic process and the greedy force would tend to make the agent converge to a sub-optimal solution with exponential speed. When agents interact it appears that the greedy force can give the right suggestions to the autocatalytic process and facilitate rapid convergence to very good, often optimal, solutions without getting stuck in local optima. We speculate that this behaviour arises because information gained by agents during the search process is used to modify the problem representation. In some sense, the region of the space considered by the search process is reduced. Even if no tour is completely excluded, bad tours become highly improbable, and the agents search only in the neighbourhood of good solutions.

The main contributions of the Ant System are the following:

3. Positive feedback is employed as a search and optimization tool. The idea is that, if at a given point an agent (ant) has to choose between different options, and the one actually chosen results to be good, then in the future that choice will appear more desirable than it was before.

3. Synergy can arise and be useful in distributed systems. In AS, the effectiveness of the search carried out by a given number of cooperative ants is greater than that of the search carried out by the same number of ants, each one acting independently from the others.

[2] It is our experience that an AS can be improved by allowing control parameters to self-adapt during the search process. This is described in a later section.

3. The Ant System can be applied to different combinatorial optimization problems. Namely, AS provides a general search heuristic that can be applied to problems that can be represented as search on a graph. After introducing the AS by an application to the TSP, Dorigo and others have demonstrated how to apply it to the ATSP, the QAP, and the JSP.

Dorigo believes the approach to be a very promising, one because of its generality, and because of its effectiveness in finding very good solutions to difficult problems.

As already pointed out, the research on behaviour of social animals is to be considered as a source of inspiration and as a useful metaphor to explain our ideas. We believe that, especially if we are interested in designing inherently parallel algorithms, observation of natural systems can be an invaluable source of inspiration. Neural networks, genetic algorithms, evolution strategies, immune networks, simulated annealing are only some examples of models with a "natural flavour". The main characteristics, which are at least partially shared by members of this class of algorithms, are the use of a natural metaphor, inherent parallelism, stochastic nature, self-adaptation, and the use of positive feedback. Our algorithm can be considered as a new member of this class. All this work in "natural optimization" fits within the more general research area of stochastic optimization, in which the quest for optimality is traded away for computational efficiency.

Télécom Bretagne

Dominique Snyers and Pascale Kuntz, [Kuntz et al. 94], have been working on graph partitioning using Swarm Intelligence. The basic principles of the approach are similar to those adopted by Marco Dorigo.

The classical optimization problem of graph partitioning is transformed into a problem of dynamical co-adaptation of species. An artificial society of autonomous animats from different species is created, and the co-adaptation of the species leads to a territorial colonization. The colonization behaviour emerges by self-organization without any a priori fitness function computation: the model represents an example of intrinsic adaptation. The emergent grouping of the animat society coincides with the solution to the graph partitioning problem.

Kuntz and Snyers have verified experimentally that this model is minimal for the set of rules controlling the society for the graph partitioning problem. Simulation results have also shown that this distributed algorithm is robust. See [Kuntz et al. 94] for further details.

Santa Fe Institute

Work at the Santa Fe Institute is mostly focused on Artificial Life. The SWARM project is concerned with the modelling of multi-agent systems. The SWARM group is building a toolkit for experimenting with swarm based systems. The goal of the Swarm Project is to produce a fast, flexible, easy to use toolkit for experimenting with and researching systems where large numbers of agents explore the environment following very simple

rules. This work was originally lead by Nelson Minar, and is heavily supported by the "father of a-life" Chris Langton. This toolkit is being used by researchers in the field and free for academic use.

The environment may be a grid, rather like a cellular automaton, or be more constrained, such as a map. In addition, it is possible to model graph like systems such as telecommunications networks and have agents roam around these.

For more information, see the WWW page at URL: http://www.santafe.edu/projects/swarm. The SWARM project pages contain comparisons of several other multi-agent simulation systems and provide useful links to other, related research in the area.

3. Ant System Initial Studies

The purpose of this study was to assess the feasibility of applying swarm intelligence to telecommunication network dynamic routing. We expected to develop an algorithm in the context of ATM networks and a software demonstrator in order to show the importance of pursuing the research in this direction. In this respect, the goal has been achieved.

We shall first present the simplified routing problem covered by the three phases of this study. Detailed algorithms will then be presented and experimental results given. Following this, conclusions are then made and the potential for the further work explored.

Problem Description

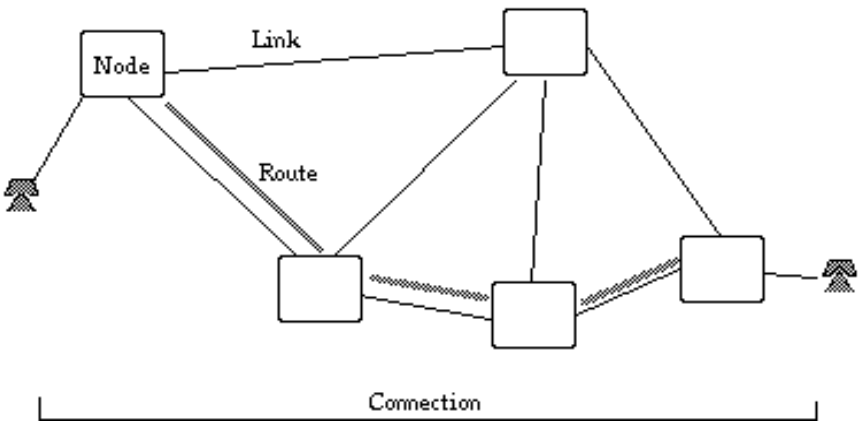
Routing is the problem of finding paths between nodes in a telecommunication network. In Figure 3 below, a telephone call is to be set up between two points by finding a route that connects the source and the destination. Links are able to carry traffic of a certain bandwidth up to the total capacity of the link. For this study, it is assumed that bandwidth is used through simple aggregation. Nodes are assumed to be capable of satisfying the switching demands placed up them. Links have an associated cost function. This is typically based on distance (e.g. traffic delay), and/or on error rates.

A connection request is characterized by the kind of traffic (voice, data, and video) which is required. This implies bandwidth requirements and potentially other constraints such as maximum error rate, maximum number of hops, or maximum delay. Different kinds of traffic can tolerate different delays and error rates, and these can be used as parameters in a cost function, along with the link's actual costs.

Point to multi-point routing is required in applications such as distance learning. In this case, a single source node transmits and receives information from a number of destination nodes, as in the case of a teacher giving a lecture to a number of students in a distance learning application.

The telecommunications network will be idealized here as a weighted graph where the vertices correspond to switching nodes, the arcs to the physical links, and the associated weights to the cost value associated to every link. Both point to point and point to multi-point routing amount to finding a minimum spanning tree in a graph.

Figure 3: A Telephone Connection



Description of the Algorithm

The swarm algorithm solution to this routing problem relies on the movements of artificial ants on the associated graph designed to make the global shortest path emerge. When a connection request is made, a new colony of ants is created and associated nests are positioned on the source and destination nodes. These artificial ants correspond to a special class of automata called reactive agents that react to their local perception of the environment by stochastically adopting predefined behaviours. The shortest path then emerges from the movement of all these ants. This corresponds to the first phase of the study that focuses only on establishing a single point-to-point connection.

There are three classes of ant. These are:

1. *Explorer* ants: these ants search for a path from a source to a destination.

2. *Allocator* ants: these ants allocate resources on the links used in a path from a source to a destination.

3. *Deallocator* ants: these ants deallocate resources on the links used in a path from a source to a destination.

As soon as we wish to establish multiple point-to-point connections, the problem becomes more constrained since the connections consume bandwidth and that after a while, some links might run out of available bandwidth. The extension is quite straightforward, the graph arcs have an associated available bandwidth and a condition is added for the ants to use a given arc: it should have enough bandwidth. Every time a path has emerged and a connection has been established the amount of available bandwidth is decreased on every arc of the path thereby adding additional bandwidth constraints to the graph.

Let us now define the local rules of the artificial ant automata.

Point to Point Routing

For this phase of the study, the algorithm is quite straightforward. Both source and destination nodes of the connection are considered as nests. The explorer ants leave the nest and explore the network following their local rules:

1. On each node, they choose a path with a probability proportional to the heuristic value (function of the cost and the pheromone level) associated with the link.
2. The ants cannot visit a node twice (they keep a tabu list of their visited nodes) and cannot use a link if there is insufficient bandwidth available.
3. Once the destination is reached, the ants return from whence they came by popping their tabu list. On their way back, they lay down a pheromone trail.

When the source node judges that a unique path has emerged, it sends a special kind of ant, the allocator, in order to allocate the bandwidth on all links used between the source and the destination.

Pheromone laid on a link will evaporate over time. This is controlled by a constant evaporation rate, r.

Multiple Point to Point Routing

This phase is similar to the previous one except that we have now several connections at the same time. Each connection corresponds to a different species and these species do not interact except by allocating bandwidth, and therefore, by imposing constraints on the other species by changing the environment.

Point to Multi-point Routing

Point to multi-point connection can be regarded as a multiple point to point connection starting from the same source node. The only modification to the phase two algorithm therefore, will concern the allocator. Rather than sending three allocators, one for each associated species, three identical allocators are sent from the source toward the destinations. Only the first allocator passing on the link will allocate the bandwidth, and fan out points (also known as multi-cast nodes) are created on bifurcation nodes.

This phase of the study considered both synchronous and asynchronous control mechanisms although we will only implemented a synchronous scheme. We believe that the fundamental underlying algorithm is well suited to asynchronous control and distribution.

Detailed explorer ant rules of behaviours

There are two phases to the movement of an explorer ant: an outward exploring mode and a backward trail-laying mode. The algorithm used by an explorer ant is shown below:

1. Initialize the route finding simulation
   Set t:= 0
   For every edge (i,j) set $T_{ijk}(t) := 0$
   Place m ants on the source node.
   Explorers agents are created at frequency e_f }
2. Set s:= 1 { tabu list index }
   For k:= 1 to m do
   Place starting node of the kth ant in $Tabu_k[s]$.
3. Repeat until destination reached:
   Set s := s + 1
   For k:=1 to m do
   Choose node j to move to with prob. $p_{ij}^k(t)$
   Move the kth ant to node j.
   Update explorer route cost: $r_k = r_k + C(i,j)$

If ($r_k > r_{max}$) then
Kill $explorer_k$
Insert node j in $Tabu_k[s]$.
At destination go to 4.

4. While s > 1
Traverse edge $Tabu_k[s]$.
$T_{ijk}(t) = T_{ijk}(t) + ph_k(r_k)$
s := s - 1

5. At source node do:
If the path in $Tabu_k$ is the same as p% of paths in PathBuffer then Create and send an allocator agent
If t > Tmax then
Create and send an allocator agent

In the above algorithm, the following symbols are used:

- $T_{ij}(t)$ is the quantity of pheromone present on the link between the ith and jth nodes,
- C(i,j) is the cost associated with the link between the ith and jth nodes.
- r_k is the cost of the route for the kth explorer agent.
- $Tabu_k$ is the list of edges traversed.
- Tmax is the maximum time that is allowed for a path to emerge.
- PathBuffer is the array of paths obtained by the (up to m) explorer agents.
- r_{max} is the maximum allowed cost of a route.
- $ph_k(r_k)$ is the quantity of pheromone laid by the kth explorer agent.
- $p_{ij}^k(t)$ is the probability that the kth agent will choose the edge from the ith to the jth node as its next hop.

The probability with which an explorer agent (k) chooses a node j to move to when currently at the ith node at time t is given by:

$$p_{ij}^k(t) = [T_{ijk}(t)]^{\alpha}[C(i,j)]^{-ß} / N_k$$

$$N_k = \Sigma_{j \text{ in } (S(i)- Tabu_k)} [T_{ijk}(t)]^{\alpha}[C(i,j)]^{-ß}$$

where α and β are control constants and determine the sensitivity of the search to pheromone concentration and link cost respectively. N_k is simply a normalization factor that makes $p_{ij}^k(t)$ a true probability. S(i) is the set of integers, {l} such that there exists a link between the ith and lth nodes.

The ant has two modes of behaviour. If travelling towards its destination, it find links at each node which the ant has not yet traversed, and have enough bandwidth available for this connection and selects a link from this set based on the probability function $p_{ij}^k(t)$.

Having selected a link, the selected link is added to the tabu list. The cost of the journey so far updated and then the link to the next node is traversed. An explorer ant that stays without being able to move for more than a given number of iterations simply dies. Similarly, an ant whose journey cost exceeds a given threshold also dies.

At the destination, the explorer ant switches to trail-laying mode. When travelling back to nest the ant pops the tabu list and moves over the link just popped dropping pheromone at a constant rate.

The source node maintains a set of statistics relating to the set of routes that have been found so far in both point-to-point and point to multi-point connections. This is achieved by querying the returning ants (and ants which are sent from the destination node to this node) about the path which they took. This information is maintained by their tabu list. The node records the frequency of ants following a particular route over a given time period (a moving window). It also records details such as the total cost of the route. A good route is one for which a proportion of ants in the current time window exceeds a specified limit, e.g. 95%. When the limit is exceeded, the node sends out an allocator ant that creates the connection by allocating resources in the network. If it does not succeed establishing the connection because, for example, another connection used all the available bandwidth, it simply backtracks. In the meantime, explorer ants continue to explore the problem space. The source node may send out an allocator ant again when the path emergence criteria are satisfied, or may choose to delay sending the allocator out again while the problem space settles to a steady state.

Short Study Experimental Results

Limited experiments have taken place. The section reproduces and discusses the results of a small investigation into the effects of the sensitivity parameters: α and β. It is prudent to question the validity of the results based on such a small sample size. However, observations over many runs show that the analysis of the results represents a reasonable picture of what happens on different graphs and connections. Our experiences using the simulation software suggest that the important parameters for the algorithm were the sensitivity to pheromone, α, and the sensitivity to cost, β. In order to investigate these, we decided to fix the other parameters. The values used are presented in **Table 1**:

Table 1: Simulation Parameters

Parameter	Value
Agent creation frequency	Every 10 cycles
Quantity of pheromone dropped	10 units
Emergence criterion	90% follow a path
Number of agents created	15
Path buffer size	40 agents
Pheromone evaporation rate	1.0 units/cycle
Maximum search time	300 cycles

The graph we used and its associated weights are shown in the Figure 4 below.

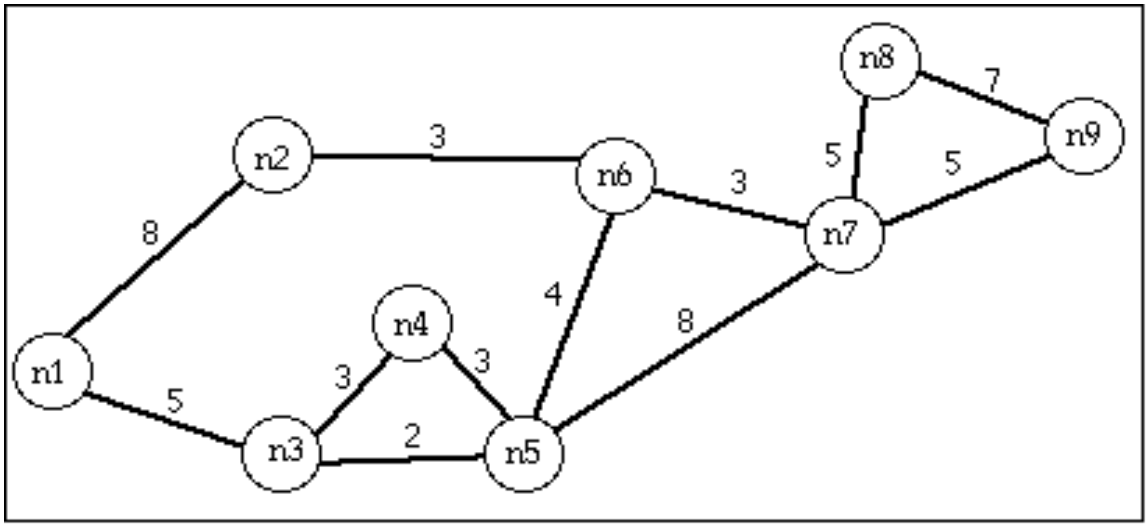


Figure 4: Initial Test Network

The initial test network contained only nine nodes and, in this phase of the study, links where assumed to have sufficient capacity to form the connection. The integer values associated with the links in Figure 4 are the cost of using that link in forming the connection. The connection requested in this study was from node n1 to node n9. One hundred runs were performed in order to assess the robustness of the algorithm. In all experiments, the path that emerged was n1-n2-n6-n7-n9. This route has the lowest cost (19) and shortest path (least number of hops) although the later is an artifact of the graph selected. Results for a selection of α and β values can be seen in **Table 2**.

Table 2: Results of Initial Experiments

α,β	2,1	2,2	4,2	8,2
Minimum	75	75	130	130
Maximum	400	220	195	190
Mean	220	175	159	150
Std Dev.	45	25	14	10

When the algorithm starts, the cost part of the probability function is important, and the greedy heuristic comes into play. The actual value of α is unimportant, as only a small amount of pheromone (approaching zero) is present on the links, therefore the only factor influencing choice of links is actual cost on that link. The choice of next hop depends almost exclusively on link cost.

As routes are found, pheromone is laid on the links that form the path. The amount of pheromone laid is inversely proportional to the total cost of the route, and acts as a global measure of its 'goodness'. This brings the reinforcement part of the probability function

into play. The sensitivity to pheromone, α, influences the choice links, and links with more pheromone are more likely to be chosen.

The first result set shows a broad spread of times to find solutions, with a very high standard deviation. The mean is also the highest of the set. The high SD means that confidence in a path is not high, and the system continues to use the greedy heuristic to explore other paths. In this particular case, we consider the SD too high. The second and third set of results show similar results for minimum, maximum and mean, but with standard deviation being the most different. The system finds results quickly, and reinforces good solutions, but confidence in the solutions found is at a level such that other solutions are not rejected, but continue to be explored. These seem to be the best sets of results. The final set of results show a fast, almost deterministic algorithm. This makes it undesirable for dynamic routing or as a candidate for a system that reacts to change quickly.

It is important that other solutions are explored and evaluated, as this is a stochastic approach. Reducing the standard deviation shows that the algorithm is more deterministic, which is not desirable. On the other hand, too high a standard deviation is also undesirable, as good solutions are not sufficiently reinforced.

Observations and analytical analysis show that with high values of α, the system becomes 'locked into' the solution found first, which in this experiment appears to be the best solution. This leads to problems later when bandwidth is removed from a link on that path. The system reorganizes, but when the bandwidth is reinstated, the previous solution, which is typically the best solution, is not found as the high sensitivity to pheromone is forcing the choice of paths where ants have previously been.

We conclude that suitable values for α and β are:

$\alpha=2$, $\beta=2$
$\alpha=4$, $\beta=2$

These ensure that a solution is found quickly, good solutions are reinforced, and that better solutions still have a reasonable chance of emerging.

Conclusions from the Short Study

The study has successfully shown point to point routing in a graph representing a telecommunications network. An algorithm has been developed with solves point to point and point to multi-point routing problems using a new stochastic heuristic method. This method is driven by a probability function that chooses between local cost and global pheromone information in order to determine the 'best' link for an ant to traverse.

Demonstration simulation software has been produced which implements the point to point algorithm. The software was used to generate the results presented in this

document. Results, both formal experiments and observations, show that the algorithm finds ‘good’ routes quickly, and continues to explore the network for alternatives. When alternatives arise -- for instance, when some bandwidth is made available -- the algorithm explores these (potentially) better solutions, and reinforces them if they are better than the previous best solution.

4. Further Enhancements to the Basic Algorithm

Introduction

This section discusses some of the finer details of the algorithm, and specific enhancements which improve the behaviour of the algorithm in a wide variety of circumstances.

There are two kinds of agents: Explorers and Allocators. Explorers explore the graph, selecting their movement according to probability function below. After some time, large proportions of the agents are likely to follow the same path. The criterion used to determine if a solution has emerged is initially quite simple: more than 90% of the agents have to follow the same path. This path is the solution given by the algorithm. The connection is then established, by reserving the required amount of bandwidth on each of the links of the solution. This is done by sending an Allocator send on that path which grabs the bandwidth for this connection. The connection is deemed to have been established when the allocator returns.

How agents choose the next link

The previous sections demonstrated that the desired agent behaviour could be achieved using a probabilistic function to govern the agents' movement. For a particular explorer located at a node in the graph, the probability of choosing a link (assuming enough bandwidth is available on that link) is proportional to the amount of pheromone on it. In addition, the 'cost' of the link is taken into account. For each link, the probability of selection is proportional to the amount of pheromone raised to the pheromone sensitivity (α), and multiplied by the cost of that link raised to the cost sensitivity (β).

$$\text{selectionProbability}_j = \frac{(\text{Cost}_j)^{-\beta}(\text{pheromone}_j)^{\alpha}}{\sum_i (\text{Cost}_i)^{-\beta}(\text{pheromone}_i)^{\alpha}}$$

The sensitivity to cost allows the agents to find better solutions in the early stages of the search (this is the so-called 'greedy heuristic'). If used alone, this will quickly lead to local optima, but due to the fact that there is a population of agents and a feedback mechanism (the pheromone trail), the search effort is distributed among the agents and tends to avoid the local optima. The use of both cost and pheromone sensitivity improves the results of the algorithm.

All the links connected to the current node are first considered. Then, the links already visited (i.e. in the *tabuList*) are removed. This *tabuList* is updated each time the agent moves: its previous position is added to the *tabuList*. This way, an agent can not go twice

on the same position. We use this to avoid cycles in the exploration. Then, the links without sufficient available bandwidth for the connection are removed from the search.

In the implementation of the algorithm, a selection value is computed for each possible next link. Its formula is the same as the one of the selection probability, but not normalized:

$$selectionValue = Cost^{(CostSensitivity)} \cdot pheromone^{(PheromoneSensitivity)}$$

Each selection value is multiplied by a different random number, and the line with the biggest result is selected as the next position. This, in effect, performs the normalization and probabilistic selection.

Pheromone

The way the pheromone is dropped on a graph is different than in nature with real ants. There are two modes for the explorers: forward mode and backward mode. In forward mode, the agents are attempting to find their way from their source to their destination. The path they have followed is stored in the *tabuList* variable. When they eventually arrive at the destination node, they switch to backward mode. In backward mode, they backtrack, retracing the path stored in *tabuList*. Agents only drop pheromone on their way back to the nest, in backward mode.

Contrary to the real ants, the aim is now to find the path with the lowest cost and not the shortest path. The amount of pheromone depends on the cost of the way they have found, and is inversely proportional to the cost of the path. In this way, a cheaper path will receive a larger amount of pheromone and will hence attract more agents. This is in effect a positive feedback loop.

Search using positive feedback would lead to run-away situations unless there were some other control mechanism. The mechanism used in this approach is pheromone evaporation. This way, previous solutions that have become out of date can be ‘forgotten’. If this were not the case, the algorithm would become locked onto an early solution (local optimum). Imagine that one path was discovered at an early stage but that, now, fewer and fewer agents use it. The evaporation process will ensure that the pheromone disappears if pheromone is not dropped quickly enough, and the new emergent ways will be favoured over it.

Source and Destination Nodes

For the implemented algorithm, half the agent population begin at the source node and search for the destination node; for the other half, the opposite is the case. Therefore, agents search from both source and destination nodes. This way, local optima due to the structure of the graph is minimized.

When explorer agents are created, they start at their source node. They look at all the links connected to that node, and chose the next based on the probability function described earlier. The agent moves from the current position to the next position, if there is one, or else commits suicide (as there is nowhere else for it to go). If it moves, the running total cost of the path is updated.

This repeats this until the explorer agent reaches it destination node. At that node, it switches to backward mode, and returns to the source node on the same path. It drops an amount of pheromone on each link. When it arrives at its source node, the connection monitor is updated with the values given by the agent. Then the agent is destroyed. This process continues until the connection monitor decides that a path has emerged.

When the connection monitor detects the emergence of a path, this path is allocated by sending an allocator agent on that path. The allocator follows the path, grabbing the required bandwidth for the connection. When it reaches the destination, it switches to the backward mode and returns to the source node, and informs the connection monitor that the path has been allocated.

Point to Multi-point Connections

Such connections have to be created when you want to send the same message from the same source point to many destination nodes. If you create *n* point to point connections between the source node and the destination nodes, in some lines you will have a high redundancy: you can repeat up to *n* times the same message and you will use n times the same amount of bandwidth. To avoid such a redundancy, you can allocate once the amount of bandwidth for one message on the common part of the path, and then duplicate the message when the different paths diverge. This provides an economy of bandwidth. The form of routing is equivalent to finding a minimum spanning tree of a graph.

In order to achieve this, each connection needs to know its associated connections (i.e. the other limbs of the point to multi-point connection). This is very important for the Allocator, which must ensure that the allocator agents created by the other connections do not grab bandwidth if others in that spanning tree have already done so.

Agent Species

In order to reuse what the previous connections have done, a species id was given to each connection. This species id was determined by the source and destination node and the bandwidth. Therefore, two connections between the same points (in whatever order) having the same bandwidth receive the same species id. The quantity of dropped pheromone is the same as before, but the quality is different.

Before, the pheromone was tagged with the identification number of the connection. Now, it is tagged with the identification number of the agent species. In that way,

connections with the same species id will use the same pheromone. For new connections, there may already be pheromone between the source and destination in the graph. The agents do not have to search from the beginning, as at least one solution to the search is already present. This modification can save considerable search time.

Load Balancing

The aim of load balancing is to find a route between the source and the destination, while trying to have a homogenous partition of the traffic. The aim is to avoid having empty links while others are full. The problem is then to favour a path with low occupancy. The first thing is to drop more pheromone on empty lines than on full ones. We modified the algorithm by multiplying the real cost of a link by a Cost Function. This function is based on the occupancy of the link (used bandwidth / total bandwidth).

Four cost functions were mainly used during all of these experiments. The first of them is the *defaultCostFunction*. This is a constant function and it is shown in Figure 5a. This function represents the cost of a simple routing algorithm without a special weight given to the occupancy of the link. This cost function is equivalent to the standard behaviour of the algorithm described previously.

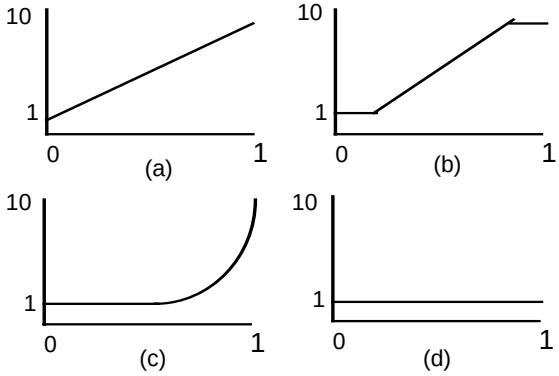


Figure 5: Cost Functions

$$\forall x \in [0,1],\ f(x) = 9x + 1$$

$$0.25 < x < 0.75,\ f(x) = \left(18x - \frac{14}{4}\right)^3$$

$$\forall x \in [0,1],\ f(x) = x^3(x + 8) + 1$$

Figure 6: Mathematical Cost Functions

The cost functions we investigated as shown in figures 5a, 5b and 5c share two characteristics. They return one if x equals zero and they return 10 if x equals one.

This normalization seems to compare the effects of each cost function for the same conditions. We have only chosen continuous functions.

The mathematical functions used for Figures 5a, 5b and 5c are given by the equations in Figure 6. The results of using this modification are presented in a later section. The mathematical form of Figure 6c requires some comment in that it closely resembles the response time characteristic associated with the closed form M/M/1 queuing result. Hereafter, we will refer to the functions in Figures 5a, 5b and 5c as *linear*, *non-linear* and *quartic* respectively.

The Genetic Algorithm-like approach

In certain cases, the absolute best path (we call it path #1) is not available due to very heavy traffic allocation on it. The algorithm therefore finds another path (path #2). During the time needed to find path #2, the agents have dropped a large amount of pheromone on the trail. Due to the pheromone sensitivity of the agents, this path is more likely to be followed. Sometimes, the agents can be attracted by that trail in such a way that they will not explore other links any more. The problem is that, if certain changes happen in the graph, for instance path #1 becoming available, the algorithm will not be aware of them because its agents will not explore new links.

To avoid to be locked into such a local optimum, the pheromone sensitivity has to be reduced at that moment. With lower pheromone sensitivity, the agents are more likely to explore other links. The problem is when to decide that the sensitivity has to be lowered and what new pheromone sensitivity to give to the agents.

Each explorer agent encodes its α and β sensitivity values that are used in the calculation of $p_{ij}^{k}(t)$. Initially, these values are selected randomly from a given range. Hence, the population of m agents initially sent out into the network has a range of sensitivity values. When these agents return to the source node, having found a route to a given destination, the route cost, r_k, is used to update the fitness value, $f(\alpha,\beta,k)$, associated with the (α,β) pair. The equation used to update $f(\alpha,\beta)$ is given by:

$f(\alpha,\beta,k) := f(\alpha,\beta,k) + \gamma(r_k - f(\alpha,\beta,k))$,
where $0 < \gamma < 1$.

It can easily be seen that an agent returning with a lower route cost than the current $f(\alpha,\beta,k)$ will cause $f(\alpha,\beta,k)$ to decrease. However, several agents must return with the same r_k value before $f(\alpha,\beta,k)$ approaches r_k. A discrete space for (α,β) was chosen in order to ensure that updates to $f(\alpha,\beta,k)$ would occur. A discounted feedback mechanism as shown above is required in this system because of the stochastic nature of the ant search. The same (α,β) encoding may result in several different r_k values and $f(\alpha,\beta,k)$ represents the average over all possible routes found in the network. Obviously, with γ set to zero it is possible to ignore previous searches with a given encoding.

As stated earlier, the source node retains a path buffer that contains m paths. The source node also retains m (α,β) pairs and their associated fitness values. When new agents need to be created and sent out to explore the network, the fitness values are used to create new (α,β) pairs. First, parent (α,β) encodings are selected based upon their $f(\alpha,\beta,k)$ values. The lower the value of $f(\alpha,\beta,k)$, the more likely the (α,β) encoding is to be chosen. A new encoding is then created by crossover and mutation operators using a mechanism

well known in Genetic Algorithms (GAs) [Goldberg 89]. Example crossover and mutation operators are shown in Figures 7 and 8.

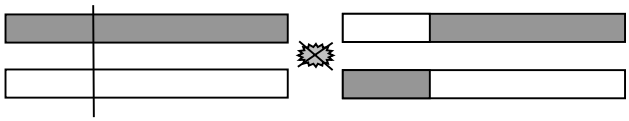


Figure 7: An example of crossover

The figure above shows the genotype of two ant parents that encode α and β. A single crossover point is chosen and two offspring are generated. One offspring is discarded and the other undergoes mutation as shown below.

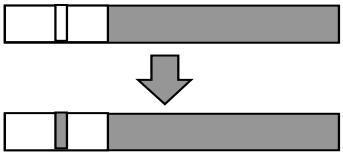


Figure 8: An example of mutation

The way we have achieved this is with a Genetic Algorithm-like process. As stated above, each agent has its own cost and pheromone sensitivity. At the very beginning, all the agents have random sets of parameters that are defined within a given range. When an agent returns, its set of parameters is stored along with the cost of the route found. Its parameters are linked to the cost of the path found. This cost has the same role as the fitness function of a GA. When creating a new agent, the sets of all of the last returning agents are considered. An intermediate population of parameters is created: each set has a probability of being chosen proportional to its fitness (the cost of the associated path). Some random parameters are automatically added to the population. The range of values for these random parameters is -0.25 to 3.0 for the pheromone sensitivity, and -0.125 to 1.5 for the cost sensitivity. Given that the encoding is a bit string, the spectrum of parameter values is discrete, a property which is essential for the use of the updating equation for $f(\alpha,\beta,k)$. The negative values allow agents to flee the main trail and therefore to explore new links. If these values are useful (i.e. they cause better paths to be found) they will be stored for future 'breeding', otherwise they will be forgotten. Then the genetic operators such as mutation and crossover are carried out. Finally, agents with the corresponding set of parameters are created and placed on the graph.

These random elements add an extra degree of variability to parameters and this is needed to avoid the convergence of the parameters.

We have observed that the algorithm is able to adapt to a new situation much more quickly when using these adaptive parameters. The time needed to discover the new path was about 2 or 3 seconds instead of half a minute to minutes before, depending of the value of the fixed pheromone sensitivity. Investigations with the adaptive parameter algorithm are ongoing and further experimental results will be provided in a future publication.

This approach differs from a conventional GA in that, in this algorithm, we are trying to *avoid* the convergence of the population because it tends to lead to local optima. This is perhaps closer to work on co-evolving populations because the environment of the agents (the network and its representation, the graph) is modified by their actions. There is considerable inter-play between the pheromone laying activities of one agent with the cost of a path found by another agent and, therefore, the fitness associated with the (α,β) encoding of pheromone and cost sensitivity values.

5. Experimental Results

In order to carry out reproducible experiments, a special connections manager was developed. With it, the user can choose the connections within the list of all the connections created which one he wants to use and in what order. He has to define the cost functions he wants to compare, and the number of runs. Then, the program will establish the first connection of its list. When the path is allocated, this connection is stopped (without removing its agents or its allocation of bandwidth from the network: the important point is that the modifications due to the former connection still remain in the graph) and the next connection is launched, and so on, until the list is empty. Then, results are saved in files. This is repeated for many runs, and then for another set of parameters. Many runs are necessary because the algorithm is not deterministic.

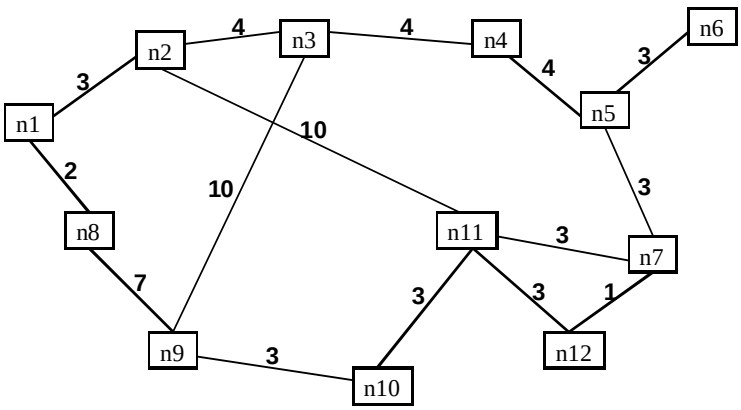


Figure 9: Network for experiment 1

The criterion used to measure if the load balancing of an experiment is good or not was the standard deviation of the occupancy of the links in the graph. A hypothetically perfect routing, with all the links at the same level of occupancy, will have a standard deviation in link utilization of zero. A very bad routing, with all the line empty but one having a utilization close to 100% will have a very high standard deviation. The criterion for good routing solutions *is the lower the standard deviation, the better the load balancing*. Therefore, we used the standard deviation as a means to measure the results of our experiments.

The use of a species id was not good for those experiments because instead of spreading, the connections having the same species id tried to follow the pheromone trail of its kind. This is the reason why we used connections with different bandwidth (even slightly - in the order of 0.1 difference) in all those experiments.

The first experiment (experiment #1) was on the following graph, Figure 9, with a set of 8 different connections between n1 and n12. In a second experiment (experiment #2), performed on the graph shown in Figure 10, we used another set of connections that seemed more realistic for us. In reality, you have a lot of connection using a small percentage of the total bandwidth. This set was made of:

- one connection whose bandwidth represents 20% of the maximum bandwidth available,
- 2 connections whose bandwidth represents 10% of the maximum bandwidth available,

- 4 connections whose bandwidth represents 5% of the maximum bandwidth available,
- 12 connections whose bandwidth represents 1% of the maximum bandwidth available.

For experiment #1, the first thing that was obvious, was that there was a spreading effect in using a cost function different from the *defaultCostFunction*. Secondly, it appeared that there was not one optimal cost function for all occupancy ranges. In fact, for an initial occupancy of 0%, the *linear* cost function gave the best results. For an initial occupancy of 25%, the *non-linear* cost function gave the best results. Finally, the *quartic* cost function provided the best results for the 50% initial occupancy.

We then tried to improve the algorithm. We had seen that multiplying the cost by a cost function could have a spreading effect, and then we tried to *force* this spreading effect. We modified the way the next link was chosen. In the basic algorithm, each possible next link has a selection value based on the probability function as discussed in previous sections.

In the enhanced algorithm, this selection value will be multiplied by a factor. This factor favours a link if it has a lower occupancy than the other possible next links (the factor has to be greater than 1 in that case). Conversely, it lowers the likelihood of selection if that link has a higher occupancy (the factor has to be lower than 1 in that case). We use an exponential function whose exponent is the difference of the occupancy of the current link with the occupancy of all the other candidates.

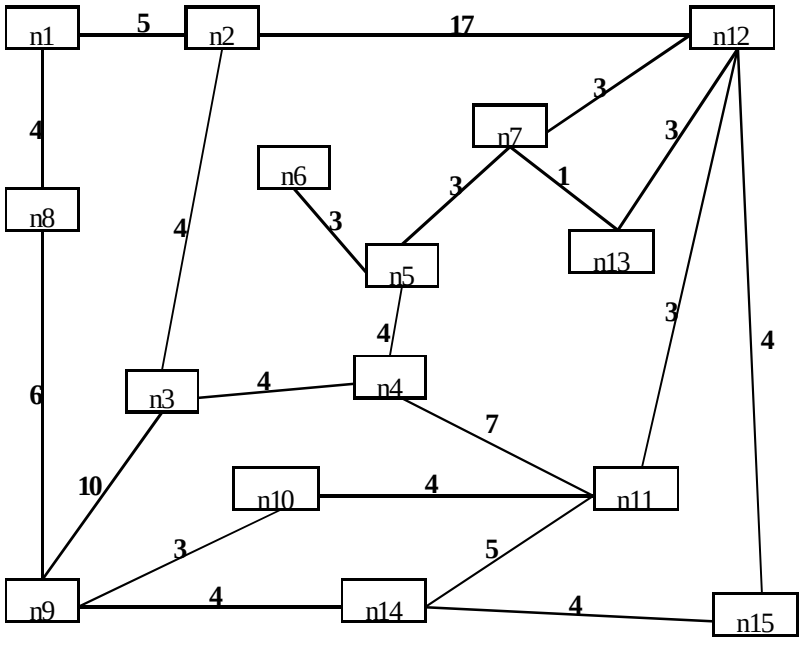
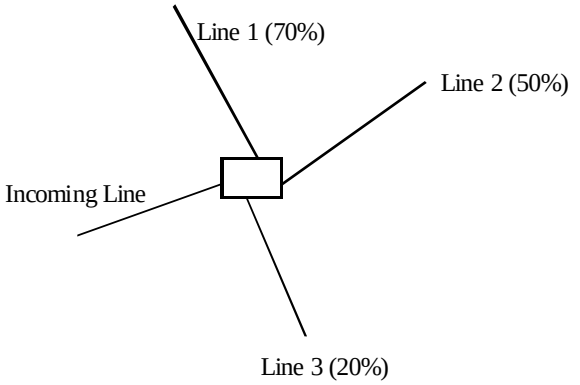


Figure 10: Network for experiment 2

$$\text{factor} = \prod_{i \neq j} e^{\frac{proportion(link_i) - proportion(link_j)}{T}}$$

The forcing factor is given by the equation above. As can be seen from the functional form, low occupancy links are exponentially favoured over high occupancy links. Figure 11 demonstrates the calculation of the factor for an example node. It can easily be seen that Line 3 with 20% has the best factor, whereas Line 1 has the worst, as expected.

In the experiments with this factor, the results of all cost functions were improved, but the ranking was different: for low occupancy, *the non-linear* cost function was better than the *linear* function. The *quartic* cost function was still the best for higher occupancy values. As a result of these experiments we were not sure that such an improvement for the standard deviation was worth if the total cost was too high, or the time needed too large.



$$factor_1 = e^{\frac{(0.5-0.7)}{T}} \cdot e^{\frac{(0.2-0.7)}{T}} = e^{-\frac{0.7}{T}}$$

$$factor_2 = e^{\frac{(0.7-0.5)}{T}} \cdot e^{\frac{(0.2-0.5)}{T}} = e^{-\frac{0.1}{T}}$$

$$factor_3 = e^{\frac{(0.7-0.2)}{T}} \cdot e^{\frac{(0.5-0.2)}{T}} = e^{\frac{0.8}{T}}$$

Figure 11: Example factor calculation

In order to assess the utility of the forcing factor, a number of experiments using the network in Figure 9 were performed. T was varied in these experiments, so we defined a total gain, taking into account the gain for the standard deviation, the total cost, and the time needed for the connection emergence. This is shown in Figure 12.

$$Gain_{SD} = 1 - \frac{SD}{SD_{Ref}}$$

$$Gain_{Time} = 1 - \frac{Time}{Time_{Ref}}$$

$$Gain_{Cost} = 1 - \frac{Cost}{Cost_{Ref}}$$

$$Gain_{Total} = Gain_{SD} + \frac{Gain_{Time}}{4} + Gain_{Cost}$$

Figure 12: T-factor gain equations

In the total gain term, the gain for time is divided by 4 because this parameter seemed to us less relevant than the cost and the standard deviation of occupancy.

The choice of the reference is highly significant for each gain. In order to be able to compare all four cost functions with different values for T, we selected the results of an experiment without factor effect, using the *defaultCostFunction* as cost function for the reference gain

measure.

From all these experiments, it appeared that there would not be an obvious winner. We can only choose the best compromise. In reality, the occupancy of network links are rather high, the best compromise could be using *quartic* cost function and T=0.5. In order to confirm that the results were not biased by the choice of the set of connections, we carried out experiment #2 and it appeared that the same set of parameters would be the best compromise.

Figure 13 shows the gain of each used cost function with varying values for T. From the left to the right, there are the results of our experiments with the following initial occupancy: 50%, 25% and 0%. The total gain is represented for each cost function (*defaultCostFunction*, *linear*, *non-linear* and *quartic*). These values are the mean values computed over 25 runs for each set of parameters.

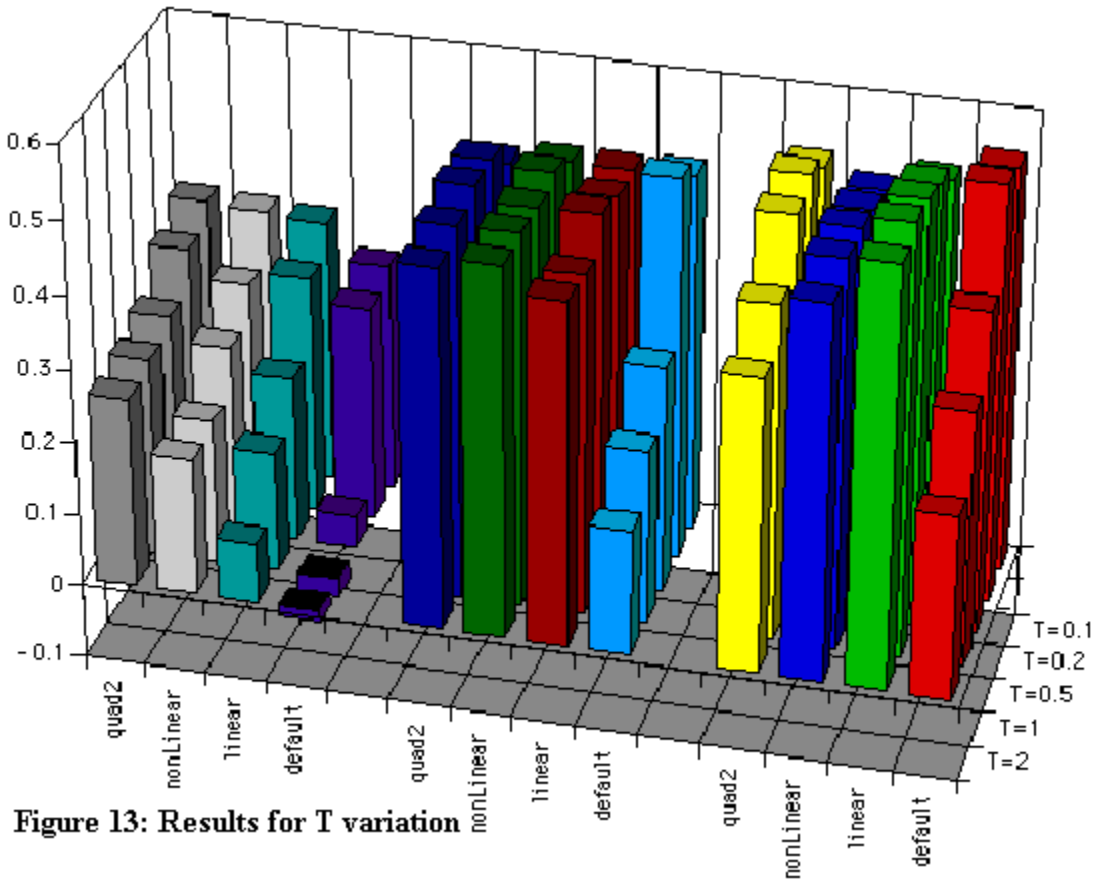


Figure 13: Results for T variation

6. Using this Approach on Real Networks

There are two primary mechanisms for utilizing this approach on a real ATM network or transmission network: finding routes on the network itself, and finding routes in the management system. The algorithm appears to work best on networks where connections are relatively long lived - in the terminology of ATM, permanent virtual circuits rather than switched virtual circuits.

Ants on the Network

For the algorithm to be able to operate on a real network, several mechanisms must be available:

- A method of assigning source and destination nodes and informing those nodes of their status (probably via a management system or embedded communications channel).
- A method of packaging agents, their execution state and their data so that they can be transported across a link (c.f. mobile agents in Odyssey or the framework designed here in the Systems Engineering Department [Susilo et al. 97]).
- On each node, a computing platform and a method for providing information about connected links to the agents.
- For each link (probably recorded at each end node) a mechanism for recording pheromone concentrations for each species and evaporating this over time[3].
- On the source node, a computing platform to determine whether a route has emerged.

All of the above points can be addressed with the application of Java, a mobile agent framework and the use of many of the Virtual Managed Component (VMC) ideas found in [Bieszczad and Pagurek 97, Bieszczad and Pagurek 98, Schramm et al. 98 and Susilo et al. 98].

Besides these requirements, some consideration must be given to the amount of bandwidth taken up by these agents. The agents themselves are relatively small, although there may be a large number of them circulating in the network at any time. In addition, there may be many connection requests being satisfied concurrently. The amount of bandwidth available in the network is likely to be several orders of magnitude higher than the total bandwidth available, and so in many situations, these concerns are probably unfounded.

However, when the network utilization is approaching its capacity, sending many agents may be deemed impractical, as it eats up too much bandwidth which could be sold to customers. From the previous sections, it is clear that the real benefits of the algorithm show when the network is approaching its capacity, and therefore it is valuable to retain the algorithm.

[3] Alternatively, evaporator agents could circulate and reduce the pheromone concentrations.

A compromise is to assign a small amount of bandwidth on each link permanently to the algorithm that will make best use of it, i.e. define an embedded management communications channel. Furthermore, when the network has sufficient capacity, the algorithm can make use of an additional amount to maximize its performance. If the network operator decides to use the final 0.01 percent of the bandwidth for network traffic, the swarm algorithm can be suspended and the management channel given over to network traffic.

A further benefit of having agents roam the network is that they can gather transmission delay and cell loss information. This can be fed back to the management system, as statistics about how well the algorithm and network are operating.

Ants in the Management System

The other approach is running the algorithm on the management system using snapshots either of the network, or by querying the nodes for information (although the latter is likely to be unacceptable). We can view the network snapshots as a simulation of the network.

This method is analogous to the demonstrator already implemented, in that the agents operate on a model of the actual network. However, as we deal with snapshots, it is possible that some information will be incorrect. The allocator agent must be analyzed and time dependencies between allocating on the network model and allocating real bandwidth on the actual network must be reduced.

This latter point raises the issue of the integration between the algorithm, the management system and the network. The algorithm results must be close enough to the real network in order for the solutions to be valid for any length of time on the real network. It is clearly not desirable to work for any length of time on an out-of-sync network model, nor is it for allocation to be attempted when the real network says that it is not possible.

This approach to integrating the algorithm with real networks is clearly easier and simpler than building agents that can roam around the network, and hence would be the recommended approach.

7. Conclusions

Outstanding Problems

Sensitivity to Parameters

The algorithm is very sensitive to both the agent parameters and the particular graph on which the algorithm operates. In order to reduce this, we focused on evolving the most important parameters (sensitivity to cost and pheromone). This was partly successful in reducing the sensitivity to the size and complexity of the graph.

Other parameters with can improve or reduce the efficacy of the algorithm are:

- number of agents created
- frequency of agent creation

These two parameters are very sensitive to the size and complexity of the graph. By default, 10 agents are created every 10 'ticks'. Often, it is better to create agents more frequently, *although it is not always the case that more agents lead to a better solution.* Indeed, these parameters are likely to be sensitive to more than the network; it appears that they are sensitive to the particular connection requested too. Further experimentation will be required to resolve this issue, although it is likely that some evolutionary approach may be beneficial.

- amount of pheromone to be dropped

The actual amount of pheromone dropped by an agent is this amount divided by a measure of how good the found route was (i.e. some function of it's cost). Therefore, this parameter is sensitive to the cost distribution on the graph, and is related to the 'Orders of Magnitude' problem, below.

- agent return buffer size (n)
- agent percentage required for emergence (m)

Deciding whether a path has emerged is a difficult problem. In this work, it is particularly difficult as, from the outset, we decided to avoid using a global monitor. Therefore, one of the nodes (the designated source node) was responsible for determining whether a path had emerged.

We used a simple scheme: of the last n agents to return, m followed the same route. This mechanism works reliably as a tool for detecting emergence. However, where there are several similar solutions to a problem, it takes a long time for one to dominate over another (it tends to oscillate).

A refinement this emergence criterion analyzed the amount of the graph that has been explored. This is done by recording the number of distinct routes that the agents find. As the search stagnates, fewer new paths are found (i.e. the standard deviation of the set of routes found approaches zero), we assume that no more new paths will be found, and then choose the current best available solution. Therefore, these parameters are not a particular problem.

Orders of Magnitude

There are two parts to this problem: cost magnitudes and pheromone magnitudes. Consider the graph below. Link AB has a cost of 1, while link AC has a cost of 100. This leads to the situation where AC is significantly less likely to be chosen than AB. It can be argued that this is exactly the desired behaviour. However, without careful consideration of relative costs on the network, this can lead to poor utilization of resources. Therefore, we recommend that costs on links be kept within one order of magnitude, as this will ensure good searching of the network. This may require a level of normalization before using the algorithm.

The other magnitude problem is to do with pheromone. When a large number of agents follow a particular route, a very large amount of pheromone can build up on that path. Indeed, the amount can be so great that it is impossible for the agents to escape that path. This problem is more significant than the cost magnitude problem, as it becomes very problematic on a network where connections come and go, freeing bandwidth on other links. When the agents become locked into a route due to a high amount of pheromone, and another, far better route becomes available, the agents will never find that route.

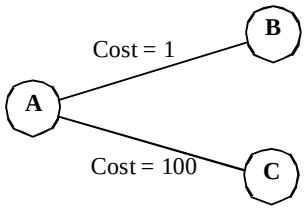


Figure 14: Order of magnitude problem

The use of a genetic algorithm allied to a simplified Q-learning mechanism served to avoid some of this, although it can still become locked into solutions. Further research on this problem is required.

Other Telecommunications Applications Areas

There are other applications within the telecommunications domain besides routing and load balancing, for which swarm intelligence may prove useful.

In addition to the quadratic assignment problem and the job shop scheduling problem, which have obvious uses within and outside telecommunications, some work has been done on temporal graphs and correlation.

The concept behind using swarm intelligence for alarm correlation is that alarms arrive separated in time and space, and this can be though of as forming a temporal graph. Finding correlations between alarms, i.e. diagnosing problems is equivalent to partitioning the graph into a number of cliques. Swarm Intelligence has been used for graph partitioning [Kuntz et al. 94].

The factors that effect the partitioning of alarms include:

- temporal closeness
- topological closeness — within and between network layers
- logical closeness — known direct causal relationships

These factors could be introduced into a swarm intelligence algorithm, and the sensitivity to each value can be varied using a power relationship with a sensitivity factor (raising to a power works well for routing and TSP).

As ants explore the graph, they will lay a pheromone trail representing the partitioning they have found. Other ants have a tendency to follow this trail, based on a sensitive factor, as for the other grouping factors.

Over time, the pheromone trails will be reinforced as more ants follow that path. This can be seen as a positive feedback search technique, which is controlled by evaporating the pheromone at a constant rate. This means that infrequently used relationships between alarms will evaporate toward zero, thus making their use more unlikely.

Solutions correspond to large accumulations of pheromone between nodes in a closed ring formation, the clique. The decision as to whether a large enough ring of pheromone has emerged has yet to be investigated.

Unlike routing problems where paths are limited to following arcs between nodes, the alarm correlation problem allows ants to explore any appropriate alarm nodes on the graph. Clearly, this can be large, but it is anticipated that the complexity of the problem can be reduced by using a sliding window technique. This removes ‘old’ alarms over time, but can also be modified to ‘forget’ alarms that have been resolved, or to remove ‘irrelevant’ alarms. This sliding window technique can be seen as a global controller over the algorithm.

Use of this graph partitioning technique should lead directly to solutions where there are multiple faults, as this corresponds to multiple partitions within the graph.

Summary

Our investigations and experiments have shown that this multi-agent search technique, called Swarm Intelligence, can solve routing problems on networks.

The main strengths of the algorithm are its robustness, the simple nature of the agents, and that it continues searching for new solutions even if a very good one was found. The algorithm also appears to perform well in problems with a large graph.

In an enhancement of the routing algorithm, we were able to obtain a load-balancing algorithm. All our experiments were on general graphs, therefore this kind of heuristic could be used as a general purpose tool (c.f. Dorigo's application of the same algorithm to the TSP, QAP, JSP with minimal changes).

The Swarm Intelligence dynamic routing algorithm showed strengths of versatility and generality. These can also be weaknesses: some other very specialized algorithms can outperform this algorithm. Its real strength is in the way it can adapt to new situations. Real ants are able to quickly find a new path when their environment changes, and the swarm algorithm also exhibits this behaviour. For instance, it can be shown that the explorer agents react quickly to changes in available bandwidth (and hence cost) by avoiding a particular link. This kind of heuristic would be very relevant in cases where the "environment" is changing rapidly, with a large flow of data. This is the case for problems like air traffic management [Ndovie, 93].

Adaptive behaviour could be used in managing incidents, or network failures. The fact that it can adapt to totally new situations may be very useful. This way, during the failure of some of its physical elements, a network would be able to manage routing. It is interesting too, that the agents can behave and act without human intervention. During network failures, this algorithm could behave as the "immune system" of the network and insure that during incident the routing of data would still be done.

It is our opinion that this work requires further 'hardening' before it can be used for industrial applications. This could take the form of further demonstrators or trials. The field of Swarm Intelligence is still rather small (<20 researchers worldwide) and immature. However, the potential benefits of this kind of approach are great: no overall control, simple agents, robustness, adaptation, and so this work is quite compelling. There are several powerful ideas contained within this algorithm and report that will be investigated by further research.

8. References

[Bieszczad and Pagurek 97] Bieszczad, A. and Pagurek, B., Towards plug-and play networks with mobile code, to be presented at the International Conference for Computer Communications ICCC'97, November 19-21, 1997, Cannes, France.

[Bieszczad and Pagurek 98] Bieszczad, A. and Pagurek, B., Network Management Application-Oriented Taxonomy of Mobile Code, to be presented at the IEEE/IFIP Network Operations and Management Symposium NOMS'98, New Orleans, Louisiana, February 1998.

[Beni 91] Key issues of the theory of cellular automata as applied to swarm intelligence, G. Beni, proceeding of the 1990 IEEE International Symposium on Intelligent Control, Philadelphia.

[Bonabeau et al. 94] Intelligence Collective, Eric Bonabeau & Guy Theraulaz, Hermes, Paris, 1994.

[Colorni et al. 94] Ant System for Job-shop Scheduling, Alberto Colorni, Marco Dorigo, Vittorio Maniezzo & Marco Trubian, Belgian Journal of Operations Research, Statistics and Computer Science, 1994.

[Dorigo et al. 92] Distributed Optimization by Ant Colonies, Alberto Dorigo, Marco Dorigo & Vittorio Maniezzo, European Conference on Artificial Life 1991, pp. 134-142.

[Dorigo et al. 96] The Ant System: Optimization by a colony of cooperating agents. Marco Dorigo & Alberto Colorni, IEE Transactions on Systems, Man and Cybernetics, Vol. 26, No. 2, 1996.

[Faieta et al. 94] Diversity and Adaptation in Populations of Clustering Ants, Baldo Faieta & Erik Lumer, proceedings of Conference on Simulation of Adaptive Behaviour, Brighton 1994.

[Goldberg 89] Goldberg, D. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley.

[Kuntz et al. 94] Emergent Colonization and Graph Partitioning, Pascale Kuntz & Dominique Snyers, proceedings of Conference on Simulation of Adaptive Behaviour, Brighton, 1994.

[Maniezzo et al. 94] Maniezzo V., A. Colorni and M. Dorigo, The Ant System Applied to the Quadratic Assignment Problem. Tech. Rep. IRIDIA/94-28, Université Libre de Bruxelles, Belgium, 1994.

[Schramm et al. 98] Schramm, C., Bieszczad, A. and Pagurek, B., Application-Oriented Network Modeling with Mobile Agents, to be presented at the IEEE/IFIP Network Operations and Management Symposium NOMS'98, New Orleans, Louisiana, February 1998.

[Susilo et al. 98] Susilo, G., Bieszczad, A. and Pagurek, B., Infrastructure for Advanced Network Management based on Mobile Code, to be presented at the IEEE/IFIP Network Operations and Management Symposium NOMS'98, New Orleans, Louisiana, February 1998. (Also available as technical report SCE-97-10).