

A ROUTING ALGORITHM INSPIRED FROM A DISTRIBUTED AUTONOMOUS MULTI-AGENT SYSTEM –THE ANT COLONY

Ruchir Jha

Department of Information Technology
Nirma University of Science and Technology
Chharodi, Ahmedabad – 382 481, Gujarat, India
E-mail: ruchir_jha@yahoo.com

KEYWORDS

Routing Algorithms, convergence rates, Ant Colonies, Meta Heuristics, base IRP.

ABSTRACT

Contemporary routing protocols are found to be less resilient to pathological conditions involving load variations and changes in local link metrics. [2] As an example, the Routing Information Protocol (RIP) presents poor convergence rates. At the same time for the Open Shortest Path First (OSPF), which was designed to provide connectivity among network nodes and to reroute traffic in case of sporadic node or link failures, frequent link metric changes can lead to wildly oscillating routing tables. Such facts thus, necessitate an approach to routing, which shall overcome the shortcomings of the routing algorithms currently in use. The paper discusses meta-heuristics based on the behavior of real ants in real time ant colonies, for routing to address these issues.

1. INTRODUCTION

Behavioral [3,4] characteristics of social animals like ants, swarms etc. have long served as meta-heuristics for solutions to stochastic combinatorial optimization problems. This is because such approaches deal with problems like premature convergence, population explosion in the solution space, etc better than conventional distributed-GA approaches. This is because Swarms and Ant-Colonies function as a decentralized and autonomous multi-agent system, which consists of myriad simple and cooperative individuals. As individuals, swarm agents (or ant colony members) behave with simple rules and based on locally perceivable information. As a collection, they run in a common environment and collaborate to achieve remarkable global goals. Their intelligence lies in the interactions among individuals and between the individuals and the environment. Ubiquitous computing as this might seem, routing in large-scale computer networks thus becomes a proposed domain for application of swarm and ant-colony heuristics.

However the ant colony routing algorithm (Antnet-RSV) stated in this paper shall interact with an IRP (Internal Routing Protocol) setup, to build on its path base. The IRP deployed at Layer-3 provides for all the basic network information requirements of the Antnet-RSV layer, which the former will provide to the latter, using its custom flooding algorithms, link-setup and connectivity testing mechanisms. Please note that the Antnet-RSV layer requires a base-IRP layer to establish a where's who know-how for the given subnet under consideration.

It can be recommended to rely upon a distance-vector algorithm, for the same, which will also minimize the routing node's "neighbor-base" building time. More, the fact that this will keep the node's form of information more local than global, may just add on as a plus, during the path-exploration phase of the ant, as we shall see in later sections. This can be considered metaphorical with the ant knowing about the whereabouts of the food source, though finding the shorter route to the same is a different issue altogether, which shall be considered hereby. In the last section, I conclude by giving an algorithm based on Darwin's Theory of Natural selection, applied to, finding a routing solution from the given solution-space.

2. THE ANTNET-CL ALGORITHM

The algorithm reported here, based in a connectionless framework, follows the work of Di-Caro and Dorigo (Di-Caro, Dorigo-1998) and is informally summarized as follows:-

Each node of the network retains a record of packet destinations as seen on data packets passing through that node. This is used to launch periodically, however asynchronously, launch "forward" ants with destinations stochastically sampled from the collected set of destinations.

Once launched the forward ant uses the routing table information to make probabilistic decisions regarding the next hop to take at each node. While moving forward, on every node the ant takes a timestamp and the node-identifier information. The same is later used

to update the routing tables along the path followed. If a forward ant is found to be traveling in a loop, that is it re-visits a given node, it is killed. On reaching the destination node, the total trip time is estimated, and a backward ant is created. The backward ant returns using the same path as followed by the forward ant, though using a priority queue, and not the queue of the data-packets. At each node on its path, the backward ant makes updates to the node's routing table to reflect the relative performance of the path. When the backward ant reaches the source, it "dies", and this can be implemented as some variant of IP's TTL.

2.1. Analysis

However the above-stated schema experiences some shortcomings. The above scheme first of all assumes that the routing tables maintained at the nodes are accurate, which in reality is never the case. If a node instantaneously goes down, or in the face of some sporadic link failure, the time taken for the network to converge might induce substantial overhead. Moreover, for a large network, with many routing nodes, the ants going out on links might need to maintain information quite global in nature (viz. the number of nodes in the network). The amount of information carried by one particular ant agent in the network would be huge for a large network. Both of these fundamentally violate the base on which the routing metaphor with behavior of social insects, stands. (Sec-Introduction). This necessitates use of an IRP employing a DV algorithm (distance vector algorithm), which then emphasizes on the performance expectations of the ant!. As it would be surely uneconomical to use this algorithm, which might then do nothing except eat up a considerable slice of the node's processing time.

3. THE ANTNET-RSV ALGORITHM

The work described in this section derives its inspiration from [Gross et al 1992, Beckers et al 1989], though the fundamental approach used is discussed in [White 96]. The algorithm works in the framework of resource-reservation and hence the 'RSV'. This approach uses three types of ant agents, namely "explorers", "allocators", "deallocators". Explorer agents exhibit the foraging behavior of ants and preferentially follow trails of pheromones laid down by previous explorers. Allocator agents traverse the path determined by explorer agents and allocate the bandwidth on the links used in the path. Similarly, when the path is no longer required deallocator agents traverse the path and deallocate the bandwidth used on the links. When explorer agents reach their destination they backtrack along the route chosen and drop pheromone in order to mark the path. Upon arrival back at the source node a decision is made whether or not to

send an allocator agent. The decision is made based upon m previous allocator agents' paths. If $p\%$ (where p is the pheromone intensity on the selected path) of the agents follow the same path, the path is said to have emerged and an allocator agent is created and enters the network in order to allocate bandwidth. Allocator agents traverse the path indicated by the highest concentrations of the pheromones dropped by their associated explorer agents.

- 1) *Create an explorer at a frequency 'f', with the destination selected, as a function of probability p, where*

$$P = \frac{\text{Traffic}_{i \rightarrow k}}{\sum_{s \rightarrow d} \text{Traffic}}$$

- 2) *Initialize stack S for ant agent 'A'*

S [top] → 0;
Top[node_identifier]=current_node
Top [time_stamp] = 0;

- 3) *Choose next-hop on the basis of routing table entry in the currently visited node with a probability Q.*

$$Q = \frac{P_{I \rightarrow k} + \Psi [L_N]}{1 + \Psi | \text{Neighbors}_C - 1 |}$$

- 4) *Create backward ant B, such that the values of stack S are now popped, and the ant on its way back using high-priority queues updates its pheromone tables.*

While (S [Top] < 0)
Top--;
Update (P_table (m, n));

- 5) *Create allocator ant at a time interval equal to some integral multiple of total time spent on exploration and scan the pheromone table. "Sniff" for highest pheromone value, initialize stack for allocator and put it on the link.*

[Fig1- Modified AntNET-RSV]

It is possible that network bandwidth has already been allocated by the time the allocator agent is sent and in this case the allocator agent backtracks to the source node rolling back resource allocation and decreases pheromone levels such that a later. A decision to re-send an allocator agent is made at a later time after a

back-off period has been observed. During the back-off period explorer ants continue to search for routes. Fig-1 provides the trace for the same.

In Step-3 of the above algorithm, the expression for 'Q' is explained hereby.

The Ant-Colony system is a society of cooperating individuals, where agents form their perceptions by making current observations at the same time, backtracking for previous perceptions formed, when a predecessor agent was faced with the same choice. This is effectively implemented with the help of this expression.

$P_{I \rightarrow k}$: When an explorer reaches a given node, the probability with which it selects its next hop is given by this variable. Its value is given by:

Pheromone (source [I] → destination [k])

Σ Pheromone (source → destination)

The pheromone values are nothing but the route preference values set by the backward ants, which had traversed the same route before the current explorer. As there can be many routes to the same destination, the above expression is evaluated.

Ψ : This is the heuristic correction factor, and it has been experimentally determined as that for the value of Ψ ranging between 0.2 and 0.5, the algorithm converges optimally.

$L_N = 1 - \text{Queue (source[I] → destination[k])}$

$\Sigma_{(n=1 \text{ to } \text{Neighbor}\{\text{current}\})} \text{Queue}^N$

As described above, the algorithm needs to maintain a perfect blend between the currently perceived values and those learnt from backtracking. The above value provides for the same. It's a measure of the queue-length on each outgoing interface of the given node, at the given instant.

Neighbors c .

It gives the number of neighbors of the current node, which the explorer is visiting at the given instant.

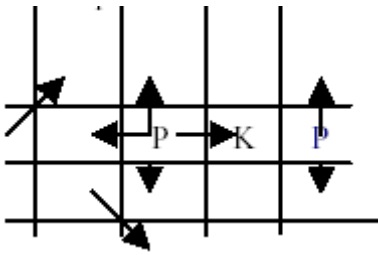
The given algorithm provides for filling the loopholes of the previous approach as discussed in 1.2. Furthermore, such an approach can be used for varied types of Internet applications seeking higher reliability of communications. However, there is one more issue to which the algorithm's resilience is not as it should be. After the exploration phase is completed, and when its time to send out the allocators, it's a possibility that the

agent's choice of path for resource allocation might wildly oscillate between two given choices, having the same pheromone intensity. One solution to this problem can be stated as assigning an 'age' parameter to every route added in the pheromone table. However for large networks with thousands of exploring agents returning to the same node at a given instant, this solution may not be quite convergent.

To combat such superfluous routing situations, an algorithm is stated in the next section, which draws its inspiration from the **Theory of Natural Selection**, and belongs to the Cellular Genetic Framework.

4. ALGORITHM-SURVIVAL_OF_THE FITTEST

Genetic algorithms are efficient algorithms for searching complex fitness landscapes inspired by biological evolution. GA-s has been successfully applied to various complex optimization problems. However while using GA-s a problem presents itself, which is premature convergence, which is as a result of rapid loss of population diversity in the GA search space. And it's because of this that the search oscillates between sub-optimal solutions [Li, X. and Sutherland]. However CGA's are said to be effectively preserving population diversity. In a cellular GA, individuals are mapped onto a 2-dimensional lattice, each cell corresponding to each individual. The operators of selection and crossover are restricted within the local neighborhood of each individual. The "isolation-by-distance" method in a way allows a slow diffusion of good genes across the lattice, thereby producing individuals capable of adaptation to the then problem environment. The model I thus present is based on a selection method inspired by predator-prey interaction dynamics. Such dynamics contributes to maintain selection pressure and therefore population diversity. In this model, the prey, which represent, potential solutions, are free to move around a 2-dimensional lattice and breed with other prey. The selection pressure is maintained by predators, which roam around the lattice and consume the weakest prey in their vicinity. This kind of selection procedure efficiently maintains healthier prey, richer on the fitness factor over successive generations. The experimental results of this procedure have yielded more productivity than the conventional GA. The robustness of this mechanism lies in the fact that it relies on a dynamic spatial structure. This is quite in contrast to the conventional GA-static spatial structure, which one may observe. Furthermore this strategy maintains selection pressure by killing off prey rather than the direct replacement theory of replacing the least fit individual with the fitter one. This in a way helps implement an essential degree of population diversity along with check on a possible population explosion.



[Fig-2 2-dimensional lattice for the Prey-predator interaction]

This model consists of a 2-dimensional lattice where the prey and predator population resides. Both can occupy one cell at a time. Initially a large number of prey are distributed across the lattice. Then the model follows the following steps:

Step: 1 Each prey is given the chance to move into one of the neighboring cells (where each cell has a probability of occupation set to 1/8 for a typical Moore Neighborhood) But we define a variable **RandomMoveProbability** = 0.5, for each prey individual to move, so that half of them would move one step on the lattice and the other half would stay where they were. This value is given to stop the scenario from slipping into a “population explosion” mode, where every prey breeds, as its got the chance to move. If the cell, a prey is trying to occupy is already occupied, the attempt is aborted, and the latter tries again, and the process may continue until a random upper bound (in our case 10) is reached. One might note, that varying the value of the upper bound, one might achieve a visible trade off between size and quality of population under consideration.

Step: 2 Each one selects from its neighbors the fittest one and breeds with the same of course excluding its own-self. If the prey has no neighbors, its not allowed to breed, otherwise the prey breeds, using the traditional operators of mutation and crossover [as shall be discussed in the next section]. The offspring generated is placed on a randomly selected unoccupied cell, and the placement policy implicitly suggests that the offspring can face a radically different breeding environment as compared to its parents. This fulfills the objective of maintaining diversity among the existing population.

Step: 3 We now enter the hunt. The predator kills its weakest neighbor prey. If a predator has no prey, it moves exactly the same way as the prey. However the predator can move more than once prey time step.

Step: 4 Go back to Step-1 and re-enter the reproduction phase if sufficient number of evaluations is not reached, else continue. The importance of this step lies in the fact that we may need to balance between the predator and prey populations. For this we adopt the following formula where extent is the number of moves a predator can make before the prey. Thus one can envision the algorithm as an iterative switch between the prey and predators in locomotion.

$$\text{Extent} = (\text{Actual_no_of_prey} - \text{Preferred_no_of_prey}) / \text{Number_of_predators}$$

A predator can kill one prey per unit increment until the value of extent is reached. For example there are 450 prey, and the preferred number is 120 and the number of predators is 80 then the extent to which predators can stay in the hunt phase is ‘4’. However after the hunt phase, the algorithm again slips into the prey reproduction phase. A potent benefit of using the above equation is, that the algorithm always has a check on the total population size, as well as, even if the Actual_no_of_preay value reaches an eventual minimum, the child prey generation is always being injected into the fitness landscape, producing better off springs.

Implementation:

The first question one might ask for the above stated procedure is what is prey and predator in the court of routing optimization.

Let

W_{\max} = maximum weight associated with a link

W_{flow} = weight associated with a given flow.

$\text{Pheromone}_{\text{path}}$ = Pheromone value in the routing table for the path under consideration.

U_{\max} = maximum link utilization

μ_{Rerouted} = no. Of rerouted flows associated with the given change made in routing pattern

Thus, consider the following objective function:-

$$\Phi = (W_{\max} * U_{\max} + W_{\text{flow}} * \mu_{\text{Rerouted}}) / \text{Pheromone}_{\text{path}}$$

A deficiency of OSPF based traffic-engineering lies in the transient behavior while changing routing pattern from one metric to another. After metric modifications are detected, ISPF routers may plausibly distribute the new link state information after having performed recomputation through the shortest path algorithm. During this transition phase inconsistencies might arise, affecting active connections, which need to be rerouted which in turn maybe through increased packet loss or

packet reordering. Therefore its advisable to consider the amount of rerouted traffic or the number of affected flows, while considering the problem of making routing algorithms adapt and converge. Thus the same is considered in the expression of ϕ .

Prey:

The objective function ϕ can be vectorized to represent a prey individual like:

$(W_{max}, U_{max}, W_{flow}, \mu_{Rerouted}, Pheromone_{path})$

Such prey values can be distributed across the 2 dimensional lattices as a random function like:

Distribution (I, J) = Random (I*j) [Prey (I, J)]

Predator:

The predator should logically consist of two modules. Finding its weakest prey, and second to kill it. This can be implemented as a set of 2 functions as below:

Hunt (I, J) = (I / U_{rerouted})^p

As one may possibly want to minimize the maximum number of flows rerouted we choose the inverse of this parameter as our fitness function. This way routing solutions with smaller maximum link utilizations receive higher fitness values, and thus have a higher chance to be reproduced when a new generation is being set up.

In order to influence the reproduction phase we apply power scaling to the above function. With $p < 1$, we can achieve that fitness values of bad solutions are increased relatively to the best ones, thus avoiding that they die out too fast and that the optimization procedure converges too early. For $p > 1$, the gap between good and bad solutions is increased, forcing the process to converge optimally.

4.1. Breeding- Crossover and Mutation

In this section, we adopt a real coded GA for the predator-prey model; each prey individual represents a chromosome, which in turn is a vector of genes, which are floating point numbers. The CGA based on the predator-prey model works similar to its binary counterpart, except that the crossover and mutation operations are slightly different. The real coded crossover involves two functions. The first behaves similar to the standard crossover operator. The difference is that instead of swapping binary values the values in the slots of floating point array or in other words the gene segments present on chromosomes is swapped. For example if we have two parents $p1 = (x1, x2, x3, \dots, xn)$ and $p2 = (y1, y2, \dots, yn)$, and the crossover

point is between $X1$ and $X I + 1$, then one child corresponds to $C1 = (x1, x2, \dots, y I + 1, \dots, yn)$ and $C2 = (y1, y2, \dots, x I + 1, \dots, xn)$. We apply this operator to 50% of the prey population. The second operator is called as blend operator (BLX- α) first introduced by Eshelman and Schaffer. BLX- α generates a child $c1 = (C1, C2, \dots, CN)$, such that Ci is a randomly chosen floating point number from the interval

[Min I - $\Delta * \alpha$, Max I + $\Delta * \alpha$]. Its been their observation that highly optimal convergence rates are achievable for $\alpha = 0.5$.

5. CONCLUSION

The algorithm discussed in this paper can be effectively used to resolve the situation encountered by Antnet-RSV, when the pheromone values, found by the allocator agents for the same destination, are same, for two distinct paths.

6. FURTHER ENHANCEMENTS

Implementing an implied metaphor of the ant-routing mechanism would be, implementing the evaporation of pheromone trails on the links, in order to minimize the $\mu_{rerouted}$ value. This is because, in the real ant colony system, pheromone trails being liquid in nature, evaporate after a given time interval, thus forcing the ants to make the same routing decisions again.

7. REFERENCES

- Schoonderwoerd. 1996. Hewlett Packard Labs, Bristol-UK, Ant System Implementation- HP-Report.
- Goss S.; Beckers R.; Deneubourg J.L.; Aron S.; Pasteels J.M. 1990. "How Trail Laying and Trail Following Can Solve Foraging Problems for AntColonies, in Hughes R.N. (ed.) NATO ASI Series, Vol. G 20, Behavioural Mechanisms of Food Selection, Springer Verlag, Berlin.
- Grassé P.P., 1959. *La reconstruction du nid et les coordinations inter-individuelles chez Bellicositermes natalensis et Cubitermes sp. La theorie de la stigmergie: Essai d'interpretation des termites constructeurs. In Insect Societies*, Vol. 6, pp. 41-83.
- Hölldobler B. and Wilson E.O., 1994 *Journey to the Ants*. Bellknap Press/Harvard University Press.
- Shapiro J. A. 1988. *Bacteria as multi cellular organisms*, Scientific American, 1988 pp. 82-89.
- White A.R.P., 1996. *Routing with Ants*, Nortel internal report.
- Eshleman Schaffer. 1990. Real coded Genetic Algorithms for real-time problem schema
- Wright-A. 1991. Genetic Algorithms for real parameter optimization.
- Li, X. and Sutherland, S. (2004). A Real-Coded Cellular Genetic Algorithm Inspired by Predator-Prey Interactions", In Kay Chen Tan, Meng Hiot Lim, Xin Yao,

and Lipo Wang, editors, Recent Advances in Simulated Evolution and Learning, Advances in Natural Computation, pages (to appear). World Scientific 2004

RUCHIR JHA was born in Ahmedabad, Gujarat, India, in 1983. He went to Nirma University, where he

is currently an undergraduate student enrolled in the Bachelor of Engineering in Information Technology course. His interests lie in the fields of Artificial Intelligence and routing simulations. He is currently working as a trainee in Hewlett Packard-India.