Ant Colony System Based Dynamic Routing In A Network

Nimish Singh Deepti Srivastava

Department of Information Technology MCSCET Lucknow , INDIA

Abstract

There has been a lot of research on the application of Swarm Intelligence to the problem of adaptive routing in telecommunications networks in the recent past. A large number of algorithms have been proposed for different types of networks, including wired networks and wireless ad hoc networks. In this paper we focus on the application of Swarm Intelligence design to one particular class of optimization problems, namely *adaptive routing in telecommunications networks* .We discuss both the principles underlying the research and the practical applications that have been proposed. Then we present Ant Colony Routing Algorithm which is used for evaluating minimum cost function for a given network which is adaptive.

Introduction

Routing is the process of scheduling paths in a network along which network traffic is sent from source to destination[1].The routing process usually directs forwarding on the basis of routing tables which maintains the routes to particular network destinations, and in some cases, metrics (distances) associated with those routes.

There are two types of routing techniques. One is **Static Routing or Non Adaptive Routing algorithms** which allow routing tables in specific routers to be set up by the network administrator. Another is **Dynamic Routing or Adaptive Routing Algorithms** that use Routing Protocols that dynamically discover network destinations and how to get to them. Dynamic routing allows routing tables in routers to change if a router on the route goes down. They get information locally from adjacent routers. A routing protocol specifies how routers communicate with each other, broadcasting information that enables them to select routes between any two nodes on a computer network.

Two popular dynamic routing algorithms are Distance Vector Routing and link state routing.

In Distance Vector Routing each router sends a vector of distances to its neighbors. The vector contains distances to all nodes in the network.

In Link State Routing each router sends a vector of distances to all nodes. The vector contains only distances to neighbors.

The main objective of all the routing algorithms is optimality, simplicity, low overhead, robustness, rapid convergence and flexibility. Optimality refers to the ability of the Routing Algorithm to select the best route .The best route depends on the metrics and metric weightings used to make the calculation. **Optimality principle** is the basic principle of dynamic Algorithm, which was developed by Richard Bellman which states that if router B is on optimal path from router A to router C, then the optimal path from B to C also falls along the same route. Suppose the route from A to B is r1 and the rest of the route is called r2.If a route better than r2 exists from B to C, it could be concatenated with r1 to improve the route from A to C, so that r1r2 is optimal.

In this paper I propose a methodology of implementing dynamic routing by evaluating minimum costs between the two adjacent hops using the ideas of Swam Intelligence. Swarm intelligence is the study of computational systems inspired by the ‘collective intelligence’[3]. Collective Intelligence emerges through the cooperation of large numbers of homogeneous agents in the environment. **Swarm intelligence** is a relatively new discipline that deals with the study of self-organizing processes nature and in artificial systems. Recently, algorithms and methods inspired by these models have been proposed to solve difficult problems in many domains. An example of a particularly successful research direction in swarm intelligence is ant colony optimization which takes inspiration from the pheromone-mediated ability of ant colonies to find shortest paths between their nest and sources of food to define a metaheuristic for combinatorial optimization based on the use of ant-like agents and stigmergic communication of artificial pheromone information.

Ant colony optimization (ACO) has been applied successfully to a large number of difficult discrete optimization problems including the traveling salesman problem, the quadratic assignment problem, scheduling, vehicle routing, etc., as well as to routing in telecommunication networks.

Another dominant sub-field is **Particle Swarm Optimization (PSO)** that investigates probabilistic algorithms inspired by the flocking, schooling and herding. Particle swarm optimization is a population-based stochastic approach based on information-sharing particle-like agents for solving continuous and discrete optimization problems. In particle swarm optimization, simple software agents, called particles, move in the search space of an optimization problem. The position of a particle represents a candidate solution to the optimization problem at hand.

Consequently, a lot of successful adaptive routing algorithms have been developed based on Swarm Intelligence ideas. Our aim here is to discuss the relationships between Swarm Intelligence paradigm and to design a Dynamic routing algorithm with minimum cost function.

Working Methodology

Procedure to compute minimum cost function using Swarm Intelligence Algorithm provides steps of the main Ant Colony System Algorithm for minimizing a cost function.

An ant is a simple computational agent in the ant colony optimization algorithm. It iteratively constructs a solution for the problem at hand. The intermediate solutions are referred to as solution states. At each iteration of the algorithm, each ant moves from a state x to state y, corresponding to a more complete intermediate solution. Each component can only be selected if it has not already been chosen (for most combinatorial problems), and for those components that can be selected from given the current component i, their probability for selection is defined as:

$$p_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}$$

where ,

ηxy is the maximizing contribution to the overall score of selecting the component (such as 1.0 distance x,y for the Traveling Salesman Problem),

β is the heuristic coefficient to control the influence of η_{xy} (commonly fixed at 1.0),

$\tau_{xy}$is the amount of pheromone deposited for transition from state $x$to y,

α is the history coefficient to control the influence of τ_{xy} and

c is the set of usable components.

A greediness factor (q0) is used to influence when to use the above probabilistic component selection and when to greedily select the best possible component.

Pheromone update

The first ACO algorithm was called Ant System and it was aimed to solve the travelling salesman problem, in which the goal is to find the shortest round-trip to link a series of cities. The ACO differs from the previous ant system because of three main aspects:

i) The **state transition rule** provides a direct way of balancing the exploration of new edges and exploitation of a priori and accumulated knowledge about the problem,

ii) The **global pheromone updating rule** is applied only to edges which belong to the best ant tour, and

iii) While ants construct a solution a **local pheromone updating rule** (local updating rule, for short) is applied.

Informally, the ACS works as follows:

i) Ants are initially positioned on cities chosen according to some initialization rule (e.g., randomly). ii) Each ant builds a tour (i.e., a feasible solution to the TSP) by repeatedly applying a stochastic greedy rule (the state transition rule).

iii) While constructing its tour, an ant also modifies the amount of pheromone on the visited edges by applying the local updating rule.

iv) Once all ants have terminated their tour, the amount of pheromone on edges is modified again (by applying the global updating rule).

v) As was the case in ant system, ants are guided, in building their tours, by both heuristic information (they prefer to choose short edges) and by pheromone information. An edge with a high amount of pheromone is a very desirable choice.

vi) The pheromone updating rules are designed so that they tend to give more pheromone to edges which should be visited by ants.

The ACS algorithm is reported below. In the following we discuss the state transition rule, the global updating rule, and the local updating rule.

A local pheromone update is performed for each solution that is constructed to dissuade following solutions to use the same components in the same order, as follows:

$$\tau_{x,y} \leftarrow (1 - \sigma) \times \tau_{x,y} + \sigma \times \tau_{x,y}^{0}$$

where

$\tau_{i,j}$ represents the pheromone for the component (graph edge) (i, j),

σ is the local pheromone factor and

$\tau_{x,y}^{0}$ is the initial pheromone value.

At the end of each iteration , the pheromone is updated and decayed using the best candidate solution found thus far (or the best candidate solution found for the iteration), as follows:

$$\tau_{xy} \leftarrow (1 - \rho) \times \tau_{xy} + \rho \times \Delta\tau_{xy}$$

where

τ_{xy} is the amount of pheromone deposited for a state transition xy,

ρ is the pheromone evaporation coefficient or the delay function.

$\Delta\tau_{xy}$ is the maximizing solution cost for the best solution found so far if the component x,y is used in the globally best known solution, otherwise it is 0.

Ant Colony System Algorithm for computing minimum cost

```
Step1) ProblemSize, Populationsize, m, $\rho$, $\beta$, $\sigma$, q0 will be the inputs and Pbest

Step 2) Compute Pbest ,
 Pbest ← CreateHeuristicSolution(ProblemSize)

Step 3) Compute Pbestcost, Pheromoneinit
 Pbestcost ← Cost(Sh);
 Pheromoneinit ← 1.0/(ProblemSize×Pbestcost);

Step4) Compute Pheromone
 Pheromone ← InitializePheromone(Pheromoneinit);

Step5)As long as ¬StopCondition()is true follow step6 to11

Step6) for i=1 to m do

Step7)Compute Si and Sicost
    Si ←ConstructSolution(Pheromone,ProblemSize,$\beta$, q0);
    Sicost ← Cost(Si)

Step8) Check condition whether Sicost ≤ Pbestcost.
 If yes then follow step 8

Step8) Pbestcost ← Sicost;
        Pbest ← Si;

Step9)Otherwise
LocalUpdateAndDecayPheromone(Pheromone,Si,Sicost,$\sigma$)

Step10) Exit the for loop

Step11)GlobalUpdateAndDecayPheromone(Pheromone,
Pbest ,Pbestcost, $\rho$)

Step12) Exit the while loop.

Step13) Return the value of Pbest
```

The local pheromone (history) coefficient (σ) controls the amount of contribution history plays in a components probability of selection and is commonly set to 0.1.

The heuristic coefficient (β) controls the amount of contribution problem-specific heuristic information plays in a components probability of selection and is commonly between 2 and 5, such as 2.5.

The decay factor (ρ) controls the rate at which historic information is lost and is commonly set to 0.1.

The greediness factor (q0) is commonly set to 0.9.

The total number of ants (m) is commonly set low, such as 10.

Conclusions and future perspectives

Recently there has been growing interest in the application of ant colony algorithms to difficult combinatorial problems. The control and management of modern computer networks, which are increasingly large, dynamic, and heterogeneous, requires the development of novel algorithms and protocols that are *fully distributed, adaptive, robust, scalable*, and can let the network behave as an *autonomous and self organizing system*. These properties are the typical fingerprints of well engineered *Swarm Intelligence Systems*. In this paper we have reviewed the applications of Swarm Intelligence and how it is useful in evaluating the routing algorithm with minimum cost function for Dynamic or Adaptive network system

References

[1] F. Akyildiz, X. Wang, and W. Wang. Wireless mesh networks: a survey. Computer Networks Journal, 47(4):445–487, 2005.

[2] Principles and applications of swarm intelligence for adaptive routing in telecommunications networks by
Frederick Ducatelle, Gianni A. Di Caro, Luca M. Gambardella

[3] 13. R. Braden, D. Clark, and S. Shenker. Integrated services in the internet architecture: an overview. RFC 1633 (Informational),1994.

[4] O. Babaoglu, G. Canright, A. Deutsch, G.A. Di Caro, F. Ducatelle, L.M. Gambardella, N. Ganguly, M. Jelasity, R. Montemanni, A. Montresor, and T. Urnes. Design patterns from biology for distributed computing. ACM Transactions on Autonomous and Adaptive Systems (TAAS), 1(1):26–66, 2006.

[5] R. Bellman. Dynamic Programming. Princeton University Press, Princeton,NJ, 1957.

[6] Ant Colony System: A Cooperative Learning
Approach to the Traveling Salesman Problem
Marco Dorigo, Senior Member, IEEE, and Luca Maria Gambardella, Member, IEEE

[7] S. Baluja and R. Caruana, "Removing the genetics from the standard genetic algorithm," in Proc. ML-95, 12th Int. Conf. Machine Learning. Palo Alto, CA: Morgan Kaufmann, 1995,

[8] B. Basturk and D. Karaboga. An artificial bee colony (ABC) algorithm for numeric function optimization. In IEEE Swarm Intelligence Symposium, 2006.