Study of Optimal Path Finding Techniques

Parveen Sharma, Neha Khurana
Computer Science Engineering
SKIET, Kurukshetra University, Kurukshetra
Corresponding Author : nehastar111@gmail.com

Abstract

Routing is a process of forwarding the data from a known source to the destination. In this process, the data may travel through several intermediate paths, and there exist a need to select the best possible optimal nodes to forward the data. This optimal selection of nodes will enable to achieve a high performance in the network. Large amount of worked has been carried out to find the optimal path in the network routing to improve its efficiency and to remove congestion problems. A good routing algorithm should be able to find an optimal path and it must be simple. It also must have low overhead, and be robust and stable, converging rapidly, and must remain flexible. There exists a lot of routing algorithm which have been developed for specific kind of network as well as for general routing purpose. In this paper, I studied some of the best optimal path techniques.

Keywords: *Ant Colony Optimization, PSO, Dijkstra's Algorithm, Genetic Algorithm, Hill climbing,Tabu Search, Floyd-Warshall's, Simulated Annealing, Greedy Algorithm*

1. Introduction

Routing is a process of finding paths between nodes in network. When one sends or receives data over the internet, the information is divided into small chunks called packets or datagrams. The data packets are sent along links between routers on internet. The weights of links in network are assigned by the network operator. The lower the weight, the greater the chance that traffic will get routed on that link. Routing process uses routing table at each node to store all the nodes which are at one hop distance from it. It also stores the other nodes along with the number of

hops to reach that node, followed by the neighbor node through which it can be reached. Router decides which neighbor to choose from routing table to reach specific destination.

2. Shortest Path Problem:

The shortest path problem is defined as that of finding a minimum-length (cost) path between a given pair of nodes. The Dijkstra algorithm is considered as the most efficient method for shortest path computation in IP networks. But when the network is very big, then it becomes inefficient since a lot of computations need to be repeated[7].

3. Techniques:

There are many optimization techniques for finding optimal path and these are defined as below:

3.1 Ant Colony Optimization(ACO):

Ant colony optimization technique is used to find the shortest path finding algorithm in spite of GPS(global position satellite) or any other method. ACO is a class of optimization algorithms modeled on the actions of an ant colony. Ant Colony Optimization (ACO) studies artificial systems that take inspiration from the behavior of real ant colonies and which are used to solve discrete optimization problems**[2]**.Our arena which is randomly created has white pixels showing clear area and black one for restricted entry. Different steps of a simple ant colony system algorithm are as follows:

3.1.1 Initialization:

To create a arena by creating image of certain pixels and introducing noise randomly acting as hurdles for the path. This noise is known as salt and pepper noise, it will be introduced in R, G, and B format of image.

3.1.2 Random Points:

Taking two random points in image to define the path and making straight line between them using loop. Straight line defines the possible hurdles coming in way.

3.1.3 Routing:

Defines the possible routes to reach the destination.

3.1.4 Imfilling:

Each object will be taken separately in a dummy image and will be dilated and eroded to fill the holes .It will be repeated for every object.

3.1.5 Finding Nearest Points:

Nearest points are taken by forming the matrices defining rows and column. Then loop starts for finding the shortest path through the hurdles and finally nearest points are attached to their startup and end point.

ACO can be used to find the solutions of difficult combinatorial optimization problems and it enjoys a rapidly growing popularity. Although ACO has a powerful capacity to find out solutions to combinatorial optimization problems, it has the problems of stagnation and premature convergence and the convergence speed of ACO is always slow. Those problems will be more obvious when the problem size increases. Therefore, several extensions and improvements versions of the original ACO algorithm were required[10].

3.2 Particles Swarm Optimization (PSO):

PSO is an optimization algorithm which has been applied to finding shortest path in the network. However, it might fall into local optimal solution. In this algorithm, the flow starts with a population of particles whose position that represents the solutions for the problem, and velocities are randomly initialized in the search space. The search for optimal position is performed by updating the particle velocities, hence positions, in each iteration

in a specific manner as follows: in every iteration, the fitness of each particle's position is determined by fitness measure and the velocity of each particle is updated by keeping track of two "best" positions**[3]**.

3.2.1 Pbest:

The first one is the best position a particle has traversed so far, this value is called "pbest".

3.2.2 Nbest:

Another best value is the best position the any neighbor of a particle has traversed so far, this best value is a group best and is called "nbest".

3.2.3 Gbest:

When a particle takes the whole population as its neighborhood, the neighborhood best becomes the global best and it is accordingly called "gbest".

In the PSO algorithm, the potential solutions, called as particles, are obtained by "flowing" through the problem space by following the current optimum particles. Generally speaking, the PSO algorithm has a strong ability to find the most optimistic result, but it has a disadvantage of easily getting into a local optimum. The PSO algorithm's search is based on the orientation by tracing *Pb* that is each particle's best position in its history, and tracing Pg that is all particles' best position in their history; therefore, it can rapidly arrive around the global optimum. However, because the PSO algorithm has several parameters to be adjusted by empirical approach, if these parameters are not appropriately set, the search will become very slow near the global optimum.

3.3 Tabu Search:

It is an iterative search that starts from some initial feasible solution and attempts to determine the best solution in the manner of a hill-climbing algorithm. The algorithm keeps historical local optima for leading to the near global optimum fast and efficiently. During these search procedures the best solution is always updated and stored aside until the stopping criterion is satisfied. The two main components of the tabu search algorithm are the tabu list restrictions and the aspiration criterion**[4]**. TS uses short-term and/or long-term memory while making moves between neighboring solutions. It is essential for a local search to be balanced in terms of quality of solutions and computing time of these solutions. In that sense, a local search does not necessarily evaluate all neighborhood solutions. Generally, a subset of solutions is evaluated.

If the optimal score is unknown (which is usually the case), it must be told when to stop looking (for example based on time spend, user input, ...).

3.4 Dijkstra's Algorithm:

Dijkastra's algorithm is a graph search algorithm that solves the single-source optimal path problem for a graph with nonnegative edge path costs, producing an optimal shortest path tree. This algorithm is often used in routing and as subroutine in other graph algorithms. It can also be used for finding costs of shortest paths from a single vertex to a single destination vertex by stopping the algorithm once the optimal path to the destination vertex has been determined. Traffic information systems use Dijkstra's algorithm in order to track the source and destinations from a given particular source and destination. The computation is based on Dijkstra's algorithm

which is used to calculate the shortest path tree inside each area of the network**[5]**. Dijkstra's labelling method is a central procedure in most shortest path algorithms. An out-tree is a tree originating from the source node to other nodes.The output of the labelling method is an out-tree from a source node s, to a set of nodes L.Three pieces of information are required for each node i in the labelling method while constructing the shortest path tree:

- the distance label, d(i),
- the parent-node/predecessor p(i),
- the set of permanently labelled nodes L

Where d(i) stores an upper bound on the optimal shortest path distance from s to i; p(i) records the node that immediately precedes node i in the out-tree. By iteratively adding a temporarily labeled node with the smallest distance label d(i) to the set of permanently labelled nodes L, Dijkastra guarentees optimality. The algorithm can be terminated when the destination node is permanently labeled**[6]**.

The major disadvantage of the algorithm is the fact that it does a blind search there by consuming a lot of time waste of necessary resources.Another disadvantage is that it cannot handle negative edges. This leads to acyclic graphs and most often cannot obtain the right shortest path.

3.5 Floyd-Warshall's Algorithm:

This algorithm was developed independently from each other by Floyd(1962) and Warshall (1962). Instead of computing a path from a given start node to all other nodes (or a single destination node), all shortest paths, i.e., from each node to all others, are computed within a single loop. As a result we obtain a matrix Dist, where Dist[i; j] denotes the distance from node i to node j. Furthermore a matrix Next can be computed where Next[i; j] represents the successor of node i on the shortest path from node i to node j (see Algo. below)

for all nodes i of N **do**
for all nodes j of N **do**
if there is an edge from i to j **then** Dist[i; j] := d(i; j) **else** Dist[i; j] := ∞
for all nodes i of N **do**
for all nodes j of N **do**
for all nodes k of N **do**
if Dist[j;i] +Dist[i;k] < Dist[j;k] **then** Dist[j;k] := Dist[j;i] +Dist[i;k]; Next[j;k] := i;

Algorithm1: Floyd-Warshall's Algorithm

Floyd-Warshall's algorithm has a time complexity of $O(n^3)$, which is equivalent to performing Dijkstra's algorithm n times.In analogy to techniques to improve the time complexity of Dijkstra's algorithm, Johnson's algorithm can be regarded as a superior approach to find the shortest paths between all pairs of nodes in sparse directed graph. It allows some of the edge weights to be negative, but no negative-weight cycles must exist. It works by using the

Bellmann-Ford algorithm to compute a transformation of the input graph that removes all negative weights, allowing Dijkstra's algorithm to be used on the transformed graph.

The Bellman-Ford algorithm allows us to test for the presence of negative cycles simply by continuing for one additional iteration, the Floyd-Warshall algorithm provides no such mechanism**[9]**.

3.6 Simulated Annealing:

Simulated Annealing (SA) is a generic probabilistic metaheuristic for the global optimization problem of applied mathematics, namely locating a good approximation to the global minimum of a given function in a large search space. It is often used when the search space is discrete. Annealing in metallurgy, a technique involving heating and controlled cooling of a material to increase the size of its crystals and reduce their defects. The heat causes the atoms to become unstuck from their initial positions (a local minimum of the internal energy) and wander randomly through states of higher energy; the slow cooling gives them more chances of finding configurations with lower internal energy than the initial one. SA may be more effective and in the process each step of the SA algorithm replaces the current solution by a random "nearby" solution, chosen with a probability that depends on the difference between the corresponding function values and on a global parameter T (called the temperature), that is gradually decreased during the process. The dependency is such that the current solution changes almost randomly when T is large, but increasingly "downhill" as T goes to zero. The allowance for "uphill" moves saves the method from becoming stuck at local minima.

For problems where the energy landscape is smooth, or there are few local minima, SA is overkill --- simpler, faster methods (e.g., gradient descent) will work better. But generally don't know what the energy landscape is for a particular problem. The method cannot tell whether it has found an optimal solution. Some other complimentary method (e.g. branch and bound) is required to do this.

3.7 Greedy Algorithm:

A *greedy algorithm* is a method for finding an optimal solution to some problem involving a large, homogeneous data structure (such as an array, a tree, or a graph) by starting from an optimal solution to some component or small part of the data structure and extending it, by considering additional components of the data structure one by one, to an optimal global solution. A greedy algorithm assumes that a local optimimum is part of the global optimimum. A Greedy Algorithm is any algorithm that follows the problem solving metaheuristic of making the locally optimal choice at each stage for finding the global optimum. Greedy algorithms produce good solutions on some mathematical problems, but not on others. In general, greedy algorithms have some steps:

1. A candidate set, from which a solution is created
2. A selection function, which chooses the best candidate to be added to the solution
3. A feasibility function, that is used to determine if a candidate can be used to contribute to a solution
4. An objective function, which assigns a value to a sol ution, or a partial solution, and

5. A solution function, which will indicate when it has discovered a complete solution

Their principal disadvantage is that for many problems there is no greedy algorithm. More precisely, in many cases there is no guarantee that making locally optimal improvements in a locally optimal solution yields the optimal global solution.

3.8 Hill Climbing:

Hill climbing is a mathematical optimization technique which belongs to the family of local search. It is relatively simple to implement, making it a popular first choice. Although more advanced algorithms may give better results, in some situations hill climbing works just as well. Hill climbing can be used to solve problems that have many solutions, some of which are better than others. It starts with a random (potentially poor) solution, and iteratively makes small changes to the solution, each time improving it a little. When the algorithm cannot see any improvement anymore, it terminates. Ideally, at that point the current solution is close to optimal, but it is not guaranteed that hill climbing will ever come close to the optimal solution. Hill climbing is used widely in artificial intelligence.

Hill climbing is good for finding a local optimum (a solution that cannot be improved by considering a neighbouring configuration) but it is not guaranteed to find the best possible solution (the global optimum) out of all possible solutions (the search space). The characteristic that only local optima are guaranteed can be cured by using restarts (repeated local search), or more complex schemes based on iterations, like iterated local search, on memory, like reactive search optimization and tabu search, on memory-less stochastic modifications, like simulated annealing.

3.9 Genetic Algorithm:

Genetic algorithm also uses adaptive heuristic search technique which finds the set of best solution from the population.GA works on the search space called population. Each element in the population is called as chromosome. Each chromosome is evaluated for fitness and this fitness defines the quality of solution[**7]**. The evolution from one generation to the next one involves mainly three steps: fitness evaluation, selection and reproduction. First, the current population is evaluated using the fitness evolution function and then ranked based on their fitness. A new generation is created with the goal of improving the fitness. GA uses three operators: reproduction, crossover and mutation. First selective reproduction is applied to the current population so that the string makes a number of copies proportional to their own fitness. This results in an intermediate population. Second, GA select "parents" from the current population with a bias that better chromosome are likely to be selected. This is to be done by the fitness value or ranking of a chromosome. Third, GA reproduces "children" (new strings) from selected parents using crossover and/or mutation operators.Crossover is basically consists in a random exchange of bits between two strings of the intermediate population. Finally, the mutation operator alters randomly some bits of the new strings. This algorithm terminates when an acceptable solution is found**[8]**.

The drawback of GA is that optimisation problems (they are called variant problems) cannot be solved by means of genetic algorithms. This occurs due to poorly known fitness functions which generate bad chromosome blocks in spite of the fact that only good chromosome blocks cross-over. Genetic algorithm

applications in controls which are performed in real time are limited because of random solutions and convergence, in other words this means that the entire population is improving, but this could not be said for an individual within this population. Therefore, it is unreasonable to use genetic algorithms for on-line controls in real systems without testing them first on a simulation model.

4. Conclusion:

Because of the complexities existing methods are not computationally feasible for deserving large scale network. As a result, standard, traditional, optimization techniques are often not able to solve these problems of increased complexity with justifiable effort in an acceptable time period. To overcome these problems, and to develop systems that solve these complex problems,

New modified GA algorithm are required. Using these nature inspired search methods it is possible to overcome some limitations of traditional optimization methods, and to increase the number of solvable problems.

5. References:

[1] Kavitha Sooda , T. R. Gopalakrishnan Nair "A Comparative Analysis for Determining the Optimal Path using PSO and GA", International Journal of Computer Applications (0975 – 8887) Volume 32– No.4, October 2011

[2] Dilpreet kaur, Dr.P.S Mundra, "Ant Colony Optimization: A Technique Used For Finding Shortest Path" , International Journal of Engineering and Innovative Technology (IJEIT), Volume 1, Issue 5, May 2012

[3] Hongyan Cui, Jian Li, Xiang Liu, Yunlong Cai, "Particle Swarm Optimization for Multi-constrained Routing in Telecommunication Networks", I.J.Computer Network and Information Security, 2011, 4, 10-17 Published Online June 2011 in MECS

[4] Anant Oonsivilai, Wichai Srisuruk, Boonruang Marungsri, Thanatchai Kulworawanichpong;"Tabu Search Approach to Solve Routing Issues in Communication Networks" , World Academy of Science, Engineering and Technology 29 2009

[5] Neha Choubey, Mr. Bhupesh Kr. Gupta; "Analysis of Working of Dijkstra and A* to Obtain Optimal Path", International Journal of Computer Science and Management Research, Vol 2 Issue 3, March 2013 ISSN 2278-733X

[6] Faramroze Engineer," Fast Shortest Path Algorithms for Large Road Networks" Department of Engineering Science, University of Auckland ,New Zealand

[7] Dr.Rakesh Kumar, Mahesh Kumar; "Exploring Genetic Algorithm for Shortest Path Optimization in Data Networks", GlobalJournal of Computer Science and Technology, Page 8 Vol. 10 Issue 11 (Ver. 1.0) October 2010

[8] Anjum A. Mohammed, Gihan Nagib;" Optimal Routing In Ad-Hoc Network Using Genetic Algorithm" Int. J. Advanced Networking and Applications, Volume: 03, Issue: 05, Pages: 1323-1328 (2012)

[9] Kai Gutenschwager, Axel Radtke, Sven Volker, Georg Zeller;" The Shortest Path: Comparison Of Different Approaches And Implementations For The Automatic Routing Of Vehicles", Proceedings of the 2012 Winter Simulation Conference C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A. M. Uhrmacher, eds.

[10] Zar Chi Su Su Hlaing and May Aye Khine, Member, IACSIT "Solving Traveling Salesman Problem by Using Improved Ant Colony Optimization Algorithm", International Journal of Information and Education Technology, Vol. 1, No. 5, December 2011