Solving Shortest Capacitated Path Problem Using a Bi-Objective Heuristic Approach (Invited Paper)

Crina Grosan, Ajith Abraham
Center of Excellence for Quantifiable Quality of Service,
Norwegian University of Science and Technology,
Trondheim, Norway
{crina, ajith}@q2s.ntnu.no

Abstract

The shortest capacitated path problem is a well known problem in the networking area, having a wide range of applications. In the shortest capacitated path problem, a traffic flow occurs from a source node to a destination node in a certain direction subject to a cost constraint. In this paper, a new approach for dealing with this problem is proposed. The proposed algorithm uses a special way to build valid solutions and an improvement technique to adjust the path. Some numerical experiments are performed using randomly generated networks having 25 - 200 nodes. Empirical results are compared with the results obtained using Genetic Algorithms which is an established technique for solving networking problems.

1. Introduction

In this paper, we deal with the shortest capacitated path problem in which a flow must be sent from a source node to a destination node in a certain direction subject to cost constraints.
Two main objectives are formulated:

- cost of the path must be minimized
- the number of common edges whose capacity is less then the required one must be minimized (in fact, it must be 0).

The network studied is a capacitated network (which means capacities are limited) and there is a cost on each edge per unit of bandwidth. As reported in the literature, huge amount of work has been done in this area. Several different approaches have been proposed for this problem: Tabu Search has been used in [5], Multiobjective Genetic Algorithms have been applied in [7], Genetic algorithms and other machine learning techniques have been extensively applied [1][2][4][8][9][10], Scaling algorithms were used in [3] and [6].
In this paper, we propose a model which works in two phases: first, a valid solution is built (valid refers here to a path between source and destination). After this, the solution is improved in a number of consecutive iterations so that all the requirements are satisfied and the path obtained is the shortest one. The proposed algorithm is validated using several networks having 25, 50, 100, 150 and 200 nodes. The proposed approach is presented in Section 2 and details about experiments are given in Section 3, followed by Conclusions in the last section.

2. Proposed Approach

We consider the following problem: a undirected graph G= (V, E), a cost function cost: $E \rightarrow R^+$ and a capacity function capacity: $E \rightarrow R^+$ are given. We are given some requirements between one pair of vertices (source and destination) $(s, d) \in V$. The goal is to find a minimal (cost wise) path between these nodes as well as to assure that the capacity of the edges is not overloaded.

The problem involves optimization (minimization) of two objectives:

- minimize the total cost of the path between s and d;
- assure that the capacity of each edge of the obtained path is not overloaded. This objective can be formulated as an objective in which the number of edges (from the built path) for each the capacity is overloaded is to be minimized. In other words, for this objective, we accept only the value 0 (otherwise that edge cannot be considered).

Solution representation and design

The following representation is used to solve the problem: a solution represents the required path to be designed. This means, if we have the source node *s* and the destination node *d*, we have to find a path between *s* and *d* of the lowest cost.
In order to better emphasize the way in which a solution is built, let us consider the network represented in Figure 1 where we have one source node (node 1) and one destination node (node 7).

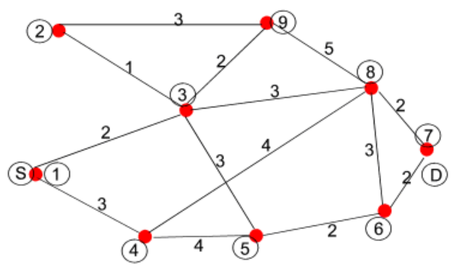
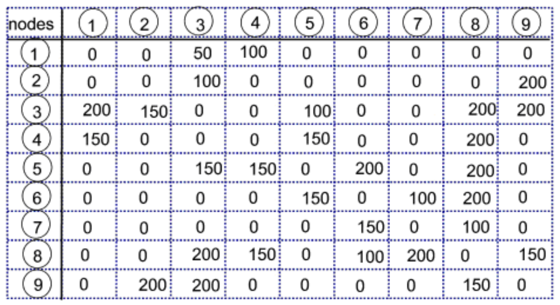


Figure 1. A network with one source (node 1) and one destination (node 7) and the associated cost.

Consider also the capacity matrix as given in Figure 2. The value 0 for the cost between two nodes (*i*, *j*) has the meaning that there is no connection between these two nodes.



nodes	1	2	3	4	5	6	7	8	9
1	0	0	50	100	0	0	0	0	0
2	0	0	100	0	0	0	0	0	200
3	200	150	0	0	100	0	0	200	200
4	150	0	0	0	150	0	0	200	0
5	0	0	150	150	0	200	0	200	0
6	0	0	0	0	150	0	100	200	0
7	0	0	0	0	0	150	0	100	0
8	0	0	200	150	0	100	200	0	150
9	0	200	200	0	0	0	0	150	0

Figure 2. Capacity matrix for the network depicted in Figure 1.

The solution is built in the following way:

Step 1. Start with the node 1 (source node) as current node.

Step 2. If there is a direct path (link) from the current node to the destination node (node 7) then move to node 7. Otherwise randomly chose another node from the network which is connected to the current node and set it as new current node.

Step 3. If the current node is the desired destination node then stop. Otherwise go to step 2.

The following constraints are to be taken into account:

- each node can be used at most one time in a path (for avoiding cycles);
- if a node is reached from where we cannot move in another node (stuck there!) because all the connected nodes were previously used in the path (for instance the situation (1 3 9 2) in the network from Figure 1) we will abort that solution.

The solution for this particular network is built in the following way:

Start from node 1. There is no path between node 1 and node 7. Then randomly chose a node from the set of nodes connected to the node 1 (which is {3, 4}). Suppose that node 3 is chosen, then the path at this point will be (1 3). There is no direct path between the nodes 3 and 7. Then, we can choose from the set of nodes with whom the node 3 is connected ({1, 2, 5, 8, 9}). Suppose we are choosing the node 8. Our path at this point will be (1 3 8). There is a direct path between the nodes 8 and 7. Therefore we can stop here and we can return the path (1 3 8 7).

First objective takes into account the cost of this path and the second objective takes into account the number of edges used by this path which cannot support the required amount of data. For instance, if the size of data which has to be transferred for source to destination is 50, then the solution built above is a valid one. But if the size is 150, then the solution is not a correct one.

Solution improvement

In order to improve a solution designed using the above procedure, we will perform some modifications on the existing path as follows:

- One node on this path is also randomly generated.
- From the selected node, the path is re-built by considering another path to reach the destination different from the current one.

For example, let us consider the network depicted in Figure 1 and the solution we designed above (the primary path): (1 3 8 7). Suppose that the node randomly generated is node 3. We will now build another path from node 3 to node 7 different from the existing one (which is (3 8 7)) but using the same procedure which we use for initializing the paths (and explained above). So, the new obtained path can be (1 3 5 6 7).

3. Experiment Results

We performed several experiments considering networks having different sizes between 25 and 200 nodes. The test data were randomly generated. We compared the results obtained by the proposed approach with the results obtained by Genetic Algorithms. Genetic Algorithms have been widely applied for solving these types of problems. We use a binary encoding where $x_{i,j} = 1$ indicates that the result path includes the arc (*i*, *j*), and $x_{i,j} = 0$ otherwise.

No of nodes	No of connections	Population size	No of generations	Proposed approach solution (cost)	GA solution (cost)
25	60	10	20	125	207
50	150	20	20	262	271
100	300	20	20	194	361
150	400	20	30	178	217
200	600	30	50	283	320

Table 1. Performance comparison for different number of nodes using the two approaches

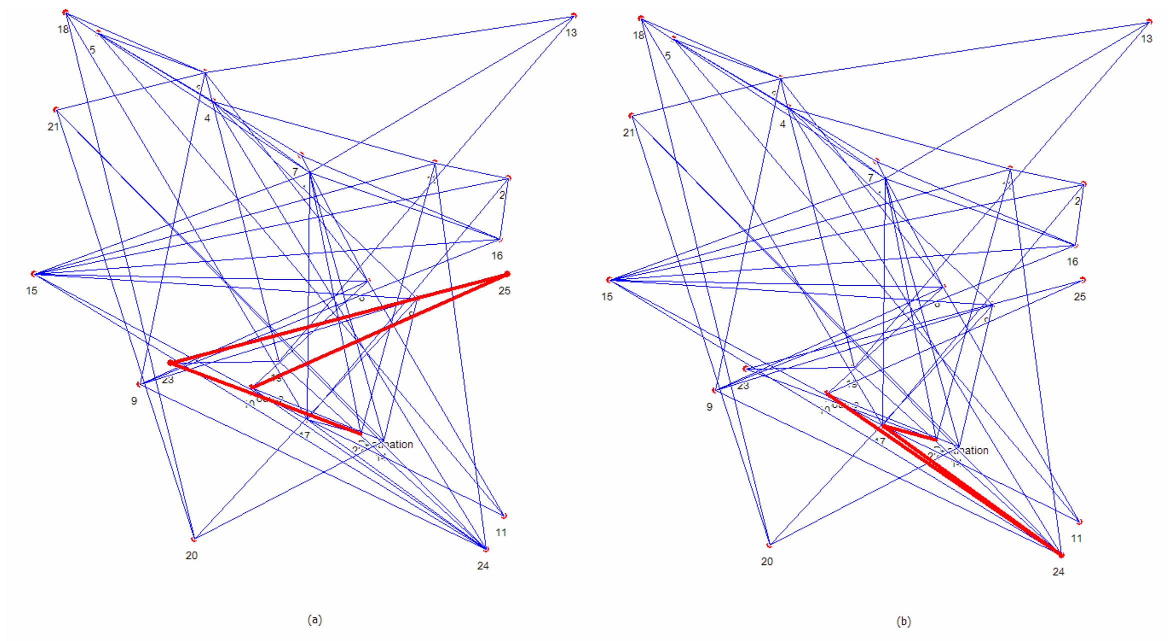


Figure 3. The paths obtained by the proposed approach (left) and GA (right) for the network having 25 nodes and 60 connections.

Genetic operators for binary encoding (such as mutation and single point crossover) and binary tournament selection are used [2].
Parameters used by each algorithm, problems details and results are presented in Table 1. Population size and number of generations used by the Genetic Algorithms are same like the number of initial solutions built and number of iterations for the proposed approach.

In Figures 3-7, the paths obtained by the proposed approach and GA for all the considered networks are depicted. As evident from the results presented in Table 1, and from the Figures 3-7, the proposed approach gives the best results for all the considered network problems while compared to GA.

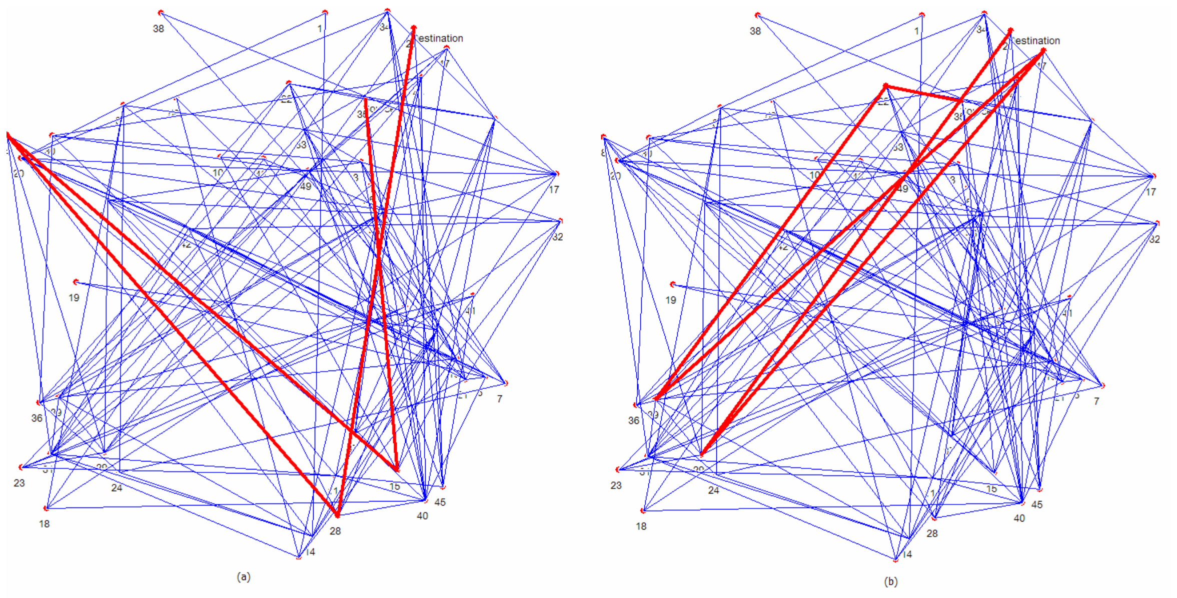


Figure 4. The paths obtained by the proposed approach (left) and GA (right) for the network having 50 nodes and 150 connections.

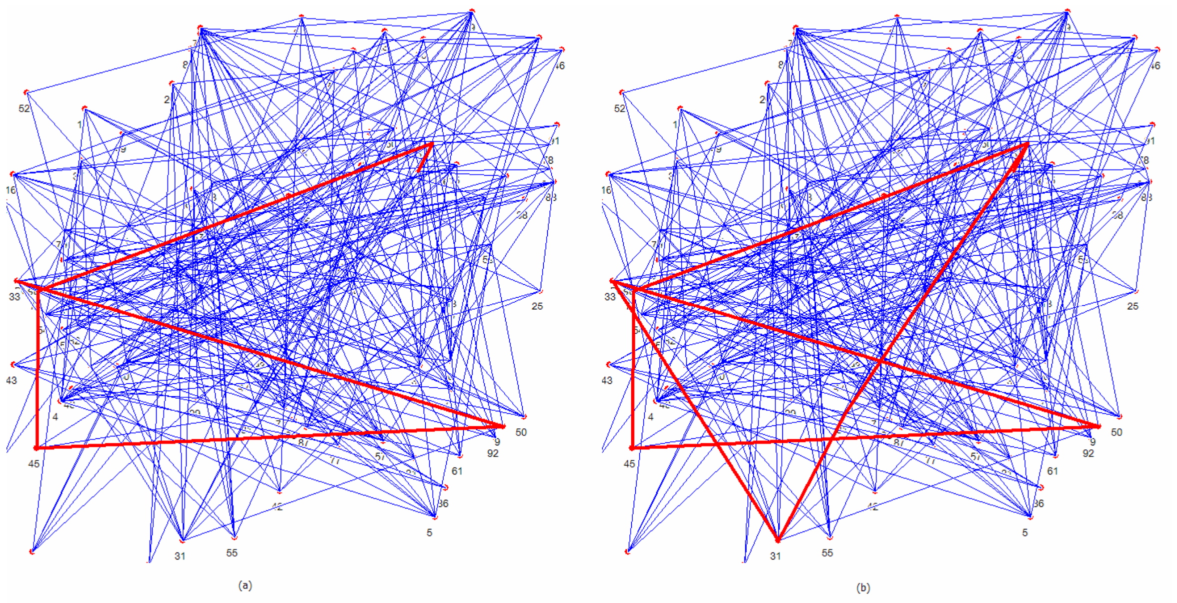
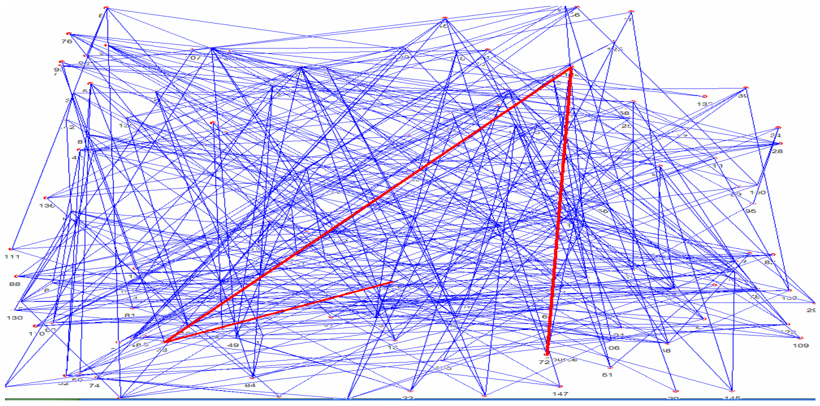


Figure 5. The paths obtained by the proposed approach (left) and GA (right) for the network having 100 nodes and 300 connections.


(a)

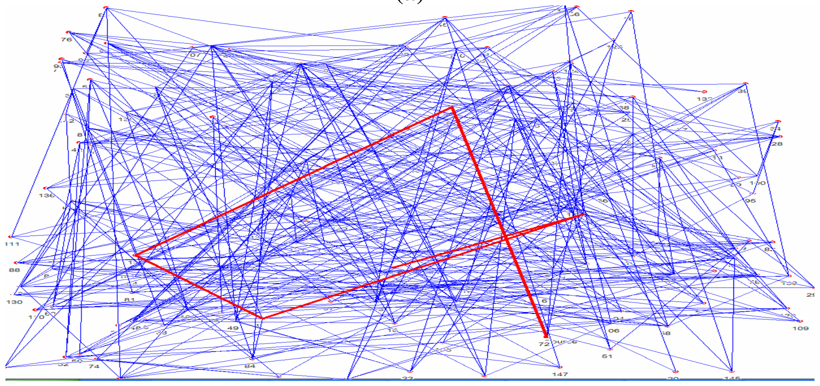
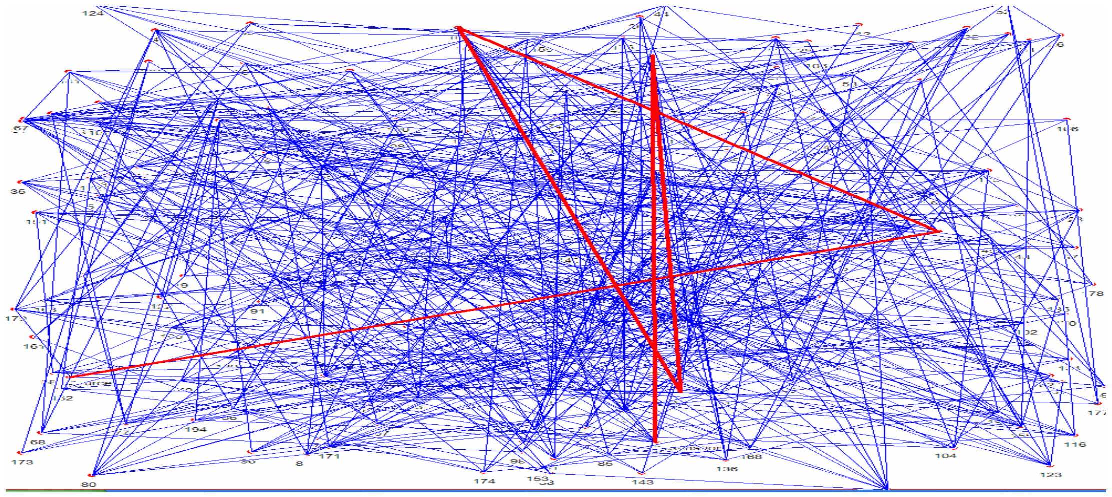
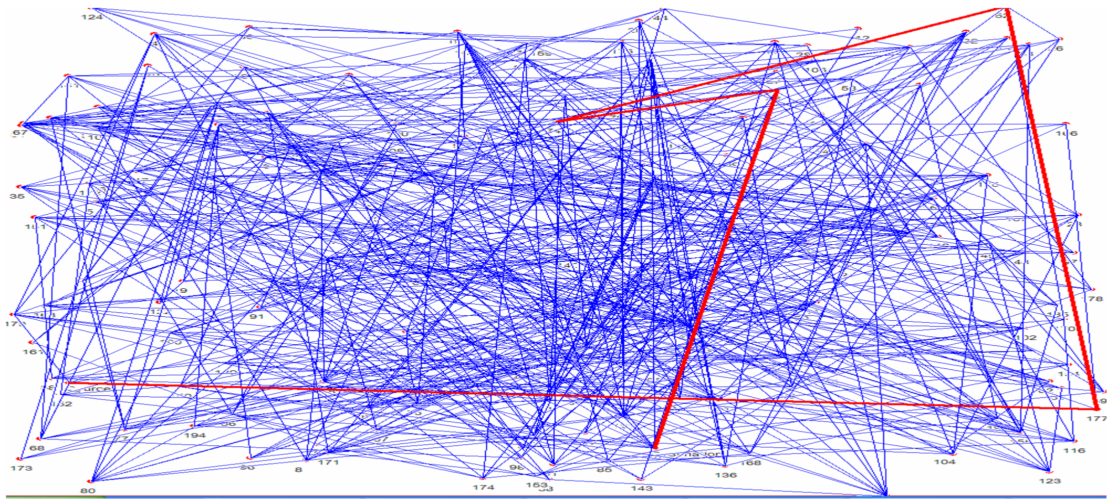

(b)

Figure 6. The path obtained (a) by the proposed approach and (b) GA for the network having 150 nodes and 400 connections.


(a)



(b)

Figure 7. The path obtained (a) by the proposed approach (b) and GA for the network having 200 nodes and 600 connections.

4. Conclusions

In this paper, a new technique for solving the shortest capacitated path problem is proposed. This approach starts with a set of solutions which are built in a special way so that no invalid solutions are created. These solutions are thereafter improved by using a simple mechanism which re-builds certain parts of the existing path. Numerical experiment and comparisons with genetic algorithms show the efficiency of this new proposed technique and its suitability for large networks.

References

[1] Ahn, C.W., Ramakrishna, R.S., Kang, C.G., Choi, I.C., Shortest path routing algorithm using hopfield neural network, Electron. Lett., 37(19), pp. 1176–1178, 2001.

[2] Ahn, C.W., Ramakrishna, R.S., A Genetic Algorithm for Shortest Path Routing Problem and the Sizing of Populations, IEEE Transactions on Evolutionary Computation, 6(6), pp. 566-579, 2002.

[3] Ahuja, R.K., Orlin, J.B., A capacity scaling algorithm for the constraint maximum flow problem, Networks, 25, pp. 89-98, 1995.

[4] Ali, M.K., Kamoun, F., Neural networks for shortest path computation and routing in computer networks, IEEE Trans. Neural Networks, 4, pp. 941–954, 1993.

[5] Anderson, C.A., Fraughnaugh, K., Parker, M., Ryan J., Path assignment for call routing: An application of tabu search, Annals of Operations Research, 41, pp. 301-312, 1993.

[6] Caliscan, C., A double scaling algorithm for the constraint maximum flow problem, Computers and Operations Research, 2006.

[7] Lo, C.C., Chang, W.H., A Multiobjective Hybrid Genetic Algorithm for the Capacitated Multipoint Network Design Problem, IEEE Transactions on Systems, Man and Cybernetics, Part B, 30(3), 461-470, 2000.

[8] Inagaki, J., Haseyama, M., Kitajima, H. A genetic algorithm for determining multiple routes and its applications, In *Proceedings of IEEE International Symposium on Circuits and Systems*, pp. 137–140, 1999.

[9] Leung, Y., Li, G., Xu, Z.B. A genetic algorithm for the multiple destination routing problems, *IEEE Transactions on Evolutionary Computation*, 2, pp. 150–161, 1998.

[10] Munemoto, M., Takai, Y., Sato, Y., A migration scheme for the genetic adaptive routing algorithm, in Proceedings of IEEE International Conference on Systems, Man, and Cybernetics, pp. 2774–2779, 1998.