A Hybrid Metaheuristic for Routing in Road Networks

Omar Dib, Marie-Ange Manier, Alexandre Caminada

A Hybrid Metaheuristic for Routing in Road Networks

Omar DIB
Technical Research Institute
SystemX
92120 Palaiseau France
Omar.dib@irt-systemx.fr

Marie-Ange MANIER
OPERA–UTBM
90010 Belfort Cedex
Marie-ange.manier@utbm.fr

Alexandre CAMINADA
OPERA–UTBM
90010 Belfort France
alexandre.caminada@utbm.fr

***Abstract*— computing the optimal route to go from one place to another is a highly important issue in road networks. The problem consists of finding the path that minimizes a metric such as distance, time, cost etc. to go from one node to another in a directed or undirected graph. Although standard algorithms and techniques such as Dijkstra and integer programming are capable of computing shortest paths in polynomial times, they become very slow when the network becomes very large. Furthermore, traditional methods are incapable of meeting additional constraints that may arise during routing in transportation systems such as computing multi-objective routes, routing in stochastic networks. Therefore, we have thought about using meta-heuristics to solve the routing issue in road networks. Meta-heuristics are capable of copying with additional constraints and providing optimal or near optimal routes within reasonable computational times in large-scale road networks. The proposed approach is a combination between genetic algorithm (GA) and variable neighborhood search (VNS). To evaluate our method, we made experimentations using random generated and real road network instances. We compare our approximate method with two exact algorithms (Dijkstra and integer programming). Results show that our approach is able to give high quality solutions within milliseconds even in large-scale networks. Moreover, the selected meta-heuristics show high flexibility rate in terms of meeting other problem requirements.**

Keywords—Routing; Shortest path; Road networks; Metaheuristics; Dijkstra's algorithm, Integer Programming

I. INTRODUCTION

Finding the best route to reach a destination in road networks is not always straightforward. Travelers usually encounter difficulties when planning their trips in elaborate road schemes. Having several possibilities to go from one place to another usually makes travelers incapable of figuring the best route out. Consequently, to help travelers to find the best routes through the complex networks, route planning has gained significant importance in recent years. Several commercial navigation products have been developed to provide travelers with driving directions helping them to reach their destinations such as .

Finding the optimal path between two places in road networks refers to solving the one-to-one shortest path problem (SPP) in a directed connected graph. Graph's nodes represent road junctions and edges connecting two nodes account for road segments.

The SPP is among the most studied network-flow optimization problems. Several algorithms have been developed to compute shortest paths since 1956. Dijkstra's algorithm is the standard solution to solve the one-to-one shortest path problem. Bellman-Ford algorithmis a shortest path algorithm used in graphs containing negative edges' weights. Floyd-Warshall algorithm computes shortest paths between all pairs of points in a weighted graph

The SPP was also formulated and solved using optimization techniques, especially the integer programming (IP). Although, IP can solve shortest paths and are flexible to meet additional problem constraints, they usually require high computational time.

Computing best routes in road networks can be done using Dijkstra's algorithm or even the IP techniques. However, this would be far too slow for users seeking answers in milliseconds, especially in large-scale networks. For instance, Dijkstra would take up to 10 seconds to solve a shortest path query in a continental road network. Therefore, extensive research have been done in order to accelerate traditional algorithms.

For example, the running time of Dijkstra can be enhanced using bidirectional search. Other techniques have been developed to accelerate routing in road such as A* search, Arc Flags and ALT (A*, landmarks, and triangle inequality).

Although, those techniques are efficient and fast enough to provide driving directions even in large-scale road networks, they become less performant or even inapplicable when additional constraints are added to the SPP such as dynamic arcs weights, multimodal shortest paths and multi-criteria paths optimization.

To overcome such shortcomings, we believe that meta-heuristics such as genetic algorithms, local search procedures are efficient candidates to find optimal or near optimal solutions within an acceptable computation time.

Meta-heuristics such as such as Genetic Algorithms (GAs), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO) have been applied for solving routing issues in various fields of applications. For instance, reference [1] proposed a genetic algorithm to find the shortest path in computer networks. Moreover, reference [2] used a genetic algorithm to solve the routing issue in road networks. Reference [3] also worked with genetic algorithm to find the shortest path in data

networks. Reference [4] combined PSO with local search and velocity re-initialization for computing shortest paths in computer networks.

Although the aforementioned works are very interesting, we have remarked that experimental results have been only done over small network instances (ex: 100 nodes). Therefore, we have no guarantee that such approaches are also efficient on large-scale networks. Furthermore, none of the abovementioned methods has been applied to solve other complex variants of SPP such as computing multi-objective shortest paths in stochastic netowrks.

In this paper, we propose a new approach for solving the routing issue in road networks. We introduce a new combination process in which we couple variable neighborhood search with genetic algorithm.

The remainder of this paper is organized as follows. In next section, we describe more formally the SPP. we present our novel approach in section 3. Section 4 is devoted to present experimental results. Finally, section 5 concludes this paper as well as proposes some future works.

II. PROBLEM DEFINITION

The one-to-one SPP, whether it is addressed in transportation, computer networks or other fields, is defined as that of finding the path with the minimum cost (time, distance...) path between a given source and destination. More formally, let *G(V,E)* be a directed graph with a node set *V* of size *n* and an edge set *E* of size *m*. Let us also assume that *W* is a weight function that assigns each edge *e (uv)* with a non-negative weight *wij* . The problem is then to find a path *p (e1, e2… ek)* between a given source node *'s'* and a destination *'t'* in such a way the sum of the edges' weights in *p* is minimized.

III. PROPOSED APPROACH

As aforementioned, the main contribution of this paper is to adapt and apply an approximate method based on a collaboration between two meta-heuristics (GA and VNS), for solving routing issue in road networks.

The proposed approach proceeds with a population of solutions as GAs works. Initial generated solutions are performed using a double search algorithm. The details of this algorithm is described later in this article.

After generating initial solutions, a VNS is applied over each individual in the first population in order to enhance their quality. Having good initial solutions usually enhances the chance of the algorithm in achieving better results. Adapting VNS to the problem is discussed later.

After improving solutions, individuals are passed through an evaluation process. The fitness function is the sum of the weights of each edge including in a path.

After the evaluation phase, genetic operations are repeatedly performed in order to improve current solutions. The algorithm starts with a crossover operator that forms two new solutions from two initial parents. A single-point crossover technique is applied with a probability of 0.9.

In contrast to traditional mutation techniques, a special mutation operation is performed in the proposed approach. That is, the VNS method is applied with a probability of 0.1 over each individual (path) in the population. Thanks to this novel technique, the algorithm will have more chances to exploit and explore new regions of the solution search space.

The whole process is repeated until the algorithm reach our stopping criteria. The following scheme represents the algorithms' steps that we will discuss in more details in next paragraphs.

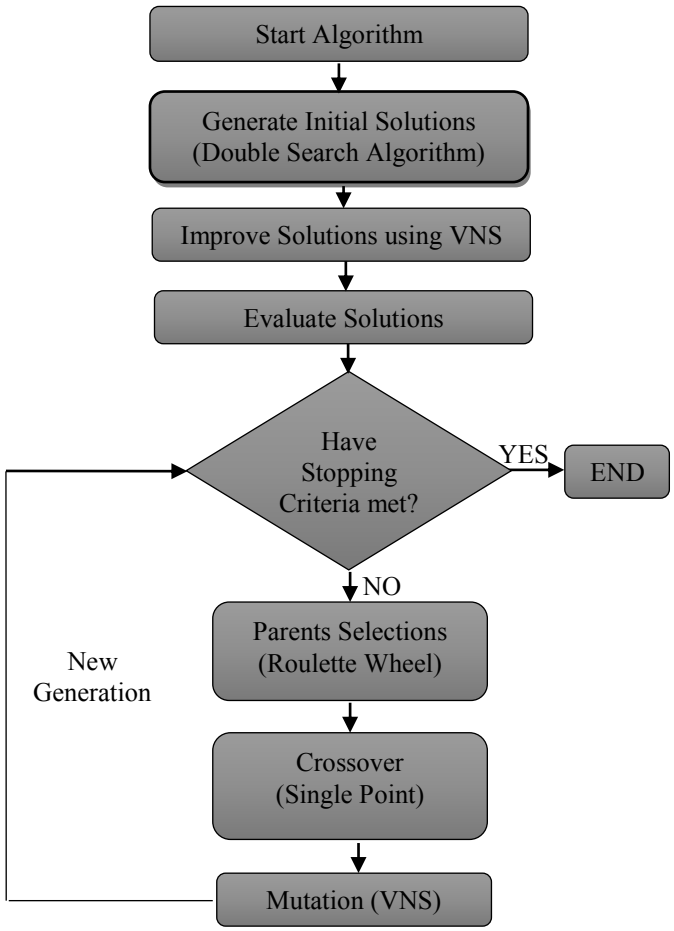

Figure 1: Steps of our algorithm

A. Encoding

Encoding of chromosomes is one of the issues that may arise when implementing a GA. Encoding is very dependent on the problem and it may increase the performance of the algorithm. The permutation encoding technique has been used to represent a chromosome in our approach. Each individual is encoded as a string of numbers, which represents the identifiers of edges in a path. For example, in Figure2, the path to get from A to G is encoded as a vector P = (AB, BD, DE, EG); Each element in P represents the identifier of the correspondent edge.

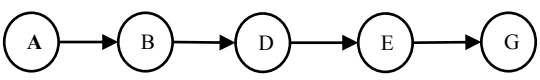

Figure 2: Encoding solutions

B. Generating initial solution

One of the challenges addressed in meta-heuristics is the generation of initial solution (s). Usually, good initial solutions might rapidly guide the search process towards important regions in the search space.

Initial solutions in the proposed algorithm are a set of paths generated using a double search algorithm. The basic idea behind is to simultaneously run a forward search from the origin point (s) and a backward search from the destination point (t). A new path is then found between the origin and the destination when the two searches intersect.

Example: Here is an example to illustrate more the algorithm. Some simplifications have been made to reach better understanding.

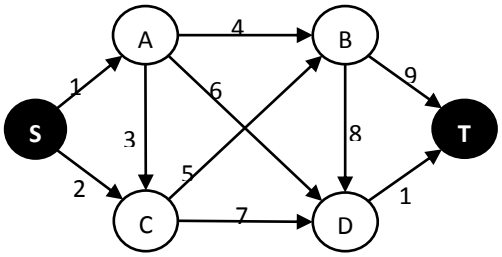

Figure 3: Getting initial feasible paths between S and T

After applying the double search algorithm to get a set of initial feasible paths between the node S and T in G (6, 10) (Figure 3), we will end up with the following paths:

Path1:	**SA**	**AB**	**BT**	
Path2	SA	AD	DT	
Path3	SA	AC	CB	BT
Path4	SC	CD	DT	

C. Enhancing Initial Solutions Using VNS

One of the major challenges addressed in VNS is the construction of the neighborhood structures. To deal with that issue, a preprocessing operation is accomplished during the generation phase of the network. The result is a VNS with two-neighborhood structures.

More specifically, each edge in the graph is examined to check if its starting and ending nodes have a common node in between. Two nodes S and T have a common node if and only if the end point of one adjacent edge of S is the same as the starting point of an incoming edge of T.

Example: By taking the edge (AC) in Fig.4, it can be noticed that the node B is shared between the adjacent edge (AB) of A and the incoming edge (BC) of the node C. Therefore, there is an alternative path to reach the node C from A, which is in this case the path (AB, BC).

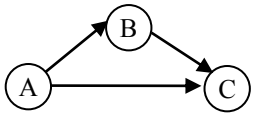

Figure 4. Constructing neighboring structures

Once such common nodes are detected, the process of constructing the two neighboring structures begins.

✓ If the length of the edge (AC) is greater than the length of the path (AB, BC), the edge (AC) can be replaced by the path (AB, BC) during the search process. That case makes our first neighboring structure. That is, the first neighboring structure is a list containing replacement paths formed by two edges for each edge.

✓ A second scenario may arise if the length of the path (AB, BC) is greater than the length of (AC), thus, we can replace the path (AB, BC) by the simple edge (AC) in any path. the second neighboring structure can then be seen as a list containing an edge that will replace a path formed by two edges.

Example: Fig.5 shows a graph with 13 edges and 8 nodes. An example of the replacements included in the first neighboring structure is to substitute the edge (SC) with the 2-edges path (SA, AC). Replacing the 2-edges path (BD, DT) by the edge (B, T) is an example of an instance existing in the second replacement structure.

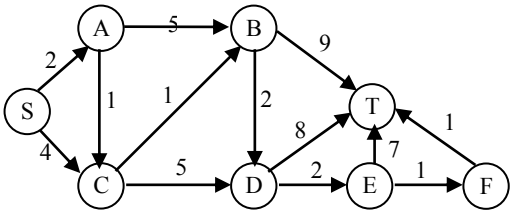

Figure 5. Performing VNS over an individual

After performing the preprocessing operation over Fig.3, the algorithm will end up with two replacement lists that represent the neighboring structures of the VNS.

TABLE 1. Items in the first replacement list

Edge	***Replacements***
(SC)	(SA, AC)
(AB)	(AC, CB)
(CD)	(CB, BD)
(ET)	(EF, FT)

TABLE 2. Items from the second replacement list

Path	***Replacements***
(BD, DT)	(BT)
(DE, ET)	(DT)

After defining neighboring structures, it is worth explaining now how the algorithm uses such information in order to improve the quality of an individual. To do so, let us assume that the path P= (SC, CB, BD, DT) (Fig.5) is an individual representing a solution to go from S to T. The length of P is 15. The algorithm applies the VNS over P as follows:

✓ Initially, the algorithm uses the first neighboring structure to perform local refinements over *P*. It goes through

each edge in *P* and tries to detect if there is a replacement for that edge. In our example, the individual *P* can be improved by substituting the edge (SC) by the path (SA, AC). The new path generated is then (SA, AC, CB, BD, DT) and its length is 14. As can be noticed, the new path does not contain edges included in the first replacement structure so the algorithm switches to the second neighborhood structure.

✓ The algorithm examines then each two successive edges and searches for a replacement. In this example, it can be remarked that the path (BD, DT) including in *P* is in the second replacement structure. Hence, the algorithm replaces it by the edge (BT). The new path generated is (SA, AC, CB, BT) and its length is improved to 13.

The VNS method is based on two-neighboring structures. Each time the algorithm gets trapped in a local minimum, the structure of the neighborhood changes. By following this technique, the algorithm will exploit and explore wide regions of the search space.

Adding more than two structures will possibly enhance the chance of the algorithm to find the best path. However, that might increase the time to accomplish the preprocessing operation.

D. Evaluating an individual

The fitness function is accomplished by taking a path in the population and adding the weights between each node pairs. The result is a non-negative number representing the path length.

E. Crossover

To perform the crossover, two individuals are selected from the population using the roulette wheel selection technique. Fittest individuals usually have more chances to pass their genes to the next generation. After selection, the algorithm tries to find a single common point to be a crossover point. Once this latter is detected, parts from each initial parent are taken to form new individuals (offsprings) to the next generation. By doing this, the algorithm will have the chance to visit new regions in the search space.

Example: Let us assume that G (Figure 6) is a directed graph with 8 edges and 7 nodes. Let us also assume that from an initial generated population, two paths are found between A and T.

The first path is the sequence of the following edges (AB, BD, DE, ET); its length is 13. The second path is (AC, CD, DF, FT) with length 17. As can be noticed from those sequences, they have in common the node D. Over that point, we will perform the crossover. That is, we construct two new paths (offsprings) from two initial individuals.

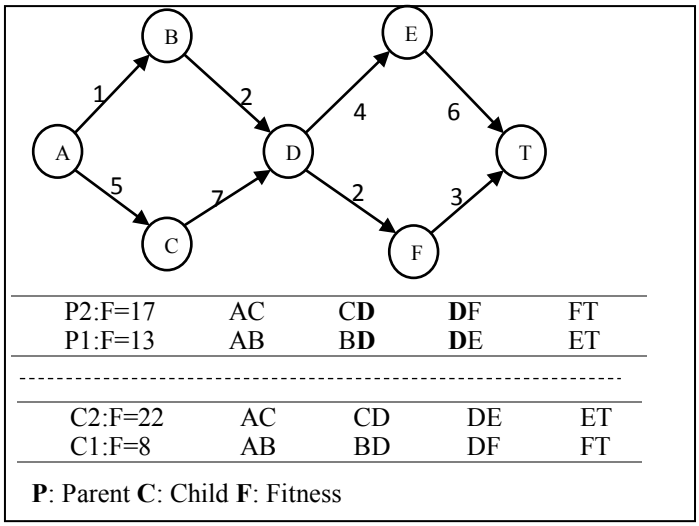

Figure 6. Single point crossover

One point worth mentioning is that after accomplishing the crossover operator, we do not care about the feasibility of the new path generated. The algorithm will always end up with a feasible path from the source to the destination. Therefore, the algorithm does not lose time to test the validity of an offspring nor to perform some additional operations to repair the unfeasible paths.

Considering more than one crossover point may enhance the quality of final solutions. However, that might cost the algorithm additional computational efforts when performing the crossover. Therefore, we decide against using such advanced crossover techniques.

F. Mutation (VNS)

Crossover operation may produce degenerate population. The algorithm may therefore get stuck in local minima. To overcome this issue, the mutation operation is performed.

The VNS is chosen as a special mutation operator. That is, the VNS is applied with low probability over the population's individuals. By doing so, the algorithm is guided towards new regions within the solution space. Therefore, the algorithm's chance to find better solutions increases.

Other mutation techniques have been applied such as order changing. However, we have realized that the mutation in this case may provide invalid paths. Additional processes should therefore be applied to reform infeasible paths. As a result, the mutation computational time will increase. It has been thereby decided against using such traditional mutation techniques.

G. Terminating condition

Approximate methods do not guarantee obtaining optimal solutions. Therefore, additional terminating conditions should be introduced in order to allow the convergence of the algorithm.

Maximum number of generations, fixed execution time, and no modifications in population elements can be considered

as algorithm stopping criteria. We have used in our research two stopping criteria:

The algorithm first stops when no change in the fitness of each individual in the population has been detected. We have used 100 generations as a number to ensure a fixed state in the population. Another stopping criterion is attended when the algorithm reaches the maximum number (say 500) of generations. We have noticed after some experimentations that our algorithm visits wide range of the search space rapidly.

Thus, there is a big chance that the algorithm converges rapidly. That explains the small number of generations defined for the stopping criterion.

IV. EXPERIMENTAL RESULTS

We have done experimentations over 50 different networks instances (from small to large size). Some of them are generated randomly thanks to a generator that we developed using Java. Other benchmarks are taken from the DIMACS website that offers graphs for real world road networks in USA.

We run algorithms and solvers on an Intel core I5 machine of 8 GB RAM. Besides, we made extensive use of generic programming techniques in order to avoid runtime overheads.

We put also particular efforts into carefully implementing efficient data structures.

For simplicity, we present in table3 results obtained from applying the proposed algorithms over six network instances. For each instance, we choose five-times two random points to construct the origin-destination queries .We compared our approach with two other exact algorithms (Dijkstra and IP). We implemented Dijkstra using a priority queue and we solved the IP generated using two solvers: Cplex 12.6 and gurobi 6.0.

The results showed that the running time of Dijkstra is highly better than IP weather it is solved by Cplex or by gurobi. However, our method outperforms Dijkstra and IP with 5% average gap to the optimality. The average speed of our method is 20-times faster than Dijkstra and more than 1000-times compared with IP.

Moreover, we can notice from the results that the time spent to solve the IP by the CPLEX solver is not the same as in GUROBI solver. It changes at each request. Although this research does not focus on accelerating the IP's solving time, we believe that the techniques used in both solvers can be enhanced. That point may be considered as a future work since it can tell us when and how we should change the parameters of our solvers depending on the problem instance.

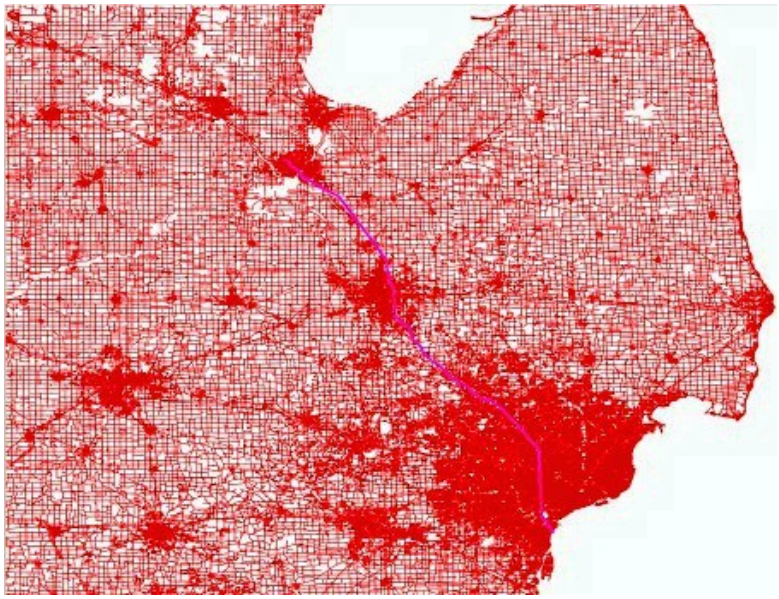

Figure 6. A real road network from DIMACS Challenge with 2,758,119 nodes and 6,885,658 edges

Table 3. Experimental results

GRAPH			RUNNING TIME (MSEC)				AVERAGE	
TYPE	JUNCTIONS	SEGMENTS	DIJKSTRA	IP		OUR ALGORITHM	GAP (%)	SPEED
				CPLEX	GUROBI			
RANDOM	1000	100000	194,29	7773,63	5767,28	2,37	0	81
			249,24	7229,51	5149,81	0,95	6	262
			54,13	7315,10	6081,39	1,23	4	44
			53,05	7551,22	5491,86	1,00	1	53

			53,81	7820,56	5907,35	1,10	0	48
RANDOM	25000	500000	797,66	153502,72	29839,36	3,13	0	254
			814,44	96358,00	18549,06	2,40	0	339
			578,03	83066,09	17961,71	1,87	0	309
			492,50	84785,43	18112,86	1,50	4	328
			538,94	87916,22	19438,90	2,53	0	213
COMPLETE	150	22350	15,96	440,10	153,98	0,53	0	30
			18,94	386,89	169,83	0,40	0	47
			5,19	418,59	216,41	0,50	0	10
			2,84	401,78	195,37	0,39	0	7
			2,26	404,61	143,91	0,37	0	6
COMPLETE	500	249500	12,53	2485,40	2451,00	0,27	2	45
			12,87	5972,23	1514,43	0,29	1	44
			28,78	2689,79	2686,63	1,03	0	28
			25,05	2481,10	1189,08	0,90	0	28
			23,84	2447,22	2225,44	1,27	0	19
DIMACS	1207946	2840210	2486,35	716962,02	169921,86	83,23	5	30
			1961,94	728352,15	2922319,64	126,26	2	16
			2150,12	772732,06	8993880,24	102,96	3	21
			1040,67	621584,55	9336397,105	40,96	4	25
			1330,92	586573,61	75536,39705	29,49	6	45
DIMACS	1890816	4657744	4650,82	-	-	129,40	4	36
			5615,06	-	-	134,30	2	42
			5088,58	-	-	114,94	0	44
			5984,08	-	-	123,27	2	49
			6214,16	-	-	115,12	1	54

V. CONCLUSION

The challenge of computing the one to one shortest path in road networks has been addressed in this paper. A new hybrid metaheuristic has been proposed and a compared with other exact algorithms has been done.

Results has proven that the combination made between GA and VNS is a powerful tool for efficiently solving routing issue in road networks.

Although the proposed approach succeeds in finding high quality solutions within milliseconds, it may not outperform some advanced speed up techniques, which are only dedicated for routing in road networks. However, the originality of the proposed approach stems from the flexibility/capacity of the selected meta-heuristics in handling additional problem requirements such as routing in a multimodal transportation network or optimizing more than one criterion at the same time.

Currently, we are applying this method for solving multimodal routing issue in a dynamic/stochastic transportation system. We are mainly considering walking, road and railway networks. The results we have obtained so far are promising and further papers will be published in the near future.

ACKNOWLEDGMENT

This research work has been carried out in the framework of the Technological Research Institute SystemX, and therefore granted with public funds within the scope of the French Program “Investissements d’Avenir”.

REFERENCES

[1] Gonen, Bilal. "Genetic Algorithm finding the shortest path in Networks." Reno: University of Nevada (2006).

[2] Behzadi, Saeed, and ALIA ALESHEIKH. "Developing a Genetic Algorithm for Solving Shortest Path Problem." NEW ASPECTS OF URBAN PLANNING AND TRANSPORTATION (2008).

[3] Kumar, Dr Rakesh, and Mahesh Kumar. "Exploring Genetic algorithm for shortest path optimization in data networks." Global Journal of Computer Science and Technology 10.11 (2010).

[4] Mohemmed, Ammar W., Nirod Chandra Sahoo, and Tan Kim Geok. "A new particle swarm optimization based algorithm for solving shortest-paths tree problem." Evolutionary Computation, 2007. CEC 2007. IEEE Congress on. IEEE, 2007