Autonomous taxi operations: algorithms for the solution of the routing problem

Giuseppe Sirigu, Mario Cassaro, Manuela Battipede, Piero Gili

Autonomous taxi operations: algorithms for the solution of the routing problem

Giuseppe Sirigu*

Ph.D. Candidate, Department of Mechanical and Aerospace Engineering, Politecnico di Torino, 10129, Torino, Italy

Mario Cassaro†

Post-Doc Researcher, Dpartement commandes des systmes et dynamique de vol, ONERA, Toulouse, France

Manuela Battipede‡ and Piero Gili‡

Associate Professor, Department of Mechanical and Aerospace Engineering, Politecnico di Torino, 10129, Torino, Italy

In past decades, airport ground operations have attracted researchers, with the aim of increasing airport efficiency and reducing the environmental impact of airport operations. Airplane taxi operations have received particular attention for their significant impact on the airport efficiency and pollutant emissions and on the fuel cost for airlines. Alternative solutions have been proposed to the engine-on taxi procedures, including the employment of autonomous vehicles to tow the aircraft between gates and runways. In order to be performed, autonomous taxi procedures require precise planning and scheduling by means of sophisticated management systems. At the base of these management systems, lie algorithms for the solution of the routing problem, which provide feasible paths on the airport surface. Two different approaches can be used: compute the paths on the fly, or pre-compute all the possible paths between all the pairs of starting/ending points on the airport grid and store them in a database that is called when needed. In this paper, four different algorithms are implemented and compared for the computation of paths on the fly: two Hopfield-type neural networks and two algorithms based on graph theory. Furthermore, two algorithms for the generation of the path database are presented: a modified version of the Breadth-first search and an implementation of the k-shortest paths algorithm. Each taxi mission, performed by the tractors, consists of three different events, called phases: one central towing phase, where the tractor tows the aircraft between gate and runway and two repositioning phases in which the tractors move from its actual position to the airplane or from the airplane back to the depot.

I. Introduction

The current vivid interest of the civil aviation in the development and implementation of innovative airport management and ground operation systems is prompted by the antithetical needs of increasing the air traffic, reducing the air pollutant and noise emissions. In particular, airport congestion is globally recognized as one of the most prominent problem areas of the next future and the straightforward solution of expanding the airfields is not that efficient and not always doable. In fact, the higher complexity of air terminals deriving from the addition of runways and taxiways will penalize the overall system efficiency by increasing the human workload and error risk, resulting in potentially hazardous situations. For example, Hilburn in[1] and the ICAO regulations[2] demonstrate the high level of risk derived by the Head Down Operations (HDOs) during taxi phases. In addition, the increasing aircraft number, jointly with the longer taxiing, will significantly contribute to an increase in fuel burn and emissions, which is in contrast with the stringent environmental regulations. Therefore, the concept of autonomous engines-off taxiing, using towing vehicles, has been recently investigated and recognized as a feasible and effective solution to the aforementioned issues. It consists in employing push-back tractors to tow the aircraft from gate to runway for departure and from runway to gate for arrival, while keeping engines off during ground movements. To make the new concept operative, some structural

*Ph.D. Candidate, Department of Mechanical and Aerospace Engineering, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129, Torino, Italy.

†Post-Doc Researcher, Dpartement commandes des systmes et dynamique de vol, ONERA, Toulouse, France

‡Associate Professor, Department of Mechanical and Aerospace Engineering, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129, Torino, Italy.

and regulatory issues need to be solved. Structurally, nose landing gears (NLGs) are not designed to be towed for long distances; this feature can lead to fatigue failures due to transversal loads. Towbarless systems decrease NLG loads, while ensuring an effective connection between the aircraft and the tractor. Thus, they can be used to tow the aircraft along the entire taxiing path. Indeed, aeronautical industries and academics are currently investigating different solutions, ranging from a semi-robotic towbarless tractor[3] to a completely autonomous system.[4] In the Taxibot solution, developed by an industrial consortium, the thrust required for taxing is provided by a diesel engine tug, while the direction control is managed by the aircraft pilot through the aircraft steering system.[3] This control setup allows to avoid regulations' limitations since the pilot stays in charge of the ground maneuvering. Even if the explained solution seems to be of ease implementation in short time and resolve the air-noise pollution and fuel consumption issues, it has a major drawback that is the increased hazardous conditions deriving from the higher complexity level of operations, to be performed by pilots and ground operators. On the other hand, an autonomous external taxiing system, with a manned tugs that tows and drives utterly the aircraft through the taxiways, will reduce the pilot workload and will optimize the airport ground movements. In this configuration, all the pre-flight procedures (*e.g.* aircraft system functionality checks, Flight Management System setup and engine warm up) might be performed by the pilot during the semi-autonomous taxi, without loss of safety.[4] The drawback consists of a regulatory limitation, especially in terms of responsibility allocation, deriving from the lack of Pilot in Control (PIC) situation.
To this purpose, another concept of airport ground operation system has been presented by our research team.[5] It consists in a semi-automatic system, activated by the control tower, in which autonomous auto-piloted tractors are capable of accomplishing towing missions between points, selected by the tower operators. Albeit a full-authority control system for tug selection and route conflict avoidance will be required for the final application, at the project status, the route selection problem is the major objective. The route selection problem is thoroughly addressed in literature and it has been solve by several techniques such as genetic algorithms,[6] heuristic methods[7,8,9] and tabu search algorithms.[10] Neural networks have been also used for vehicle and computer network routing. In particular, Hopfield-type Neural Networks (HNNs)[11,12,13] have been extensively used in the past to these purposes, as they minimize an energy function, which can be properly defined to perform a shortest path search[14,15] or can be used in a non-static environment.[16] Lately, among all the other solutions, the graph theory and the Dijkstra's algorithm became established methodologies, because of their fastness and robustness, to solve routing selection problems in any kind of environment[17,18,19]. In addition, a modified version of the Dijkstra algorithm, which uses heuristic functions to define a preferential search direction, called A^*[20,21],[22] allows for further computational time reduction with respect to the standard Dijkstra's algorithm, particularly noticeable for large dimensions of the searching domain.
In this paper, a software solution for the push-back tractor route selection problem in a discretized airport environment (apron) is presented. Four different algorithms are implemented: two Hopfield-type neural networks and two algorithms based on the graph theory. Their objective is the shortest path, accounting for restricted airport areas, preferential directions and dynamic obstacles. The computed route checkpoints serve as reference for the tractor autopilot. Two different missions are analyzed, concerning the towing of departing and arriving aircraft respectively. A single mission consists of three different events, called phases: Phase 1 goes from the actual tractor position (eventually the parking zone) to the selected aircraft (parked or just landed); Phase 2 is the actual engine-off taxi towing; Phase 3 is the tractor return to its own parking zone. The paper is organized as follows: Section 2 gives an overview of the autonomous taxi system, which includes the routing algorithm. Section 3 contains the problem formulation referred to our test case airport; in Section 4 the mathematical formulation of the proposed algorithms is presented and their application justified. Section 5 describes the algorithms' implementation and the results obtained for the selected test case. Pertinent conclusions are reported in the closing section.

II. Autonomous taxi system

The proposed autonomous fleet management system has been designed to solve three main problems: the departure sequence scheduling, the pushback scheduling, and the taxi route planning. Furthermore, the tug dispatch problem is solved, assigning an autonomous vehicle to each mission (departures and arrivals). The output of the algorithm is a schedule for each tug that enables just-in-time runway operations: the taxi-in starts as soon as the aircraft clears the runway and connects to the tug, whereas the airplane is delivered to the runway at the time when it should takeoff. The structure of the fleet management system is shown in Fig. 1.[23]

In order to define the tug schedules, the algorithm assigns a trajectory to each tractor mission based on the data provided by the runway and the pushback schedulers, and evaluates the cost associated with it in terms of the electric energy consumed. After the best set of non-conflicting trajectories has been found, the tasks are assigned to each tractor in the fleet, thereby maximizing the tug utilization. Finally, the generated schedules are communicated to air

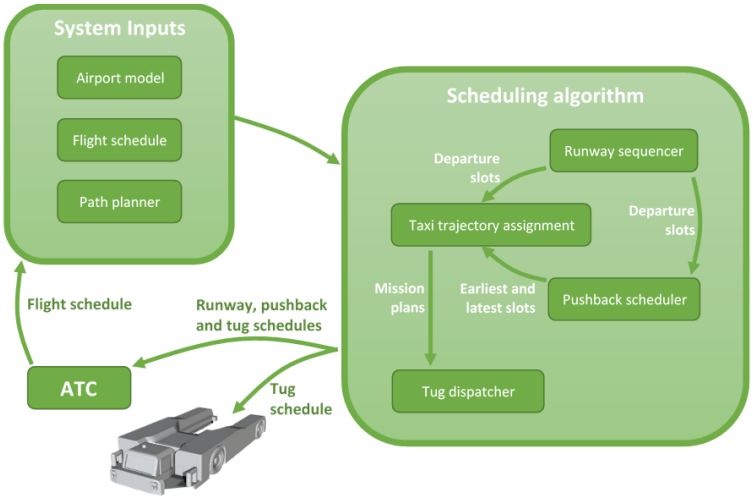

Figure 1. Fleet management system.

traffic control (ATC) and to the tugs.

If the considered airport is characterized by the presence of more than one runway, the airplanes might be required to cross one or more active runways to reach their assigned runway or stand. To prevent excessive numbers of crossings and consequently the risk of runway incursions, the proposed autonomous taxi system is designed to tow the aircraft to the closest runway to the apron or vise versa. Therefore, aircraft must use their own engines to go from the disconnection point to the take-off runway or from the landing runway to the connection point. As far as departing aircraft are concerned, the time spent traveling from the disconnection point to the take-off runway with their own engines enables the engine to warm up before departure. In order to define the time at which a landing airplane reaches the connection point, a constant taxi speed of 10 m/s is considered; if the runway to be crossed is busy, the airplane will stop at the holding point until the runway is clear.

In this context, a path planner is required to provide the path for every taxi mission as an input to the scheduling algorithm. Two different approaches could be used to generate the paths required for solving the scheduling problem: the paths could be generated on the fly when required by the algorithm or a database containing all the possible paths between points of interest in the airport (i.e. taxiway insertion points, aircraft parking lots, and tractor deposit) could be precomputed and loaded as a system input.

III. Problem Formulation

A single mission of the tug is discretized in three different phases which have different initial and final checkpoints, whether an aircraft departure or arrival mission is being performed.

- Phase 1: reaching the targeted aircraft (A/C parking slot for departure mission, runways for arrival mission);
- Phase 2: perform engine-off taxi towing (apron to runways for departure mission, vice versa for arrival mission);
- Phase 3: returning to the tug parking area (runways to parking for departure mission, apron to parking for arrival mission).

As already mentioned, the problem to be solved for each phase is a shortest path route selection problem in a non-static environment with constraints. The considered constraints are the apron ground vehicles permitted ways, the one-way only routes and the different obstacles that can obstruct the way. Therefore, the trivial straight-line trajectory between initial and ending checkpoints of each phase is not always a solution of the problem. Fig. 2, which represents our test case, shows the actual taxiways and ground vehicle permitted areas in the *Sandro Pertini* airport of Turin.

For the proposed application, the spatial domain is given by the airport runways, taxiways, ground vehicles routes, and aircraft parking lots. Route selection algorithms require a domain discretization in checkpoints and arcs with directions and possible obstacles. For the proposed application, this operation is automatically performed by a tool coded on purpose that discretizes differently depending on the path optimization algorithm to be used.

IV. Proposed Algorithms

The proposed algorithm consists of two main parts: the first one loads the airport geographical data (longitude and latitude) and discretizes the domain, based on the apron features; the second solves the route selection problem computing the shortest path for each mission phase. The computation on the domain is performed only once at the beginning, while the route selection algorithm is run three times per mission (arrival or departure). This section gives an overview of the mathematical formulations of the different methodologies employed and compared to solve the route selection problem under analysis.

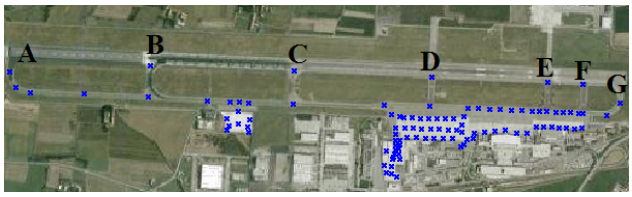

Figure 2. Turin Airport "Sandro Pertini", satellite picture with directions for taxi and ground movement. Credits: Pictures 2015 DigitalGlobe, Map data ©2015 Google

A. On the fly path generation

1. *Neural Network*

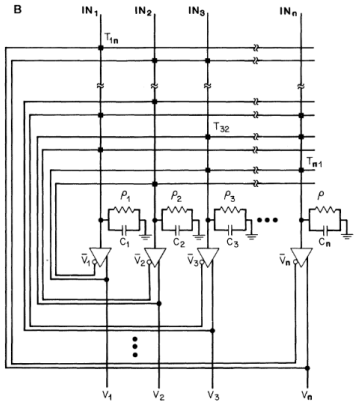

Figure 3. Hopfield neural network electric circuit model.[13]

HOPFIELD NEURAL NETWORK. Artificial neural networks are paradigms of a non-analytic way of solving problems, which are applied in different fields of science and engineering because of their versatility.[24] This system, inspired by biological nervous systems, is able to learn and make intelligent decisions without a precisely defined algorithm or a complete set of input data. Although the work of individual neurons can be quite slow, the overall neural network is very fast, which is due to the large neuronal connections and parallel processing,[24,25] Among all types of neural networks, particular attention is given here to the recursive neural networks,[14] which are characterized by a feedback signal that allows the net to use the neuron output value as an input, to obtain a faster and more robust response. Typically, this is the time delay of the signal.[24] Hopfield's neural networks are representative of this type of networks,[11,12] and its structure is shown in Fig. 3.

One of the main features of the Hopfield neural network is the possibility of a simple hardware implementation. Hopfield and Tank proposed their neural network structure[12] capable of solving different complex problems by the minimization of an energy function that has to be properly defined. This approach was demonstrated on the well-known and computationally very complex Traveling Salesman Problem (TSP) with 30 nodes.[12] Since then, many researchers have used similar models for solving a large variety of combinatorial optimization problems.

MODIFIED HOPFIELD NEURAL NETWORK. An Hopfield-type neural network has been proposed by Zhong et al.[16] for real-time collision-free robot path planning in a non-static environment. This approach is based on the propagation of the target activity through the local connectivity of neurons, which is formulated using harmonic functions. The main advantage in using harmonic functions lies in the local minima removal. In their application, Zhong et al. used the Laplace operator discretization applying a finite difference scheme on a 8 neighbors grid.[16]

2. *Graph Theory*

Graph representation is very effective in describing real-world situations. Graphs are schemes consisting in points (nodes) and links (edges) connecting a pairs of nodes.[26] The basic graph form is composed of unweighted and undirected edges. However, some real applications could be more effectively represented by means of a weighted graph. The main idea is to associate a weight, or cost, to each edge and the total cost of a path in the graph is defined as the sum of the costs of each selected edge.[27] This approach allows for easy representation of optimization problems aiming to find a graph subset with minimum or maximum weight, as the shortest route selection problem. Furthermore, by specifying edges directions, it is possible to reproduce an asymmetrical domain. For the application under investigation, the relative distance between two points is used as edge weight and the edge direction is used to simulate one-ways only routes.

DIJKSTRA'S ALGORITHM. The original version of the Dijkstra's algorithm finds the shortest path between a defined node in the graph and every other; however, the algorithm can be stopped when the shortest path to the destination node has been determined. It is an example of greedy algorithm; in fact, at each step, the neighbor with shortest distance from the actual considered point is chosen as sequent point in the path.

A* ALGORITHM As mentioned, the *A** algorithm is an improvement of the Dijkstra's algorithm obtained by adding a heuristic function to fasten the solution.[20] The introduced evaluation function is

$$f(v) = l(v) + h(v) \tag{1}$$

where, $h(v)$ is the distance from v to t. Thus, $f(v)$ represents the distance from s to t, passing through v. $h(v)$ can be estimated by means of an heuristic function $\hat{h}(v)$, for example the Euclidean distance. However, the optimal heuristic function is problem dependent and so forth cannot be defined uniquely.

B. Path database generation

Although several algorithms have been proposed in literature to select the shortest path between two nodes in a graph, there is no widely used algorithm for finding all the possible paths between two nodes. A modified version of the breadth-first search (BFS) algorithm was implemented; the proposed approach does not take into account the distance between the nodes during the path construction and provides paths visiting each node no more than once.

However, as far as big airports are considered, the number of existing paths, given an origin and a destination, can be considerable (over 100 in the Schiphol airport), with most of them resulting too long to effectively improve the optimization solution. Consequently, an algorithm able to find a limited number of paths is required. Several algorithms for the solution of the k-shortest path problem have been proposed.[28–30] Yen's algorithm was developed to compute the k shortest loop-less paths in a positive weighted graph.[28] The original version is based on the Dijkstra's algorithm for the computation of the shortest path between two nodes; however, different algorithms can be used within Yen's scheme. From a time complexity point of view, if the Dijkstra's algorithm is used, Yen's algorithm worst case running time is $\mathcal{O}\left(kn\left(m + n \cdot \log n\right)\right)$, with n and m respectively number of nodes and number of edges in the graph.[31]

Conversely to Yen's algorithm, in Eppstein's one, paths are not required to be loop-less; this is the simplest algorithm for the solution of the k-shortest path problem. Furthermore, it has lower time complexity than the former, which, with recent improvements reached the optimal time of $\mathcal{O}\left(m + n \cdot \log n + k\right)$.[31]

For the present application, Yen's algorithm has been implemented to find the $k = 5$ shortest paths, thereby creating a database for each considered airport, containing the 5 shortest paths for each starting/ending node pairs of the taxi grid.

V. Implementation and Results

The proposed algorithms for the path generation on the fly have been implemented for test and comparison purposes in Matlab® environment without using any existing toolbox. The code, which is optimized for airport environment, allows for an automatic discretization of the apron area of certain airports belonging to an available database.

A. Results comparison

The result analysis suggests that the final optimized solutions, computed by the four implemented algorithms, slightly differ; this might be caused by the different domain discretization employed. However, the length of the paths are very similar as shown in Table 1 for the departure mission.

A substantial difference between the implemented algorithms arises when considering the computational time required to generate the optimal shortest path. The code was executed on a Windows 8.1 Pro platform supported by an Intel Core i7-4810MQ CPU and 8 GB RAM. Table 2 reports the required computational time for each algorithm for both test cases. It is clear that the computational time required by the graph theory-based algorithms is some order of magnitude smaller than the one required by the two neural networks. In particular, the modified HNN is the slowest algorithm, because of the huge amount of nodes (neurons) required for the proposed application, especially during the first phase of the computation, when the target activation must propagate to the initial tug position so that it starts virtually moving.

Table 1. Optimal trajectories length comparison for the Departure mission.

Algorithm	Phase1 [m]	Phase2 [m]	Phase3 [m]	Total length [m]
HNN	1097.20	1442.50	1342.60	3882.30
mHNN	1071.90	1407.50	1376.10	3855.50
Dijkstra	1097.20	1442.50	1342.60	3882.30
A*	1097.20	1442.50	1342.60	3882.30

Table 2. Computational time comparison for both missions.

Algorithm	*Departure* [s]	*Arrival* [s]
HNN	705.01	767.83
mHNN	1839.66	1272.93
Dijkstra	0.01	0.01
A*	0.01	0.01

VI. Conclusion

This paper presents a software solution for a route selection problem applied to airport environment for the path planning of innovative and automatic aircraft tugs, responding to the need of engine-off taxi in the next airport generation. Four different algorithms, based on artificial intelligence and graph theory, have been implemented for the "on the fly" generation and their performance compared. The developed software works with any airport and is able to properly discretize the spatial domain (*e.g.* taxiways, runways and aircraft parking lots) in function of the employed algorithm. To solve the problem, an entire engine-off towing operation is called mission and is divided into three sub-operations, called mission phases. Furthermore, two algorithms for the pre-computation of a path database. A unique test case, referred to the real data of the *Sandro Pertini* airport of Turin, has been performed for each algorithm for comparison purposes. The results demonstrate that all the algorithms converge to a unique shortest path, exception done for the modified HNN, which result in a slightly longer route. In addition, the two graph theory based algorithms are several order of magnitude more efficient, requiring an almost negligible computational time, making them more attractive for a possible real-time application.

References

[1]B., H., "Head Down Time in Aerodrome Operations: a Scope Study," 2004.

[2]ICAO, "Manual on the Prevention of Runway Incursions," Tech. Rep. Doc 9870 AN/463, International Civil Aviation Organization, 2007.

[3]Lufthansa, "Innovative TaxiBot now used in real flight operations," 2015.

[4]Kim, J., K., A., K., A., and T.D., R., "ANTS-Automated NextGen Taxi System," FAA Design Competition 2009-2010, 2010.

[5]Battipede, M., Della Corte, A., Vazzola, M., and Tancredi, D., "Innovative airplane ground handling system for green operations," 27^{th} *International Congress Of The Aeronautical Sciences ICAS*, 2010.

[6]B.M., B. and M.A., A., "A genetic algorithm for the vehicle routing problem," *Computers and Operations Research*, Vol. 30, No. 5, 2003, pp. 787–800.

[7]S., K.-H., C., P., A., Y., and M., R., "Heuristics and memetic algorithm for the two-dimensional loading capacitated vehicle routing problem with time windows," *Central European Journal of Operations Research*, Vol. 21, No. 2, 2013, pp. 307–336.

[8]B., B. and H., G., "Planning as heuristic search," *Artificial Intelligence*, Vol. 129, No. 12, 2001, pp. 5 – 33.

[9]S., M.-M., R., K., and G., L., "Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows," *Transportation Research Part B: Methodological*, Vol. 38, No. 8, 2004, pp. 669 – 685.

[10]M., G., F., G., Potvin, J., and ., T., "Parallel Tabu Search for Real-Time Vehicle Routing and Dispatching," *Transportation Science*, Vol. 33, No. 4, 1999, pp. 381–390.

[11]J.J., H., "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences*, Vol. 79, No. 8, 1982, pp. 2554–2558.

[12]J.J., H. and D.W., T., "Neural computation of decisions in optimization problems," *Biological Cybernetics*, Vol. 52, No. 3, 1985, pp. 141–152.

[13]J.J., H. and D.W., T., "Computing with neural circuits: a model," *Science*, Vol. 233, No. 4764, 1986, pp. 625–633.

[14]M.K.M, A. and F., K., "Neural networks for shortest path computation and routing in computer networks," *Neural Networks, IEEE Transactions on*, Vol. 4, No. 6, Nov 1993, pp. 941–954.

[15]N., K., I., R., and B., R., "Route Selection Problem Based on Hopfield Neural Network," *Radioengineering*, Vol. 22, No. 4, 2013, pp. 1182–1193.

[16]Y., Z., B., S., and Y., T., "A new neural network for robot path planning," *Advanced Intelligent Mechatronics, 2008. AIM 2008. IEEE/ASME International Conference on*, July 2008, pp. 1361–1366.

[17]E.W., D., "A note on two problems in connexion with graphs," *Numerische Mathematik*, Vol. 1, No. 1, 1959, pp. 269–271.

[18]M.H., X., Y.Q., L., Q.L., H., Y.X., Z., and G.F., L., "An improved Dijkstras shortest path algorithm for sparse network," *Applied Mathematics and Computation*, Vol. 185, No. 1, 2007, pp. 247 – 254.

[19]A.V., G. and R.E., T., "Expected Performance of Dijkstra's Shortest Path Algorithm," Tech. rep., NEC RESEARCH INSTITUTE REPORT, 1996.

[20]P.E., H., N.J., N., and B., R., "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *Systems Science and Cybernetics, IEEE Transactions on*, Vol. 4, No. 2, July 1968, pp. 100–107.

[21]W., Z. and R.L., C., "Finding Shortest Paths on Real Road Networks: The Case for A*," *Int. J. Geogr. Inf. Sci.*, Vol. 23, No. 4, April 2009, pp. 531–543.

[22]C., L., "An Iterative A* Algorithm for Planning of Airport Ground Movements," *Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence*, 2010, pp. 413–418.

[23]Sirigu, G., Battipede, M., Clarke, J.-P., and Gili, P., "A Fleet Management Algorithm for Automatic Taxi Operations," *ICRAT 2016 7th International Conference on Research in Air Transportation*, 2016, pp. 1–5.

[24]Haykin, S., *Neural Networks: A Comprehensive Foundation*, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd ed., 1998.

[25]Rahman, S., Ansari, M., Moinuddin, A., and et al., "Solution of Linear Programming Problems using a Neural Network with Non-Linear Feedback," *Radioengineering*, Vol. 21, No. 4, 2012, pp. 1171.

[26]A.J., B. and U.S.R., M., *Graph Theory with Applications*.

[27]N., C., *Graph Theory: An Algorithmic Approach*, Computer Science and Applied Mathematics.

[28]Yen, J. Y., "Finding the k shortest loopless paths in a network," *management Science*, Vol. 17, No. 11, 1971, pp. 712–716.

[29]Eppstein, D., "Finding the k shortest paths," *SIAM Journal on computing*, Vol. 28, No. 2, 1998, pp. 652–673.

[30]Aljazzar, H. and Leue, S., "K: A heuristic search algorithm for finding the k shortest paths," *Artificial Intelligence*, Vol. 175, No. 18, 2011, pp. 2129–2154.

[31]Hershberger, J., Maxel, M., and Suri, S., "Finding the k shortest simple paths: A new algorithm and its implementation," *ACM Transactions on Algorithms (TALG)*, Vol. 3, No. 4, 2007, pp. 45.