An Optimization Approach for Airport Ground Operations with A Shortest Path Algorithm

Orhan Eroglu, Zafer Altug Sayar
TUBITAK BILGEM, BTE
Smart Transport Systems, Senior Researcher
Kocaeli, Turkey
orhan.eroglu@tubitak.gov.tr, zafer.sayar@tubitak.gov.tr

Guray Yilmaz
Turkish Air Force Academy, ASTIN
Computer Engineering Department, Chair
Istanbul, Turkey
g.yilmaz@hho.edu.tr

***Abstract*—Tower and ground controls take significant role as much as arrival and transit traffic management on aeronautics. Both the safety of the vehicles and livings on the airport, and minimization requirement for flight costs with the help of optimizations on ground operations point out the importance of tower and ground controls. Especially the recent improvements on air transport have resulted in a drastic increase in air traffic intensity. As an inherent consequence of this progress, ground traffic volume must be diminished starting from modification on behaviors of air traffic controllers. Optimization of ground movements of vehicles by means of advanced algorithms is taken into consideration in this study.**

The study aims to compute the shortest paths for vehicles from apron or their current position to especially runway thresholds, taxi endpoints or any position and to report the results to controllers. Although such reports cannot be restrictive, those can be significantly usable decision support alternatives for controllers.

In order to implement the system under consideration, Floyd-Warshall shortest path algorithm has been utilized. This node-based algorithm has been chosen since the ground routings in an airport are performed by controllers using virtual node points defined on the taxiways. The system created with the help of this algorithm is named "Floyd-Warshall Ground Route Optimization System" (FLOW-GRO). In the FLOW-GRO system, the minimum taxi-time between each node in the airport is taken as the weight of that edge since the maximum taxi speed on each taxiway segment may vary. Algorithm is fed by a start and an end point as inputs, and returns a list of nodes as an output that represents the shortest way. The application of the FLOW-GRO System has been examined in a well-working Air Traffic Management Simulation System (ATCTRSIM) that has been developed by TUBITAK BILGEM.

As an inevitable conclusion of this study may be that, ground traffic controllers can be continuously helped with shortest path suggestions generated by a decision support system. Therefore, controllers may acquire a tendency to create vehicle routes in the shortest possible way in the long run, thanks to the system. A potential future phase of the study may be the evaluation of the routing performance of traffic controllers over the ground movements with the help of the system. In addition, conflict avoidance in a dynamic manner can be examined.

Keywords—Shortest path, Floyd Warshall, optimization, ground controllers, FLOW-GRO.

I. Introduction

In recent years, whole the world faces the increasing demand for air transport and respectively increasing air traffic loads. Air traffic is seen to going to be twice or three times the current density, in 2020's by authorities [1].

Tower control and ground traffic management include the traffic movements performed on the parking bays in airports, taxiways and runway thresholds, and on the runways throughout the takeoff. If the fact that, even the current load on the ground traffic significantly exhausts traffic controllers in tower control is taken into consideration; any increase in the ground traffic will lead the control and management of ground traffic to become more difficult.

Very initial ones of precautions in order to cope with tower control and ground traffic that get harder day by day, may be listed as revision and optimization of traffic control procedures and standards, improving monitoring abilities by development of new traffic surveillance and communication technologies, and hardening the practices of air traffic controllers. In addition, new decision support systems usable for controllers to manage ground traffic may be developed. Decision support systems are computer-based systems that basically suggest varying potentially useful directives for human operators or users before or during any decision [2].

One of the major handicaps about airports, which have high ground traffic load, is that route conflicts and routing delays may occur more often when taxiing of vehicles is not performed in the shortest possible way [3]. Therefore, a computer supported decision system that can continuously suggest the shortest route between a start and a destination position for the sake of managing taxiing operations on the airport may play a significant role on diminishing the ground traffic load.

In this study, a computer system is proposed, which is developed for the aim of generating the shortest path between two points during the taxi procedures of ground traffics. It is assumed that aprons, taxiways and lineup areas in airports all construct a graph structure including a number of nodes. Therefore, Floyd-Warshall shortest path algorithm [4] is utilized to form the shortest taxiway between any two points in such a graph. Implementing this algorithm, the FLOW-GRO system, which can act as a decision support system for the aim of generating the optimum taxiways, is created. The FLOW-GRO system has been examined and tested by tens of different taxi routings using the node-based taxiway graphs of a set of airports form Turkey, and results have been reported.

II. Floyd-Warshall Shortest Path Algorithm

The base of the Floyd-Warshall algorithm stands on a simple formulation, actually. If a graph consists of X, Y and Z nodes and the shortest distance between X and Z is depicted as min(distance(X,Z), then this distance is equal to the sum of the distance between X and Y (distance(X,Y)) and the distance between Y and Z (distance(Y,Z)). The algorithm is the systematic application of this process for the whole graph.

The pseudo-code of the Floyd-Warshall algorithm is shown in the **Fig. 1**. The algorithm takes advantage of two matrices: The D matrix includes the distances between each connected node-pair, and the S matrix includes the sequential relationship between nodes, i.e. it holds the last node before the terminal one, if it exists, in the shortest path between two nodes. The indices ***k*** stands for the current iteration number, ***i*** stand for the row number of the matrices and ***j*** stands for the column number of the matrices. The algorithm iterates one minus the number of nodes and updates the matrices in each iteration. At the end of all the iterations, there exist two matrices keeping the shortest distances and neighborhood information about all the nodes [5].

In order to make the description of the algorithm easier, a simple graph given in the **Fig. 2** is used to show iterations step by step. For example, the distance and sequentiality matrices are created. Since the distance matrix is composed of direct distance between nodes, the very first distance value between two nodes is taken as infinity if they are not directly connected. The sequentiality matrix is initially an empty matrix and its values are generated throughout iterations. However, the each value of columns for the shortest paths of a value in each row may be used for the initial values, i.e. it means a direct connection between any node-pair to exist. The initial case of the D and S matrices (D_0 and S_0) are shown in the **Fig. 3**

```
procedure [array] FloydWarshall(D, S)
for k in 1 to n do
    for i in 1 to n do
        for j in 1 to n do
            if D[i][j] > D[i][k] + D[k][j] then
                D[i][j] = D[i][k] + D[k][j]
                S[i][j] = S[k][j]
Return S
```

Fig. 1 Floyd-Warshall Algorithm

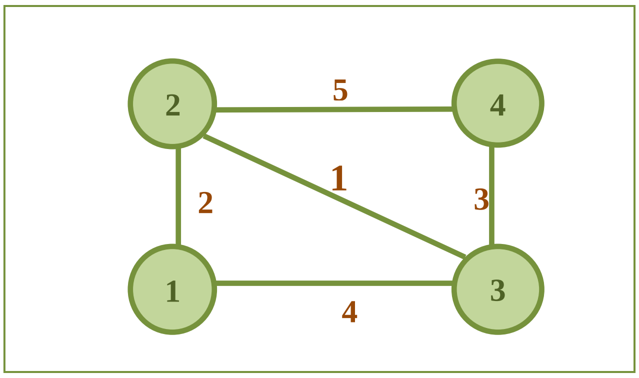

Fig. 2 A demonstration

D0	1	2	3	4
1	-	2	4	∞
2	2	-	1	5
3	4	1	-	3
4	∞	5	3	-

S0	1	2	3	4
1	-	2	3	4
2	1	-	3	4
3	1	2	-	4
4	1	2	3	-

Fig. 3 D_0 and S_0 matrices

The given pseudo-code of the algorithm simply means that, if the sum of new distances between i^{th} and j^{th} nodes when the k^{th} node is placed between them is smaller than the older distance, then the k^{th} node must be placed in the corresponding index in the S matrix the corresponding distance value in the D matrix must be updated with the new value.

The algorithm then iterates and generates D_1 and S_1 matrices (given in the **Fig. 4**) from D_0 and S_0 matrices, D_2 and S_2 matrices from D_1 and S_1 matrices, and eventually D_{n-1} and S_{n-1} matrices from D_{n-2} and S_{n-2} matrices. Basically, the procedure operates as follows: The iteration number ***k*** is also the index of the matrices (D_k and S_k) generated at the end of the iteration. In each iteration, the new values of D_k is obtained updating the D_{k-1} with respect to the node id in the index k. For instance, at the first iteration (***k*** = 1), values in the first row and the first column of D_0 matrix in the **Fig. 3** are used to update the other values in the matrix, respectively.

Sample runs of the pseudo code given in the **Fig. 1** and updates on the matrices are as follows:

For k=1, i=2, j=3:

If $D_0(2,3) > D_0(2,1) + D_0(1,3)$ then

$D_1(2,3) = D_0(2,1) + D_0(1,3)$

If the values in the D matrix are to be used:

Since $1 < 2 + 4$, then no update is performed.

For k=1, i=3, j=4:

If $D_0(3,4) > D_0(3,1) + D_0(1,4)$ then

$D_1(3,4) = D_0(3,1) + D_0(1,4)$

If the values in the D matrix are to be used:

Since $3 < 4 + \infty$, then no update is performed.

Matrices after the first iteration are given in the **Fig. 4**.

D1	1	2	3	4
1	-	2	4	∞
2	2	-	1	5
3	4	1	-	3
4	∞	5	3	-

S1	1	2	3	4
1	-	2	3	4
2	1	-	3	4
3	1	2	-	4
4	1	2	3	-

Fig. 4 D_1 and S_1 matrices

After the generation of D_1 and S_1 matrices, the next iteration starts. Since iteration index ***k=2***, matrix updates are to be performed on the indices excluding ***i=2*** and ***j=2***.

A sample run of the second iteration and updates on the matrices are as follows:

<u>**For k=2, i=1, j=3:**</u>

If $D_1(1,3) > D_1(1,2) + D_1(2,3)$ then

$D_2(1,3) = D_1(1,2) + D_1(2,3)$

If the values in the D matrix are to be used:

Since $4 > 2 + 1$, then update is performed and

$D_2(1,3) = 2 + 1 = 3$, and

$D_2(3,1) = D_2(1,3)$ since the matrix is symmetric, and

$S_2(1,3) = S_2(3,1) = 2$ since the new fact is acquired that the shortest path between the 1^{st} and the 3^{rd} nodes contain the 2^{nd} node.

After performing all the updates of the iteration with the index ***k* = 2**, the D and S matrices have a form as shown in the **Fig. 5**.

Similarly, ***k=3*** and ***k=4*** iterations finally create the eventual matrices given in the **Fig. 6**.

The shortest path between two nodes can be easily observed from the D_4 matrix in the final phase. Additionally, the S_4 matrix holds the sequence of the shortest path between two nodes. For example, the distance between the nodes 1 and 2 is equal to 2 as shown in D_4. In addition, the fact that $S_4(1,2)$ is equal to 2 and $S_4(2,1)$ is equal to 1 shows that the nodes 1 and 2 are directly connected.

As a counter example, the path between the nodes 1 and 4 can be considered. The total distance between two nodes is equal to 6, as shown in the matrix D. When the S matrix is regarded in addition, the fact that path between these two nodes points out the node 3 can be observed. The path is currently 1→3→4 and the sub-path between the nodes 1 and 3 must be concerned. If this process is continued through the step in which there exist no new node in the path, the shortest path between two nodes is found to be 1→2→3→4.

D2	1	2	3	4
1	-	2	3	7
2	2	-	1	5
3	3	1	-	3
4	7	5	3	-

S2	1	2	3	4
1	-	2	2	2
2	1	-	3	4
3	2	2	-	4
4	2	2	3	-

Fig. 5 D_2 and S_2 matrices

D4	1	2	3	4
1	-	2	3	6
2	2	-	1	4
3	3	1	-	3
4	6	4	3	-

S4	1	2	3	4
1	-	2	2	3
2	1	-	3	3
3	2	2	-	4
4	3	3	3	-

Fig. 6 D_4 and S_4 matrices

III. THE FLOW-GRO SYSTEM

The Floyd-Warshall algorithm offers a significantly favorable solution for optimization phenomena in airport ground operations since it works well on problem spaces including node-based graphs. Parking bays, taxiway segments, lineup areas and even runway points on airports can be represented by node points in order to construct the graph model of an airport [6], as shown in **Fig. 7** and **Fig. 8**. Line segments that connect these nodes form the whole taxiway of airports and these segments may be assumed to be edges in the graph model. Therefore, the modeled airport graph, which consists of nodes and edges, suggests a suitable model for Floyd-Warshall algorithm to be executed on.

Performing the Floyd-Warshall algorithm on an airport graph, finding the shortest taxiway between any two airport nodes is guaranteed mathematically. In order to acquire this functionality, The FLOW-GRO system, which can take any node-based airport data as input, has been developed.

The most interesting proposal of the FLOW-GRO system arises on deciding the edge weights. In general, Euclidean distance is the very first option to be used as edge weight since any two taxiway nodes have Euclidean distance between them. On the other hand, the Euclidean distance may be inappropriate in several cases in ground traffic management because taxiway speeds may vary. The main challenge stems from this variation and another parameter should be placed in order to be utilized as edge weight. Thus, the minimum taxiing-time of each edge, which can be calculated by dividing the edge distance by the maximum allowed taxi speed, is chosen for the edge weight.

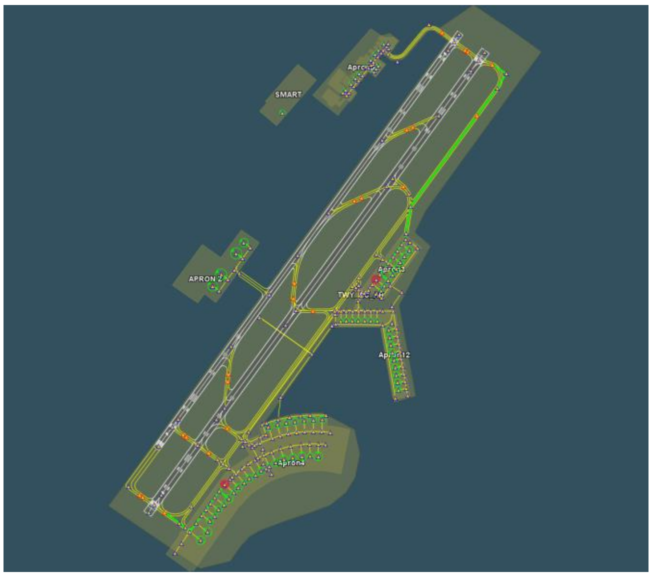

Fig. 7 Airport graph model from TUBITAK ATM simulation

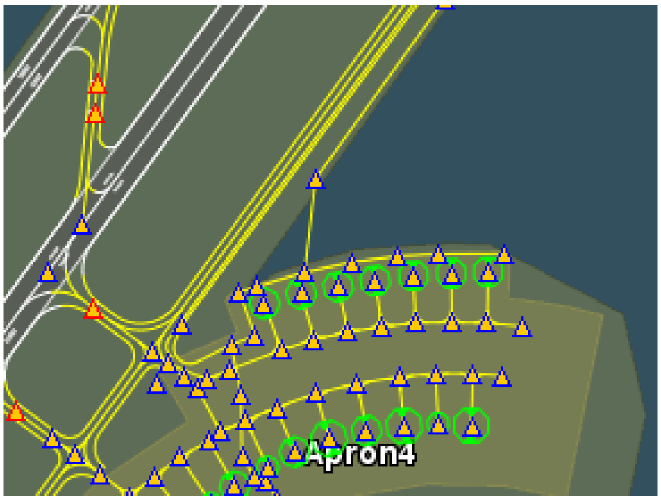

Fig. 8 Airport graph model of TUBITAK (Detailed view)

The FLOW-GRO system has been examined several times in the ATM simulation system that is developed by TUBITAK-BILGEM (The Scientific and Technological Research Council of Turkey – Informatics and Information Security Research Center). This simulation system includes 4 different airports in Turkey and each of them consists of hundreds of alternative, varying speed taxiways.

IV. Conclusion and Future Work

The study do not cope with the hardest challenges, beside it proposes the usage of a well-known method for a common problem in air traffic management: optimization of taxiway routings. It is a simple application work of a node-based algorithm in a domain of more complex computational problems.

As an inevitable future work of this study, the ability of performing conflict avoidance dynamically should be given to the system. In addition to static computation of the shortest taxiways, the system may also determine whether conflict exists between different traffics dynamically, and it may also propose the next possible shortest path for the conflicting ground vehicles.

Another exciting potential of the FLOW-GRO system is that it can be exploited for the sake of evaluating the ability of traffic controllers to generate the most appropriate taxiing routes over the ground movements.

Acknowledgment

Thanks to Isa Tasdelen, who is the chief researcher of the Department of Smart Transportation Systems in TUBITAK, for his valuable discussions. Thanks to my lovely wife and son for their precious existence.

References

[1] H. Erzberger, "Transforming the NAS: The next generation air traffic control system." 24th International Congress of the Aeronautical Sciences, Yokohama, Japan. 2004.

[2] A. P. Sage, Decision support systems. John Wiley & Sons, Inc., 1991.

[3] J. B. Gotteland et al. "Aircraft ground traffic optimization." ATM 2001, 4th USA/Europe Air Traffic Management Research and Development Seminar. 2001.

[4] E. W. Weisstein, "Floyd-Warshall Algorithm.", 2008.

[5] R W. Floyd, "Algorithm 97: shortest path." Communications of the ACM5.6: 345, 1962.

[6] A. G. Marin, "Airport management: taxi planning." Annals of Operations Research 143.1: 191-202, 2006.