11th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference, including the AIA
20 - 22 September 2011, Virginia Beach, VA

TPMagic, a universal airport surface traffic planning analysis and optimization tool

Paul C. Roling[1]
Delft University of Technology, Delft, Netherlands

Adding runways and taxiways is a way of solving capacity problems at major airport. As this also increases the intensity of airport ground operations, safety and efficiency might be compromised. This is one of the main reasons why a significant amount of research has been done in this field, such as research with respect to Advanced Surface Movement Guidance and Control Systems. A taxi planning model developed at the TU Delft uses MILP techniques, combined with pre and processing to allow simulation and optimization of the routing of aircraft on taxiways for major airports. A study of Hartsfield–Jackson Atlanta International Airport shows the benefits an Advanced Surface Movement Guidance and Control Systems can have.

Introduction

As most major airports are operating close to their maximum capacity and are already one of the main causes of delay in the air transport system, the capacity of airport infrastructure is one of the most constraining elements for growth of the system. One of the solutions to deal with these delays is increasing the capacity of airports, which entails enlarging terminals and adding runways and thus also taxiways. As this also increases the intensity of airport ground operations, concerns can be raised both with respect to safety and efficiency, especially under low visibility conditions. A significant amount of research has been conducted to enhance airport ground movement efficiency under all weather conditions, partially with respect to Advanced Surface Movement Guidance and Control Systems (A-SMGCS[1]).

The taxi planning model presented in this paper (TPMagic), based on earlier work[2], uses Mixed Integer Linear programming (MILP) through a commercial package called CPLEX[4]. Whilst it has previously been shown for Amsterdam Schiphol airport, the model has now also been applied to other airports, in order to see if TPMagic is flexible enough. Models have been implemented for London Heathrow, as shown in Figure 3, Frankfurt, Dallas-Fort Worth and Atlanta.

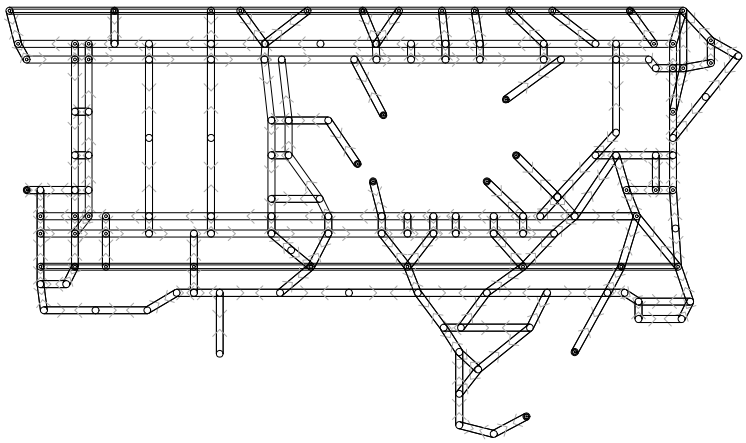

Figure 1. Model of London Heathrow (LHR)

[1] Researcher, Faculty of Aerospace Engineering, P.C.Roling@tudelft.nl

Software design

The taxi planning software can be split up in seven main components, which are illustrated in Figure 2 :

- Airport model generator
- Flight schedule generator
- Route generator
- MILP model builder
- Optimizer
- Output convertor
- Traffic simulator
-

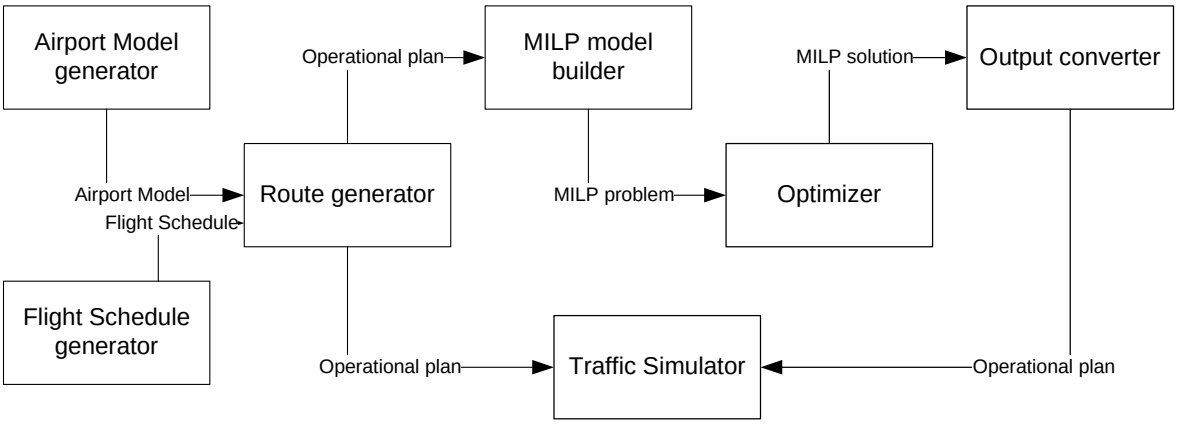

Figure 2. TPMagic components

A. Airport Model Generator

The airport model generator allows the user to create an airport model, as shown in Figure 3, by adding nodes and changing the properties of these nodes. These properties are location (x,y), connections to other nodes, distance to these nodes and whether the current node is a hold, start point or end points as part of a runway or apron.

The location of a node is related to a user defined reference point of the airport model (usually the first node that is entered). By setting a connection to another node this allows aircraft to travel from this node to the other node, simulating a taxiway. The distance to the other nodes is set by default by the relative distance between two nodes, but can also be set manually to a different value for curved or cornering taxiways. Travel in the opposite direction is allowed by setting the current node as a connection in the other node, which should naturally only be done if travel is actually allowed in both directions on the same taxiway. TPMagic includes a routine which automatically makes all taxiways two way.

When a node is defined as a hold node this allows aircraft to stand still and wait on this node, which effectively splits the route an aircraft can take into segments. Whilst adding hold nodes allows more options, it also increases the complexity of the problem and the number of decision variables, thus assigning hold nodes must be done with care. The best nodes to set as hold nodes are the ones right before main crossings.

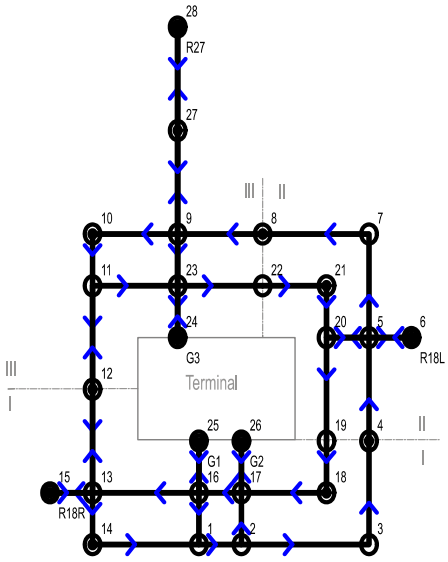

Figure 3. Test airport model

When a node has been defined as part of an apron or a runway it can also be defined as a start or an end point and routes will be calculated to or from that node and that node can be assigned to flights in the flight schedule.

When an aircraft uses a runway to take off or land, all of the nodes belonging to this runway are automatically blocked for thirty seconds.

B. Flight Schedule Generator

The flight schedule generator allows the user to define a schedule with respect to the arrival and departure times of aircraft. To do this, the user needs to add a flight, select a start node and an end node. A start node can belong to a runway, which defines it as an arriving flight, or belong to an apron, which defines it as a departing flight. A start time and an end time also have to be set. For an arriving flight the start time is the time that the arriving aircraft leaves the runway and the end time is then the latest time the aircraft should be at the apron. For a departing aircraft the end time is the scheduled take-off time at the runway and the start time is the earliest time the aircraft can depart the apron. Each aircraft can also be given a reduced maximum speed. Aircraft should not be scheduled to land or take off at the same time as this causes unnecessary conflicts.

For larger and more realistic scenarios, which use schedule data as the source, the flight schedule is generated by the airside capacity and delay model of the Airport Business Suite[5] (ABS). The ABS then assigns aircraft to the different runways at realistic time intervals, dependent on the available arrival and departure capacity. AS the schedule created by the ABS contains a departure and/or arrival runway instead of a node number, one or more node numbers need to be assigned per runway. Also Apron / gate nodes have to be assigned to the flights.

The landing time serves as the start time for an arriving aircraft and the departure time as end time for a departing aircraft. A large enough taxi time buffer for the respective runway apron distance needs to be applied to create a respective end or start time and allow different possible routes.

C. Route generator

The route generator searches for possible routes between all the start and end node combinations, which can then be applied for these routes to the flights in the flight schedule.

1. First generation

In the first version of the program all routes are found by using a brute force branch and bound algorithm, which:

- starts searching at a start node and branches for each possible connecting node and does this until
  - there aren't any connections from the current node
  - the current node is already part of the branch
  - the current node is the end node, which makes it a valid route to use from that start node to that end node
- stops when enough routes are found, a number or iterations has taken place after finding the first route or no more nodes can be added to the branch.

When all routes have been found a selection has to be made to which routes, up to a maximum set by the user, should be used. The first route is always the one with the shortest distance but for the alternate routes the difference between the routes is also very important. To avoid getting only very similar routes in more expansive airport models, the next routes are selected on the product of their length with a factor which represents the difference with the previous routes. This factor, which is always lower than one, is multiplied for every previously selected route and is basically the number nodes that are different between each two routes divided by the total number of nodes of both routes. The total process of finding all routes for a model takes a lot of time, especially if models have many connection options between the nodes.

2. Next generation

A variant of Dijkstras[6] algorithm has recently been implemented. This algorithm starts by creating a distance map for a single start node which is then used to get the shortest routes to all set end nodes, as illustrated on the left in Figure 4. As a main feature of TPMagic is that is can process alternative routes, more routes must also be created. As Dijkstra will only return one the shortest path per destination node, the distances for all the links of the found route are then multiplied by a factor of (in this case) two, to force the algorithm to find a route that is different, but will only take reroutes if the alternative part of a route is less than twice as long. When the second or later route is found, the penalty factor for this route is then also applied to the model thus if a part of a route is used in two previous routes, it will only be used again if an alternative is less than four times as long. If a route is not equal to an earlier found route it is stored and used. Figure 5 shows that whilst the first route is the always the shortest, the third route is not necessarily shorter than the second one.

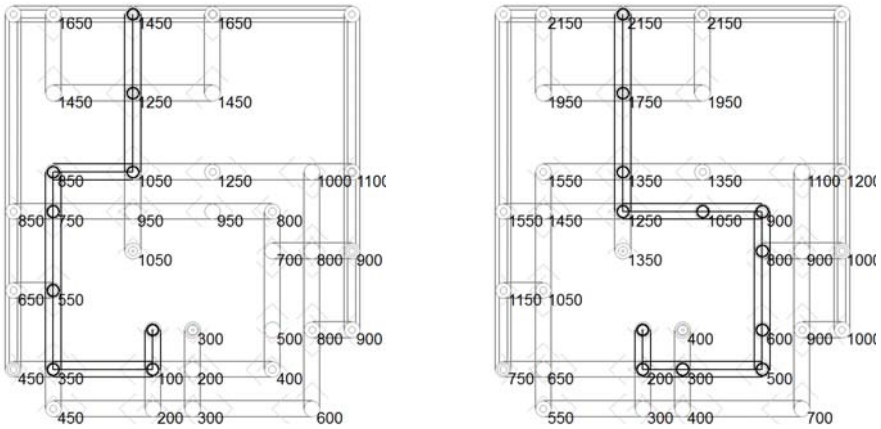

Figure 4. Dykstra distance maps for a first and second route of the test model

The 'new' algorithm is clearly faster. Where the branch and bound algorithm would take about 2 minutes and 45 seconds to find all routes for the normal Atlanta model, the new Dykstra based algorithm will take less than 29 seconds. For the two way situation, finding the routes takes almost 5 minutes and 50 seconds for the branch and bound based algorithm, where the Dykstra based takes only 26 seconds, so even less than the two way system.

There are also some improvements in the routes that are found. As can be seen in Figure 5, the old algorithm could actually find a slightly longer than necessary second route because it had more different nodes.

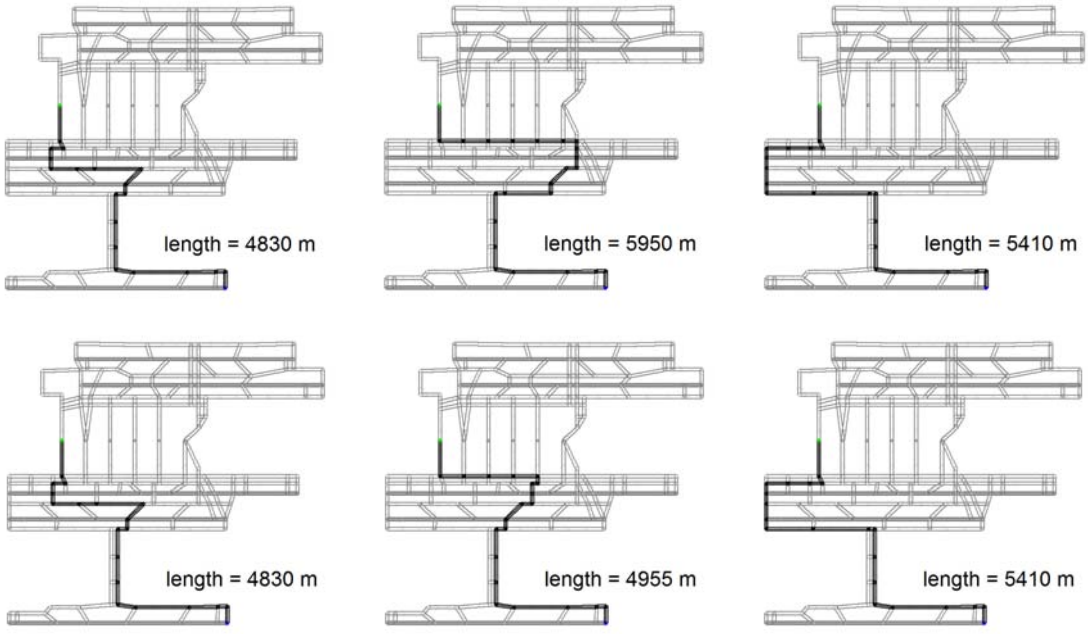

Figure 5. Alternative routes from last exit runway 08 to Gates T for branch and bound algorithm (top) and Dykstra based algorithm (bottom) for the basic Atlanta model

D. MILP Model builder

The MILP model builder converts the routes per flight, as generated by the route generator, to the LP problem for CPLEX. As further explained in previous articles [2, 3], the MILP model consists of an objective function and a number of constraints. The objective function is a function of a cost factor times the unconstrained taxi time for each

aircraft and a cost factor times the delay of each aircraft. The constraints include a node and link occupancy constraints, a route and delay selection constraint and an incremental delay constraint.

The node and link occupancy constraints, which are set per node/link and time interval, block a node or link during the time a flight is using it to prevent other flights from using it at the same time. Both nodes and links must be constrained as constraining only nodes would allow opposing aircraft to cross each other and constraining only links would allow aircraft to pass each other on crossings.

The route and delay selection constraints take care of the route and delay selection by forcing the optimizer to allocate exactly one route and a specific amount of delay per segment of each flight.

The waiting time constraint constrains each incremental segment of a same route and makes the delay at a segment equal or higher than the delay at the segment before it. To block a node whilst an aircraft is incurring delay, also referred to as holding, on that node an additional variable is used.

E. Optimizer

The optimizer is responsible for starting CPLEX with the correct parameters, including a time limit on the duration of the optimization. After CPLEX is started, it checks if the optimization is completed by regularly checking for the existence of the output file. When the output file is detected, the output converter is started.

F. Output convertor

The output convertor reads the output file with the problem solution, which formulates which route and delay per segment is chosen for each flight. When the output file is read, it translates the results back to the routes per flight. The routes per flight are then saved to a database and sent to the traffic simulator.

G. Traffic simulator

The traffic simulator is able to present the routes per flight in time by drawing all aircraft present at their coordinates at every time interval on the airport model. If a conflict is present it will light up the conflicting aircraft. It is also possible to save these images.

Analysis of Hartsfield–Jackson Atlanta International Airport (ATL)

To validate the value A-SMGCS in general and TPMagic specifically, a model shown has been made of the busiest airport in the world: Hartsfield–Jackson Atlanta International Airport (ATL). ATL handled 980 386 movements in 2005[7], more flights than any other airport in the world.

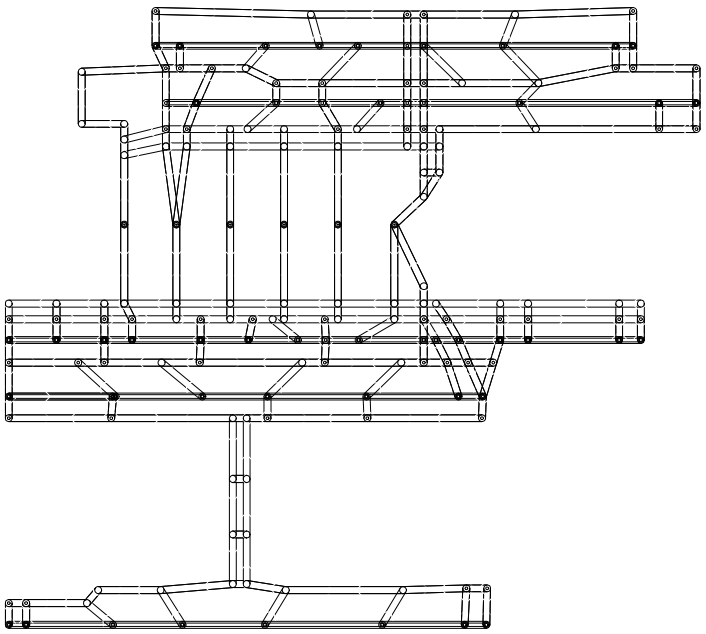

Figure 6. Basic taxi planning model of ATL

The base model of ATL, as shown in Figure 6, has 171 nodes but to limit the link length to a maximum of 150 meters, so they do not limit capacity, the links are split up and the total number of nodes is increased to 315. Where applicable, taxiways are limited to traffic in one direction or two directions.

The six aprons are each modeled by just one node to reduce the need to randomly assign many different gate numbers. In this case flights where randomly assignment to the aprons, but if accurate data with regard to flight-gate combinations is available, this could of course be used to create an even more accurate model. There are 46 start nodes from which arriving aircraft can leave the runway and 24 start nodes from which departing aircraft can enter a runway. This results in 420 different start - end node combinations for the entire model.

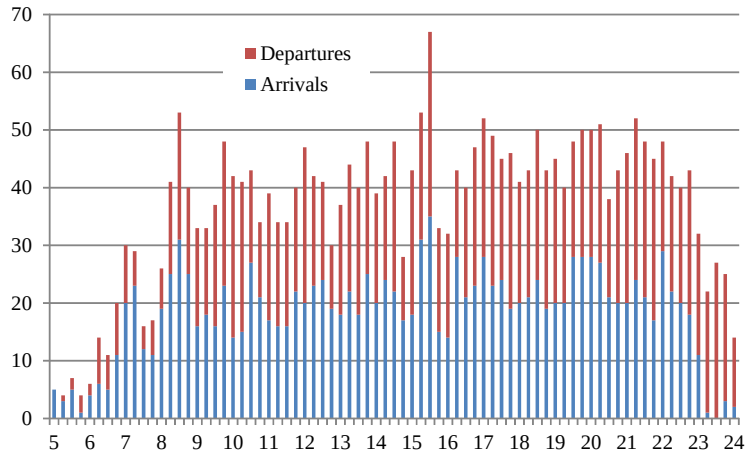

Figure 7. Distribution of flights over the day

The traffic scenario used is produced by the Airport Business Suite[5] (ABS) assuming an east to west flow during the busiest day of 2005, of which the resulting distribution of flights over the day is depicted in Figure 7. As solving a full day (with 2829 flights) is not required and takes too much time to solve, the schedule for is reduced to 05:00 till 09:00. This reduced time is assumed to still be representative for a high traffic density as it contains 341 flights and covers the main morning peak, between 08:00 and 09:00.

The speed for all aircraft is assumed to be (an average of) 12 meters per second (~23 knots) for all taxiways and speed reductions have not been applied, though this is possible. To validate the different settings of TPMagic, the model is run with a number of different settings, which are listed in Table 1.

Table 1. Cases analyzed

	Routes per start - end combination	[Costs of route time] / [cost of waiting time]
Base	Unconstrained	-
FCFS	First come - first serve	-
1, 0.5	1	0.5
2, 0.5	2	0.5
3, 0.5	3	0.5
4, 0.5	4	0.5
5, 0.5	5	0.5
3, 1.0	3	1
3, 0.1	3	0.1

- The base settings runs the simulation with each individual aircraft following its shortest route, without taking into account other aircraft This case has 159 conflicts during the 4 hour period and serves as a reference.
- The first come- first serve (FCFS) case makes aircraft wait whenever they meet another aircraft to eliminate. This is representative for a normal operation.
- The remaining cases are optimized and use different settings for the number of routes per start – end combination and the cost of longer routes compared to the cost of delay. Case '1, 0.5' uses only one route per start – end combination and used a cost of rerouting which is half that of waiting which is used in most cases. Case '3, 1.0' and '3, 0.1' have three routes per start - end combination, just as '3, 0.5', but for '3, 0.1', the cost of delay due to rerouting is one tenth of the cost due to delay where for '3, 1.0' these costs are equal.

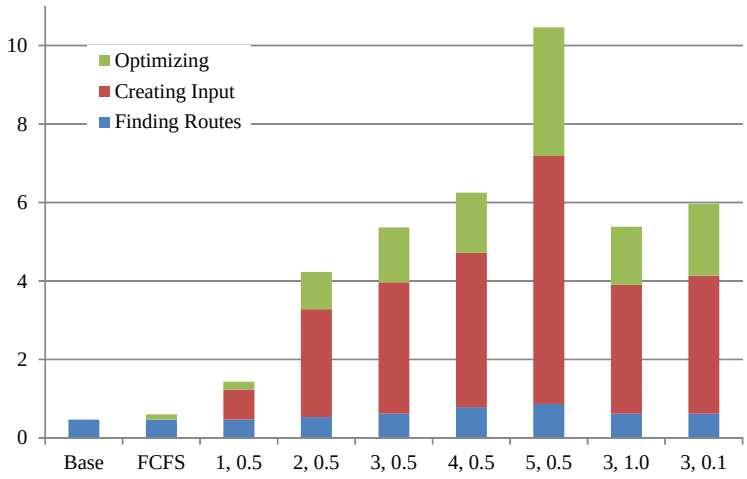

Figure 8. Processing times (minutes)

The main calculation times for the different cases are shown in Figure 8. For the base case, only the calculation of a single route for each start end combination is requires, which takes about 28 seconds. For the FCFS case some calculation is required additionally, which is completed in about 13 seconds. For the optimized cases the optimization time, which includes the time required to process the results, increases from 12 seconds for the case with only one route to 1 minute and 50 seconds for the case with five routes. The cases with 3 routes and different cost settings show very similar calculation for around 1 minute and 30 seconds, which seems very acceptable for a planning of four hours.

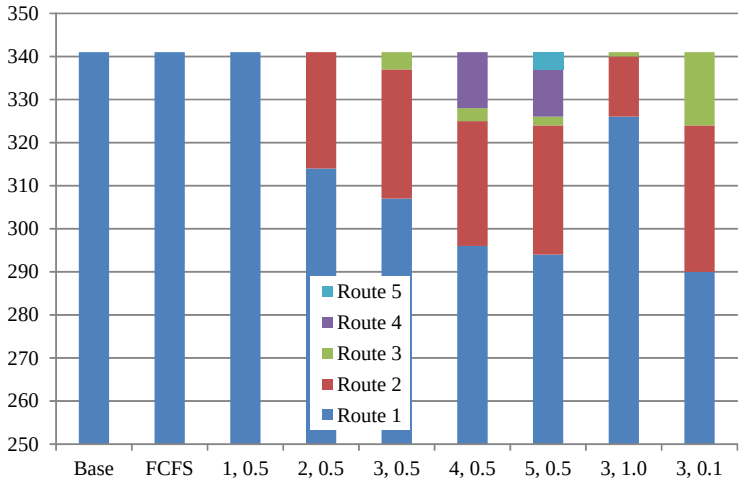

Figure 9. Routes assigned to flights (vertical axis not to scale)

Whilst time required for the optimization is limited, most of the processing time for the optimized cases is actually in writing the LP files of around 10 megabyte, which contains the MILP problem. The writing of these files is currently quite inefficient and could be made much more efficient by optimizing the writing process or even using a directly linked version of CPlex.

The distribution of route numbers for each aircraft is shown in Figure 9. The base case, FCFS and '1, .5' only has one available route per start – end combination and thus only route one can be used. For the cases with alternate routes these are used increasingly with an increased number of routes, though the majority of aircraft still take the shortest route. The increase of reroutes from 4 to 5 routes is so limited, that the extra route does not justify the extra computing time required as shown in Figure 8.

When comparing '3, 0.1', '3, 0.5' and '3, 1.0' it can be seen that the number of rerouted aircraft goes up as the cost of rerouting is relatively lower, which is what we would expect. '3, 1.0' shows that, even is the costs of rerouting and waiting are equal, some aircraft are still rerouted as this effectively saves time overall. Rerouting could this even be useful if overall time is the only objective.

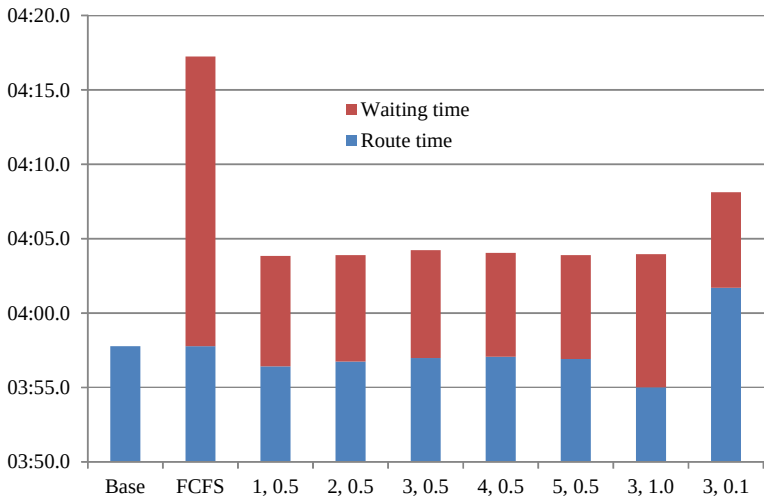

Figure 10. Average route and waiting time in minutes (vertical axis not to scale)

Figure 10 shows the average travel time results for all the aircraft. The Base and FCFS case have an equal route time, the delay required to solve conflicts is almost 20 seconds per aircraft. The route time for the one route case is actually less than for the Base and the FCFS cases, which it should not be. The 1.4 second difference is probably caused by the different way in which they are calculated. Whilst the 1 route case does not allow aircraft to reroute, it does optimize where which aircraft waits, which shows an average decrease in delay of about 12 seconds.

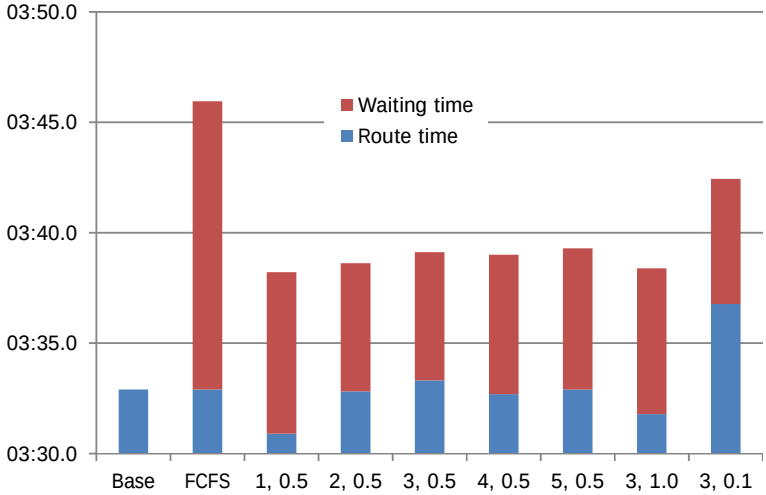

Figure 11. Route and waiting time results for free, two way, flows (vertical axis not to scale)

The differences between the different numbers of available routes are very limited. For the '3, 0.1' case, the delay is reduced significantly, but at a cost of much longer routes. While it is not shown, the maximum delay of this case is only 30 seconds, where it is 60 seconds for all the other cases. For the '3, 1.0' case, the delay is significantly higher, but the routes much shorter. Rerouting does thus not seem to show large average savings in travel time.

A final interesting question is what happens if a free flow system is applied to the model, so all taxiways are changed to be two-way, the result of which are shown in Figure 11. Using alternate routes also seems to have more overall effect on delay and, as can be seen by comparing Figure 11 to Figure 10, the average taxi time drops by about 25 seconds. As this is a reduction in time of about 10%, it could thus mean a significant reduction in fuel consumption and thus costs, and carbon dioxide emissions, which could be obtained in practice by developing and implementing and A-SMGCS systems.

Conclusion

TPMagic is able to model, calculate and optimize taxi times for models of large airports, such as Hartsfield–Jackson Atlanta International Airport (ATL). Whilst the processing times for a given scenario do increase with the number of routes, allowing three or four alternate routes for a time span of a few hours in a single run leads to optimization times of less than two minutes, which should be acceptable.

An optimized planning can result in significant time changes compared to a first come-first serve system, even when aircraft are constrained to a single route. Whilst alternate routes are applied by the optimization routine to up to 10% of all the aircraft to up to 10% of all the aircraft, this does not show significant overall improvements for taxi times and delay at reasonable relative cost values. The overall benefit of rerouting for the ATL case thus seems limited, unless it operates in a free flow system. More airports could be analyzed to see in which way this depends on specific characteristics of airport model.

The analysis of ATL shows that developing an A-SMGCS system on an airport and using all taxiways for all traffic in both directions could cause significant reductions in taxi times and thus also reductions in fuel consumption and carbon dioxide emissions.

References

[1]International Civil Aviation Organization, 'Advanced Surface Movement Guidance and Control Systems (A-SMGCS) Manual', First Edition – 2004, Doc 9830 AN/452

[2]Roling, P. C., 'Airport surface traffic planning optimization: a case study of Amsterdam Airport Schiphol,' *AIAA Aviation Technology, Integration and Operations (ATIO) Conference*, 2009.

[3]P. C. Roling and H. G. Visser, 'Optimal Airport Surface Traffic Planning Using Mixed-Integer Linear Programming', *International Journal of Aerospace Engineering*, Volume 2008 (2008), Article ID 732828

[4]ILOG, CPLEX release 9.1, 2005.

[5]P.C.Roling, 'Evaluation of Anchorage International Airport Using the Airport Business Suite', *26th International Congress of the aeronautical sciences*, 2008, AIAA 2008-8891

[6]Dijkstra, E.W., 'A note on two problems in connexion with graphs', *Numerische Mathematik 1*: p269–p271.

[7]Airport Concil International, *Traffic Movements 2005 FINAL*, http://www.aci.aero