Optimizing AMAN-SMAN-DMAN at Hamburg and Arlanda airport

Dag Kjenstad, Carlo Mannino, Tomas Eric Nordlander, Patrick Schittekat and Morten Smedsrud
SINTEF ICT
Oslo, Norway
Email: name.surname@sintef.no

***Abstract*—Air Traffic Management tries to provide efficient and safe movement of airplanes at and near the airports, a complex task normally divided into Arrival, Surface and a Departure Management Problem. These problems are all tightly connected and should be seen and solved as one. Generally the airports handle them independently, which prevents the needed perspective to ensure good global solutions are made. In this paper we present an integrated approach to the overall problem along with an optimization algorithm that heuristically decomposes the problem so routing, sequencing, and conflicts resolution are carried out in subsequent stages. Our approach has been validated in experiments on Hamburg airport, where we showed remarkable improvements in punctuality and taxi times compared to the expert controllers. Here we also describe new extensions to our model for our upcoming Arlanda airport experiment and interesting future paths to take.**

I. Introduction and Problem description

Air traffic management involves organize and control the flow of traffic on the ground and in the airspace around the airport in a safe and efficient manner. Typically, it considers three distinct problems: The Arrival Management Problem (AMAN), the Surface Management Problem (SMAN) and the Departure Management Problem (DMAN). The AMAN problem involves landing sequencing and ensuring proper separation. The SMAN Problem decides how the arriving and departing airplanes are moving on the airport given the earlier expected landing times and decides take-off times. The DMAN problem decides the take-off times and sequences for departing airplanes. The research literature on surface routing and departure management is already quite large, see recent surveys ([1], [2]).

Even though in principle the three problems are tightly connected and should be seen and solved as one, it is common practice of the airport management to handle them independently and to separate holding areas as buffers between different problems. To handle the complexity manually, the single problems responsibilities are further decomposed so several controllers may be involved in guiding a single airplane along its path. For example, when considering the responsibility of airplanes movement from gate to take-off it includes the Clearance controller (provide flight plan, etc.), Apron controller (give a go-ahead for pushback, etc.), Ground controller (responsible for the taxing), and Runway controller (responsible for airplanes on runway).

While this division of responsibility may be practical for managing the complexity it does prevent the high level of coordination needed to ensure that global optimal decisions are made by each controller. The effect of one controller's decision propagates through to other controllers. E.g. one small valuable adjustment of one controller can very well create havoc for other controllers further down the trajectory. Due to complexity, finding good global solutions manually is too hard. However, optimization techniques can handle such complexity and assist the human controllers in making good global decisions. In this paper we extend the work presented in [6] and introduce a mathematical model which integrates AMAN, DMAN, and the SMAN problem. We solve the model by a heuristic decomposition of the integrated problem where routing, sequencing, and conflicts resolution are carried out in subsequent stages. In particular, first feasible routes for all flights are found. Then the coupled AMAN-DMAN problem is solved *to optimality* by exploiting a time-indexed 0, 1 formulation (see [5]) for the problem. Finally, a complete, conflict-free schedule of all aircraft surface movements is found. The major differences with the approach presented in [6] are the inclusion of arrival windows (against fixed arrival times) in the problem and in the time-indexed formulation for the integrated AMAN-DMAN, some new strengthening inequalities based on some old fixing ideas, and a new, LP based heuristic to solve conflicts among aircraft arising in the use of airport resources, such as runways or taxiways.

We first compared our approach against expert controllers on the Hamburg airport: In these experiments we used three different scenarios that differ in the number of airplanes and in the maximum arrival/departure rate. The air traffic controllers were asked to work with the same constraints and objective as our model to comply with the airport safety rules, aim for punctuality, and minimize taxi time. Our approach shows remarkable improvements in punctuality and taxi times compared to the expert controllers. We are currently setting up new experiments for Arlanda airport, which will be conducted within a couple of weeks. The two airport experiments differ enough that the results cannot be compared but should be seen as complementary to each other. In Hamburg we looked at SMAN-DMAN versus expert controllers and at Arlanda we will compare an existing AMAN and DMAN (working separately) with our AMAN-SMAN-DMAN optimization model. The Hamburg runways were crossed while in Arlanda we work with one runway for both arrivals and departures. Also, the Arlanda experiment will include a temporary closed runway.

The paper is organized as follows: Section 2 provides information about our model and Section 3 describes our solution algorithm. In Section 4 we explain our experimental setup and show our results. We conclude and describe future work in Section 5.

II. The model

We let $F = L \cup D$ be the set of flights to be controlled in the time horizon H, with L be the set of arrival (landing) flights and D be the set of departure flights ($L \cap D = \emptyset$), hereafter denoted as *arrivals* and *departures*. Arrival and departure gates are assumed assigned and cannot be changed. Arrival and departure times may vary within given time windows, depending on the specific flight. Finally, departures may be dropped at very high cost (in practice this corresponds to the need of obtaining a new departure slot and departure window).

For each arrival $l \in L$, λ_l denotes the target landing time, with $\lambda_l \in [\alpha_l, \beta_l]$, the *arrival window* associated with l; also we let ω_l be the *wanted in-block time*, i.e. the time the arrival is expected to reach the stand. Note that the arrival window gets smaller as flights approach the airport, and reduce to a single point when the aircraft is very close. Also, in some applications (like the Hamburg case described later) it is never possible (for the algorithm) to modify the target arrival time. For each departure $d \in D$, we let δ_d be the target departure time with $\delta_d \in [\alpha_d, \beta_d]$ as the *departure window* for $d \in D$. The departure window stems out from real-time negotiations. Finally, we let ω_d be the *target off-block time*, namely the time in which departure d will be ready to leave the gate. In our model, a take-off must happen during the departure window, otherwise the flight is considered *dropped*. Dropped flight will eventually receive a new departure window, but this process is not handled in our model.

Two major types of decisions have to be represented in our model. Firstly, we need to establish a route for each airplane (a spatial decisions). Secondly, the precise timing of all airplane movements through the airport must be determined. The *schedule* comprises all such timings (temporal decisions), including take-off and landing times for departure and arrival flights, respectively.

As usual, the airport is represented by means of a directed graph $G(V, A)$, where arcs correspond to airport segments, and nodes are starting and ending points of segments. The nodes and the arcs of the airport graph are the *airport resources*. For a flight $f \in F$, a *flight route* is simply an alternating sequence of nodes and arcs $r^f = (v_0, a_1, v_1, a_2, \ldots, a_k, v_k)$ with $v_0, v_1, \cdots \in V$ and $a_1, a_2, \cdots \in A$ (a similar representation of vehicle movements may be find in other contests, e.g. trains in railways or metro stations, see [9], [10]). For each flight $f \in F$, let $R(f)$ be the set of *feasible* routes from the initial position of f to its destination; the latter is either a stand for arrival flights, or a take-off point for departure flights. In general, routes can start anywhere on a taxiway, depending where each individual flight happens to be in the start of the control horizon. Any departure (arrival) can enter (exit) the runway(s) from a given set of entry points (exit points), which depends on the aircraft type associated with the flight. While in our experiments (and model) the landing point is fixed for all flights, the take-off point may vary depending on the size and the type of vehicle. Finally, for a given flight $f \in F$ some concatenations of two arcs may be forbidden due to turning restrictions or size limitations.

For every arc $a \in A$ and every flight $f \in F$, we let l_a^f be the running time for f through a. We assume that flights traverse arcs in fixed time, and cannot stop within an arc, so waiting is only allowed at nodes. For all pairs of ordered flights $(f, g) \subseteq F \times F$ and each node $v \in V$ (arc $a \in A$), we let $\tau_{fg}(v)$ ($\tau_{fg}(a)$) be the minimum time separation between f and g when f is running v (a) before g. Let $r^f \in R(f)$ be the route assigned to f. A schedule t is a real vector which associates with every (non-dropped) flight f, and every node v and arc a of r^f, the quantities t_v^f and t_a^f, which are the times when flight f *enters* node v and arc a, respectively. These quantities represent the time the flight starts occupy the corresponding airport resource. The resource is released when the flight enters the next resource on its route. Distinct flights cannot occupy simultaneously the same resource or incompatible resources. A schedule satisfying this condition is called *conflict-free* or simply *feasible*. In contrast, if the condition is violated by a schedule for a pair of flights, then a *conflict* arises.

For a given schedule and flight, we call *taxi time* the duration of its movements from landing to gate (for arrivals) or from gate to take-off (for departures). Since we deal with real-time planning, when considering flights already on their taxiways we will refer to *residual taxi time* to denote the time left from the current position to the destination.

The objective function includes several terms, arranged in a hierarchical fashion: The primary objective is to minimize the number of dropped departures; then the sum of the deviations from the wanted departure and arrival times; finally, taxi times must also be minimized (in order to reduce fuel consumption and maximize passengers' comfort).

III. Solution Algorithm

The ground movements' optimization problem can be summarized as follows:

Problem 3.1: For all arrivals $f \in L$ and all non-dropped departures $f \in D$ find a route in $R(f)$. Find a feasible schedule so that the sum of the dropped flights cost and the schedule cost is minimized.

In order to tackle this problem we decompose it into three solution steps: First, for every flight a feasible (shortest) route is found. Next, optimal (tentative) arrival and departure times are computed. Such times naturally define a sequence for arrivals and departures on the runway(s). Lastly, we generate a conflict-free schedule for the movements of all flights which respects the established sequence. Next, we summarize the three stages:

1. Find for all $f \in F$, the shortest *legal* route r^f in $R(f)$. Let $\mathcal{R} = \{r^f : f \in F\}$ be the set of such routes.
2. Compute a minimum cardinality set $\bar{D}$ of dropped departures; compute tentative arrival and departure times for each non-dropped $f \in F$.
3. Compute a conflict-free schedule for all non-dropped flights respecting runway(s) sequence established at Step 2.

Step 1. Calculating the shortest legal route. Not all paths in the airport graph are feasible for every flight. In particular, some segments may be available only for a subset of flights. Also, turning restrictions exist, and some sequences of arcs may be forbidden for certain aircraft. A route for a flight f that satisfies all its turning restrictions and only uses arcs available for f is called *legal* (for f). In order to quickly find the shortest legal route, in [6] we introduced the *legal line graph* $L(f)$ for $f \in F$, which is a subgraph of the directed line graph of the airport graph with the property that every path in $L(f)$ corresponds to a legal route for f (of equal length). Then, any shortest path algorithm can be applied to the graph $L(f)$ in order to find a shortest legal route for f. Also, in [6] we show that $L(f)$ is not (much) larger than the airport graph.

Step 2. A time-indexed formulation for arrivals and departures time. We present here a 0,1 Linear Program to tackle the optimization problem at Step 2 of our decomposition. We limit the description to the case of single arrival runway and single departure runway (the two runways may coincide, as in Arlanda, or differ, as in Hamburg). The extension to multiple runways is straightforward. First, we discretize the time horizon and let $H = \{1, 2, \dots\}$ be the periods. For all departures $d \in D$ we assume that $\alpha_d, \delta_d, \beta_d, \omega_d \in H$ and we let $H_d = \{\alpha_d, ..., \beta_d\} \subseteq H$ be the feasible departure window, with $\delta_d \in H_d$. Similarly, for an arrival $l \in L$ we assume $\alpha_l, \lambda_l, \beta_l, \omega_l \in H$ and let $H_l = \{\alpha_l, ..., \beta_l\} \subseteq H$ be the feasible arrival window, with $\lambda_l \in H_l$. Finally, when f precedes g, τ_{fg} denotes the separation time between flight f and flight g at landing or take-off, depending whether f is arrival or departure and g is arrival or departure. For any flight $f \in F$, let TX_f be the the minimum taxi time, namely the minimum time necessary to f to run the assigned route r^f (TX_f is computed by using the expected speed of f on the arcs of r^f).

For each departure (arrival) $f \in F$ and each time period in $t \in H_f$ we introduce a binary variable x_{ft} which is 1 if and only if f takes off (lands) at time t. Taking off or landing at time t has a cost c_{ft}. For departure d (arrival l) such cost increases with $|t - \delta_d|$ ($|t - \lambda_l|$). Finally, for each departure d we introduce a binary variable y_d which is 1 if and only if d is dropped. Dropping a departure $d \in D$ has (typically large) cost w_d.

Then the constraints can be written as follows:

- Each arrival must be assigned a landing time:

$$\sum_{t \in H_l} x_{lt} = 1 \qquad l \in L. \tag{1}$$

- Each departure must be assigned a departure time or is dropped.

$$\sum_{t \in H_d} x_{dt} + y_d = 1 \qquad d \in D. \tag{2}$$

- A departure d cannot leave the stand before its off-block time ω_d, so it cannot take-off before $TX_d + \omega_d$:

$$x_{dt} = 0 \qquad d \in D, t \in H_d, t < TX_d + \omega_d \tag{3}$$

  Similar constraints may be written for already taxiing flights.

- Constraints on runway separation between flights become:

$$x_{ft} + \sum_{k \in \{t, \dots, t+\tau_{fg}\}} x_{gk} \leq 1, \qquad t \in H_f, (f, g) \in F \times F, f \neq g. \tag{4}$$

- Finally, the objective is to minimize the cost of dropped flights plus the overall deviation from the wanted arrival and departure times:

$$\sum_{d \in D} w_d y_d + \sum_{f \in F, t \in H_f} c_{ft} x_{ft} \tag{5}$$

An interesting feature of the above program is that it provides a lower bound for the overall optimal cost due to it makes use of the shortest taxi times and neglects potential conflicts on the taxiways which may actually slow down the aircraft. So, if no conflicts arise along taxiways, this solution is actually an optimal feasible solution.

Strengthening the formulation. Here we exploit an idea introduced in [11] and further developed in [4] for pruning a dynamic programming search for the sequencing problem. Consider two distinct departures $f, g \in D$, and assume that they correspond to equivalent flights, which implies that the associated cost functions and all time separations to other flights are identical. The idea is that there is a natural ordering between equivalent flights. So, suppose without loss of generality that $\alpha_f \leq \alpha_g$ and assume $\delta_f \leq \delta_g$. Then it is possible to show that there exists an optimal solution in which f takes off before g and we may neglect all solutions with g taking off before f. This condition can be expressed by the following linear constraint in the x variables:

$$\sum_{k \in \{\alpha_g, \dots, t\}} x_{gk} - \sum_{k \in \{\alpha_f, \dots, t\}} x_{fk} \leq 0 \quad t \in \{\alpha_g, \dots, \beta_g\} \tag{6}$$

It is easy to see that the above constraint cannot be obtained as a conic combination of other constraints of the formulation, which means that it is not implied.

Step 3. Computing a complete, conflict-free schedule. At this stage we need to establish the time in which a flight $f \in F$ should enter every node and arc of its route $r^f = (v_0, a_1, v_1, a_2, \dots, a_k, v_k)$. I.e we need to associate a schedule vector $t^f = (t^f_{v_0}, t^f_{a_1}, t^f_{v_1}, t^f_{a_2}, \dots, t^f_{a_k}, t^f_{v_k})$ with the route of each flight f. The overall schedule t must 1.) Satisfy the order of arrivals and departures on the runway established at Step 3. 2) Satisfy all precedence and separation constraints and 3) Respect landing times and 4) minimize the actual deviation from the wanted departure times plus the additional objectives described earlier. In practice, a linear function $c(t)$ of the schedule well approximates our objective, if necessary including some additional variables to model piece-wise linear terms.

The schedule must satisfy *simple* and *disjunctive* precedence constraints. Simple precedence constraints model fixed precedence relations: for instance, if the arc $a = (u, v)$ belongs to the route r^f of flight f, then $t^f_a \geq t^f_u$ (since f enters arc a *after* it enters node u), and $t^f_v = t^f_a + l^f_a$ where l^f_a is the time

needed by f to cross arc a (note that this equality constraint is equivalent to a pair of simple precedence constraints). Also, since the relative order on the runway is established at Step 2, it can be expressed again by simple precedence constraints of type $t_v^f - t_u^g \geq \tau_{gf}$, where τ_{gf} is a suitable time separation. Each simple precedence constraint is identified by an ordered 4-tuple $p = (f, g, u, w)$ and a constant l_p, where f and g are (not necessarily distinct) flights, and u and w are (not necessarily distinct) airport resources. We denote by $\mathcal{P}$ the set of all 4-tuples corresponding to the simple precedence constraints of our problem.

Finally, when two taxiing aircraft f, g need to access a pair of incompatible resources (or the same resource), a decision on *who goes first* must be taken. Namely, we may assume that *if* g goes first, then the schedule t satisfies the constraint $t_v^f - t_u^g \geq l$, otherwise t satisfies $t_w^g - t_z^f \geq m$. Note that v, u, w, z denote airport resources, which from case to case may all be distinct or repeat. We denote by $\mathcal{C}$ the set of all ordered 6-tuples (f, g, v, u, w, z) associated with the above disjunctive pair of precedence constraints. So, each 6-tuple $c = (f, g, v, u, w, z) \in \mathcal{C}$ corresponds to a potential conflict in the use of resources by the routes of any two flights f and g.

More formally, with every element $c = (f, g, v, u, w, z) \in \mathcal{C}$, we associate two constants l_c and m_c. Let $\bar{t}$ be a feasible schedule and let $c = (f, g, v, u, w, z) \in \mathcal{C}$: Then *either* $\bar{t}$ satisfies the constraint $\bar{t}_v^f - \bar{t}_u^g \geq l_c$ or $\bar{t}$ satisfies the constraint $\bar{t}_w^g - \bar{t}_z^f \geq m_c$ (but not both). The choosing which of the two constraints associated to a conflict $c \in \mathcal{C}$ is to be satisfied by our final schedule is called *solving conflict* c.

Next, we sketch an LP based heuristic to resolve conflicts and find a feasible schedule. For every $c \in \mathcal{C}$, we introduce slack variables δ_c and γ_c. Consider the following linear program:

$$\begin{array}{lll} \min & c(t) + \epsilon\gamma + \epsilon\delta & \\ (i) & t_v^f - t_u^g \geq l_p, & p \in \mathcal{P} \\ (ii) & t_v^f - t_u^g + \delta_c \geq l_c, & c \in \mathcal{C} \\ (iii) & t_w^g - t_z^f + \gamma_c \geq m_c, & c \in \mathcal{C} \\ & t, \delta, \gamma \geq 0 & \end{array} \tag{7}$$

where we let $p = (f, g, v, u)$ and $c = (f, g, v, u, w, z)$, and ϵ is a small constant. Let $\bar{t}, \bar{\delta}, \bar{\gamma}$ be an optimal solution to (7). If $\bar{\delta}_c \bar{\gamma}_c = 0$ for all $c \in \mathcal{C}$ then $\bar{t}$ trivially satisfies all disjunctive constraints and the schedule is feasible[1]. Otherwise, let $\bar{\mathcal{C}} \subseteq \mathcal{C}$ be the non-empty set of violated conflicts, i.e. $\bar{\delta}_c \bar{\gamma}_c > 0$ for all $c \in \bar{\mathcal{C}}$. Choose a "first occurring" conflict in $\bar{\mathcal{C}}$, namely one minimizing the quantity $\min\{\bar{t}_z^f, \bar{t}_u^g\}$. Let $\bar{c} = (\bar{f}, \bar{g}, \bar{v}, \bar{u}, \bar{w}, \bar{z})$ be such conflict. In order to resolve conflict $\bar{c}$ we need to establish which of the two precedence constraints in the disjunction should be satisfied by our schedule and which should be neglected. We are guided in this choice by the solution to (7), which allows us, to select and enforce the constraint which minimizes the increase in the objective function. This corresponds to "pivoting out" the associated slack variable (see [3]). So, if we choose $\bar{g}$ precedes $\bar{f}$, then the constraint $t_{\bar{v}}^{\bar{f}} - t_{\bar{u}}^{\bar{g}} \geq l_{\bar{c}}$ is added to the linear program and constraints (ii) and (iii) associated with $\bar{c}$ are dropped. We have a new LP and we iterate until $\bar{\mathcal{C}}$ is empty[2]. Actually, since pivoting out a variable is equivalent to fixing it to 0, which in turn amounts to adding one constraint to the original LP (in our example $\delta_{\bar{c}} = 0$), this allows us to effectively use the dual simplex method in order to accelerate the re-optimization process in the next LP.

[1]We assume that the natural precedence constraints $t_v^f \geq t_z^f$ and $t_w^g \geq t_u^g$ are included in P

IV. Experiments and Results

In the Hamburg experiment, the algorithm is compared against expert controllers in a realistic simulated environment of the Hamburg airport using the NAVSIM simulator (University of Salzburg) [7]. In three one-hour scenarios, air traffic controllers were obliged to follow all safety rules applicable on the Hamburg airport. They were asked to minimize taxi time and to be as punctual as possible. Afterwards, the algorithm was subjected to the same scenarios, objectives, and safety rules. The three scenarios differs in number of airplanes and in maximum arrival/departure rates. As seen from Table I, the algorithm decrease in average taxi time is between 33% and 36% while punctuality improves by 57% to 67% (the average taxi time and punctuality are measured in seconds.). These results are statistically significant: We have conducted a pairwise one-tail t-test for each scenario. A t-test's p-value lower than 0.05 signifies that the algorithm performed significantly better than the controllers. In this case, all calculated p-values are significantly lower than 0.05.

TABLE I. Results

Objective	Controllers (s)	Algorithm (s)	p-value	$\Delta(\%)$
Avg. taxi time	283.02	183.52	$< .0001$	-35.2
Avg. punctuality	243.32	80.59	0.0105	-66.9
Avg. taxi time	299.15	199.32	$< .0001$	-33.4
Avg. punctuality	190.17	80.44	0.0003	-57.7
Avg. taxi time	298.93	199.79	$< .0001$	-33.2
Avg. punctuality	144.30	59.74	0.0052	-58.6

Note, we needed to simplify our model to represent the real-time NAVSIM simulation environment — NAVSIM simulation model include accelerations and real-time deviations while we adopted constant speeds but used lower value than the average speed of NAVSIM simulation. Also, we assumed the target off-block times are accurate and known in advance. Our integration of SMAN-DMAN allows for more global coordinated decisions. For the experiments we used a laptop with Intel i7 CPU, 4 cores (4x2.7 GHz) and 4 GB RAM. The average run-time was for each scenario was 15s.

We currently prepare extensive experiments on the Arlanda airport that are complementary to our Hamburg experiments. Now we will communicate with NAVSIM simulation environment, DMAN (SAAB), and AMAN (Thales) and in contrast with Hamburg experiment, the controllers will use our optimization technology through the AMAN/DMAN user interfaces. Moreover, in Arlanda we will work with one runway, used for both arrivals and departures. This runway will also be temporary closed to investigate the optimization algorithm behavior. Also, this time, not everything is known

[2]One can show that the process converge under mild assumptions. Some care must also be taken in order to avoid to get trapped into infeasible solutions

up-front to test our approach ability to maintain stable solutions in a dynamic environment. E.g. the target times will become gradually known or can be modified in real-life. Actual off-block and landing will be differing from planned ones to ensure that the algorithms need to react quickly on any deviation to plan. Also, our speed model has been improved—the airplane speed is adjusted depending on the curve radius. Preliminary results on these experiments will be ready by mid November.

V. Conclusions and Future Work

The Arrival, Surface, and Departure Management Problem are tightly connected and should be seen and solved as one. It is common practice of the airports to handle them separately, which prevents the needed global perspective to ensure good coordinated decisions are made. Our mathematical model integrates the three problems and our algorithm decomposes the problem where routing, sequencing, and conflicts resolution are carried out in subsequent stages. Our optimization approach on departure management and surface routing has been validated in experiment on Hamburg airport, where we showed remarkable improvements in punctuality and taxi times compared to the expert controllers. We describe our model extensions for the upcoming Arlanda airport experiment. The two airport experiments are complementary to each other. Also, we are currently working at an extension of our decomposition approach, so that the three stages will be run iteratively exchanging information.

Acknowledgment

This work is co-financed by EUROCONTROL acting on behalf of the SESAR Joint Undertaking (the SJU) and the EUROPEAN UNION as part of Work Package E in the SESAR Programme. Opinions in this work reflect the authors views only, and EUROCONTROL andor the SJU shall not be considered liable for them or for any use that may be made of the information contained herein. All authors contributed equally to the paper. The authors are particularly grateful to Amela Karahasanović and Aslak Wegner Eide for their help.

References

[1] J. A. D. Atkin, *On-line decision support for the take-off runway scheduling at London Heatrow airport*, Ph. D. Thesis, University of Nottingham, 2008.

[2] J.A. Bennekk, M, Mesgarpour, C.N. Potts *Airport Runway Scheduling*, 4OR, 9, pp. 135–138, 2011.

[3] D. Bertsimas, J. N. Tsitsiklis, *Introduction to Linear Programming*, Athena Scientific, Massachussets.

[4] G. De Maere, J. Atkin, and E.K. Burke, *Pruning Rules for Optimal Runway Sequencing*, available at http://www.cs.nott.ac.uk/ gdm/publications/pruningRules.pdf, *sumitted*

[5] M. Dyer and L. Wolsey, *Formulating the single machine sequencing problem with release dates as a mixed integer program*, Discrete Applied Mathematics, no. 26 (2-3), pp. 255-270, 1990.

[6] D. Kjenstad, C. Mannino, P. Schittekat, and M. Smedsrud, Integrated surface and departure management at airports by optimization, *IEEE Xplore Digital Library, Modeling, Simulation and Applied Optimization (ICMSAO), 2013 5th International Conference on*, Hammamet 2013, pp. 1-5.

[7] T. Gräupl, B. Jandl, and C.-H. Rokitansky, *Simple and Efficient Integration of Aeronautical Support Tools for Human-In-the-Loop Evaluations*, in Proceedings ICNS'12, 2012.

[8] J.L. Gross and J. Yellen, *Graph Theory and its applications*, Chapman & Hall, 2006.

[9] L. Lamorgese and C. Mannino, The track formulation for the train dispatching problem, *Electronic Notes in Discrete Mathematics* 41 (2013), 559-566.

[10] C. Mannino, Real-time traffic control in railway systems, *Proceedings of Atmos'11*, A. Caprara and S. Kontogiannis (Eds.), OASICS Vol. 20, 2011.

[11] H.N. Psaraftis, A dynamic programming approach for sequencing groups of identical jobs, Operations Research Vol. 28 (6), pp. 1347–1359, 1980.