Scheduling de-icing vehicles within airport logistics: a heuristic algorithm and performance evaluation

Anna Norin, Di Yuan, Tobias Andersson Granberg and Peter Värbrand

(JORS-10-0253-P)

Scheduling De-icing Vehicles within Airport Logistics: A Heuristic Algorithm and Performance Evaluation

Anna Norin[a,b] [1], Di Yuan[a], Tobias Andersson Granberg[a], and Peter Värbrand[a]

[a]*Department of Science and Technology, Linköping University, SE-601 74 Norrköping, Sweden*

[b]*Swedavia Teknik, Swedish Airports, SE-601 79 Norrköping, Sweden*

Abstract

Most delays in the air transport occur at the airport. A particular reason is the complexity of managing the large number of supporting flows in airport logistics. We consider the optimisation problem of scheduling de-icing vehicles that is one of the key supporting logistic flows in the turn-around process of aircraft. The objective is to minimize the delay of flights due to de-icing, and the travel distance of the de-icing vehicles. We study the complexity of the problem, and develop a solution algorithm using greedy randomized adaptive search. A case study of real-life data from Stockholm Arlanda Airport shows that optimised schedule leads to significantly better performance in comparison to intuitive and simple scheduling strategies. The benefit of optimisation in reducing the waiting time for de-icing is further demonstrated via dynamic simulations.

Keywords

Air transport, Logistics, Multi-objective, Planning, Vehicle Routing, Heuristics

[1] Corresponding author, anna.norin@liu.se

Introduction

The air transport system (ATS) consists in three major components: airline operations, air traffic management (ATM), and airport operations. Airline operations range from aircraft scheduling, crew assignment, to maintenance planning. Examples of ATM are ensuring that there is enough space between aircraft for safety, and deciding which runway a certain aircraft will take off from. Airport operations include all processes occurring at and around the airport, such as deciding which gate a certain flight will be connected to or which fuelling vehicle will serve a certain aircraft. Airport operations involve the management of many flows. There are flows of cargo and passengers with luggage, referred to as *value flows* since those are the flows that generate value to the ATS. To serve the value flows, support flows are needed. These can be divided into *main support flows* (aircraft and crew) and *minor support flows* (cleaning, catering, fuel, baggage handling, de-icing, water and sanitary service).

It is well known that today's airport is a bottleneck in the ATS. The processes and flows at airports together form a very complex system. Many of the processes are critical to the overall performance of ATS. These processes are managed by different actors with contradicting objectives. The overall efficiency of the system is a (complex) function of the individual efficiency of every single actor. The concept of collaborative decision making (CDM) (Euro-CDM, 2010) addresses the issue of lack of communication between the actors, with the objective of better predictability and punctuality. A further step is to maximize the overall efficiency by utilizing the information made available by CDM. To this end, we use airport logistics to refer to the planning and control of all resources and information that create a value for the customers utilizing the airport**.** The customers in this aspect are the passengers and cargo service consumers, as well as airlines, restaurants, shops, and other actors operating at the airport.

Within airport logistics, the efficiency of the flight turn-around process is crucial. The process starts when an aircraft touches down, and ends until the aircraft takes off again. During turn-around, many supporting flows are present; passengers and baggage have to be unloaded, the aircraft has to be cleaned and fuelled, the toilets have to be emptied, and the food supplies re-stocked. In winter season, snow and ice may have to be removed before the aircraft can take off again. To plan and execute the turn-around of one single aircraft does not pose that much of a challenge. However, at most major airports, numerous turn-around processes take place simultaneously. Many of these processes use shared resource, like fuel vehicles, cleaning staff and baggage handlers.

The Gantt charts in Figure 1 illustrate the turn-around of two aircraft, based on real-life data from the SAS airline at Stockholm Arlanda Airport. Even in such a small example, one can see the impact of resource sharing

on turn-around efficiency. In particular, the cleaning staff has chosen to serve Flight 1 before travelling to Flight 2 (the vertical line crossing the activities of the two flights). The passengers of Flight 2 have to wait before they can board, because of the cleaning activity. If the two flights did not need to share the cleaning resource, or if planning of the cleaning activity is done such that Flight 2 receives cleaning earlier, there will be less waiting before boarding can start, and the turn-around time of Flight 2 can be reduced significantly.

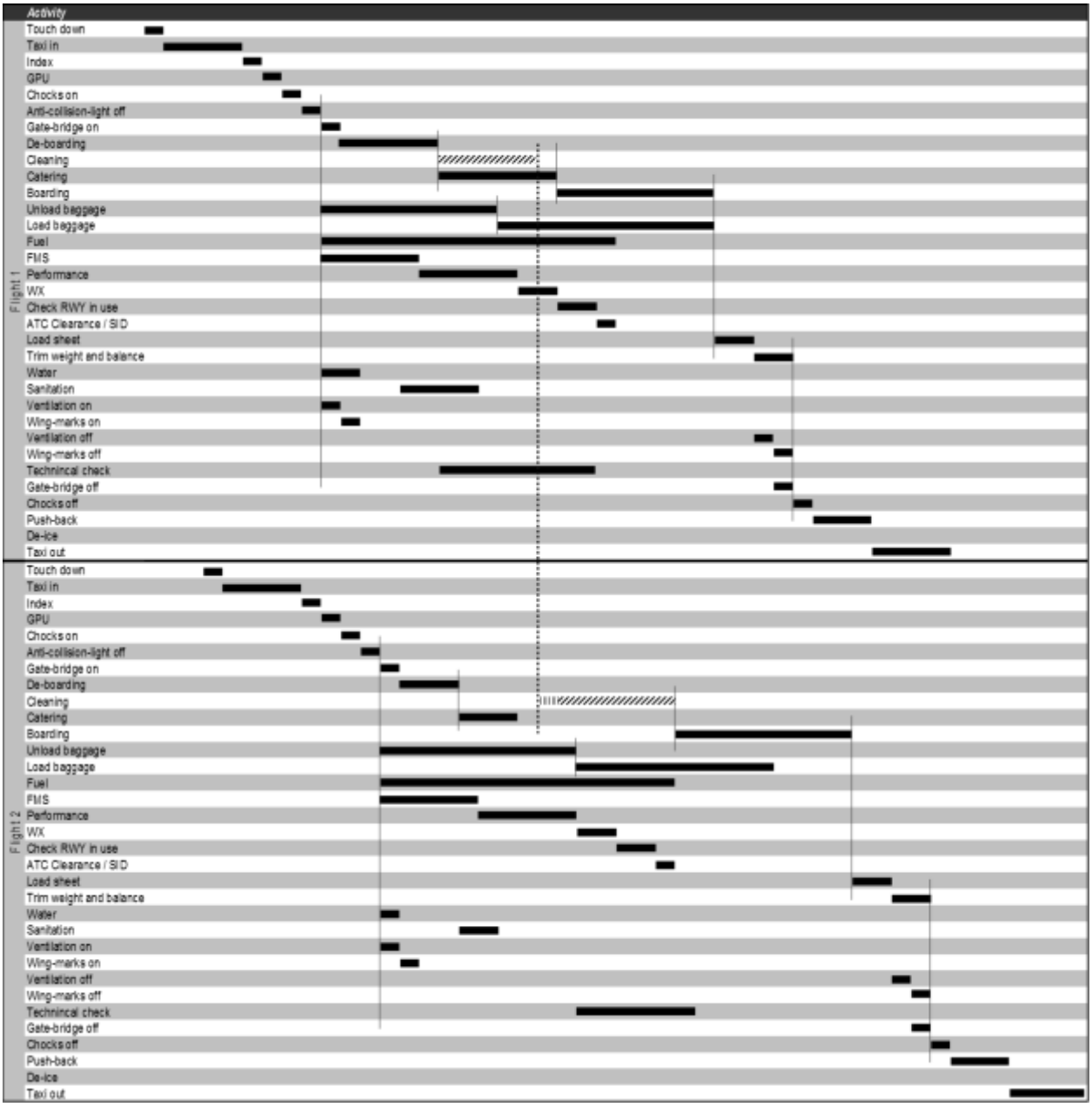

Figure 1 Gantt charts for the turn-around processes of two flights.

The complexity of the planning problem grows rapidly when there are numerous aircraft with overlapping turn-around processes and they have to share a limited amount of resource. In this paper, we investigate the sharing of

the de-icing capacity from an operational research perspective. By acquiring the information of expected de-icing requests via CDM, the planning problem amounts to optimally routing the de-icing vehicles such that the impact of de-icing on turn-around is minimized. A particular reason for investigating the de-icing activity is it contains one additional dimension of complexity in comparison to the other activities in turn-around: De-icing must take place within a certain time slot before takeoff (called hold-over time), i.e., it may not be performed too early (unlike other activities), to ensure that the anti-icing effect is active during takeoff.

In the remainder of the paper, we formalize the optimisation problem of routing de-icing vehicles, develop algorithms for problem solution, and present computational results for a case study of Stockholm Arlanda Airport. The optimised de-icing plan is substantially better in comparison to intuitive and simple scheduling strategies. Moreover, dynamic simulations taking into account all turn-around processes demonstrate that optimised de-icing yields a clear reduction in the overall delay.

Scheduling De-icing Operations

Even a very thin layer of frost and ice has a negative effect on the lifting force and the control of aircraft, and therefore it poses a very serious safety concern in aviation. De-icing must be performed if any part of the aircraft is covered with snow or frost, or there is precipitation and the temperature is close to zero. At Scandinavia airports, the de-icing period is between October and April. The de-icing process consists in two steps. In the first step, frost and ice are removed from the aircraft, usually by a warm, buoyant glycol mix. The second step is the use of a fluid for anti-icing to prevent new ice from building up. Because of the hold-over time, it is not meaningful to de-ice an aircraft a long time before takeoff.

The aim of our study is to approach more efficient de-icing operation by better planning. To this end, we develop optimisation algorithms to route the de-icing vehicles. The two terms routing and scheduling will be used interchangeably throughout the paper, because the routing solution specifies an operational schedule of each vehicle, that is, which aircraft it shall serve, as well as the serving order and times. In addition, the routing includes the visits at the refill station to fill up de-icing fluid.

For performance evaluation, we will carry out a case study based on real-life data at Stockholm Arlanda Airport. An illustration of the airport layout is shown in Figure 2. The triangle shows the depot of the de-icing vehicles. The refill station is marked by the square. Aircraft are located at the gate area along the terminal buildings. This service area is marked grey in the figure. At present, there are three de-icing companies at the airport: SAS

Ground Service (SGS), NorthPort, and Nordic Aero. In this study, they are treated as one single actor. The vehicles are heterogeneous in their capacity of carrying de-icing fluid.

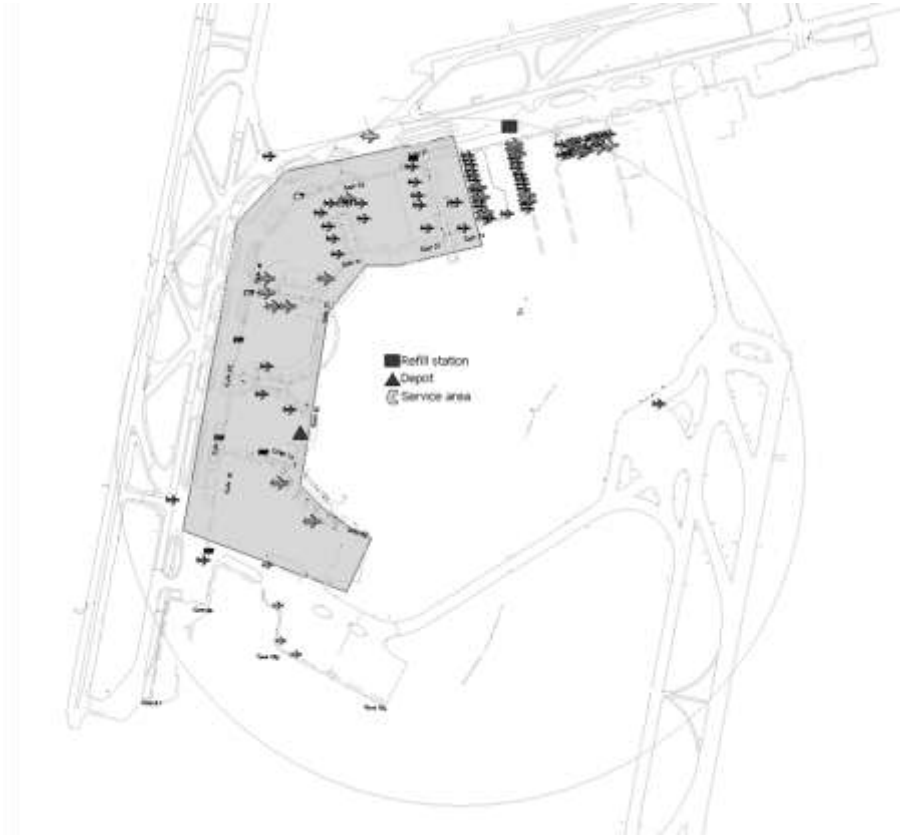

Figure 2 The de-icing facility locations and service area at Stockholm Arlanda Airport

The de-icing requests are given by a time table of the flights. For each flight, the time table specifies the Scheduled Time of Departure (STD) and the assigned gate. Every vehicle starts at the depot to the gate area to perform the scheduled de-icing assignments. There is a fixed set-up time for each de-icing operation. Examples of activities in the set-up phase include adjusting the vehicle position for the aircraft in question (to ensure the whole aircraft can be covered by de-icing fluid), elevating the working platform, and connecting the hosepipe. For the sake of simplicity, the constant setup time is not explicitly used in the models or the solution algorithm in the remainder of the paper, although it is taken into account in the computational experiments as well as the simulations. The time of covering an aircraft by de-icing fluid as well as the fluid consumption vary by aircraft type as well as weather condition (temperature, level of humidity, etc.). Thus some margin in the amount of fluid is needed before a vehicle performs an assignment. For this reason, we define a refill level that is high enough to

accommodate any single assignment in most weather conditions. Every time the amount of de-icing fluid of a vehicle drops below this level, the vehicle returns to the refill station, before proceeding to the next assignment.

We consider two planning objectives. The first is to minimize the total delay that de-icing causes to the scheduled flights. This objective targets the efficiency of the overall turn-around process at the airport. From the de-icing companies' point of view, however, a relevant performance measure is the total travel time, or equivalently, the total travel distance of the de-icing vehicles. These two objectives are combined in our optimisation approach.

The Optimisation Framework

Problem definition

The input data for designing a de-icing schedule consists in several parts. The first part specifies the de-icing demand, including a timetable of the flights, and for each flight the STD, assigned gate, and the average amount of fluid and time required for de-icing the aircraft. Each de-icing operation has also a constant set-up time. The second part of data contains location and distance information, that is, the locations of the depot and refill station, and the distances between these facilities and the gates, as well as the distances between the gates. The last part of the data originates from the de-icing operator, and specifies the number of de-icing vehicles and the capacity of each vehicle.

Minimizing the delay of flights due to de-icing is not necessarily coherent with small travelling distance of the vehicles of the de-icing company. In the optimisation process, we aggregate these two performance metrics into a single objective functions by means of weights. Figure 3 summarizes the notation in problem definition.

K	The set of de-icing vehicles.
N	The set of assignments, i.e., the set of flights requiring de-icing during the planning horizon.
w_{ij}	Distance between the gate of i to that of j, $i, j \in N, i \neq j$.
f_i	Time required by de-icing assignment i, $i \in N$.
STD_i	The scheduled time of departure of the aircraft corresponding to assignment i, $i \in N$.
a	Weight of delay in the objective function.

b	Weight of travelling time in the objective function.

Figure 3 Notation used in defining the de-icing scheduling problem.

A solution to the scheduling problem has the form $R = (R_1, R_2, ...R_{|K|})$, where R_k specifies the sequence of routes taken by vehicle *k*. A route, in its turn, is a sequence of assignments served by the vehicle. The routes start and end at the refill station. An exception is that the first route of each vehicle starts at the depot. The solution is feasible if the total amount of de-icing fluid needed by the assignments of each route does not exceed the capacity of the vehicle serving the route, and if every assignment appears in exactly one of the routes of the vehicles.

The two performance metrics, the travel distance and delay, can be easily calculated for any feasible solution. For vehicle *k*, the total distance, denoted by $d(R_k)$, is simply the sum of the distances between the locations (depot, gates, and the refill station) visited in the order specified by the routes in R_k. The route specification R_k also determines the arrival time to each assignment and thereby the delay, if any, caused by de-icing. Suppose vehicle *k* arrives to assignment *i* at time $t_i(R_k)$. The time the vehicle can start proceeding to the next assignment (or return to the refill station), denoted by $p_i(R_k)$, is calculated as follows.

$$p_i(R_k) = \begin{cases} t_i(R_k) + f_i & \text{if} \quad t_i(R_k) + f_i \geq STD_i \\ STD_i & \text{otherwise} \end{cases}$$

In the first case, vehicle *k* couldn't finish the de-icing operation before the scheduled flight departure time of assignment *i*. The vehicle starts setting up the de-icing operation immediately upon arrival, and $p_i(R_k)$ equals the time point when de-icing finishes. Let $l_i(R_k) = p_i(R_k) - STD_i$ denote the resulting delay. In the second case, the delay $l_i(R_k) = 0$, and the time of moving on from assignment *i*, $p_i(R_k)$, is set to be equal to the flight departure time STD_i, although the vehicle may be able to finish de-icing earlier. This is because de-icing is the last airport logistic activity before taxiing out to the runway. Hence even if the vehicle may leave earlier, the time difference is very short and can be neglected. Note that, in case of no delay, setting the finishing time to STD implicitly takes into account the hold-over time. If a vehicle arrives too early, the finishing time remains STD. In effect, it means that the de-icing operation always respects the hold-over time.

The de-icing vehicle scheduling problem is to find a feasible solution of route sequences $R = (R_1, R_2, ...R_{|K|})$, such that the weighted sum of the distance and delay is minimized, that is,

$$\min \quad c(R) = a\sum_{k\in K}\sum_{i\in R_k} l_i(R_k) + b\sum_{k\in K} d(R_k)$$

We end the problem description by remarking a couple of simplifications made in the optimisation framework. First, the study does not deal with personnel scheduling, which will have to take into account lunch breaks and shift exchanges. As a result, all vehicles can be used continuously without having to go back to the depot during the operation. Second, the refilling time is the same for all vehicles, although the vehicles differ in capacity and the amount of remaining fluid when they arrive at the refill station. For our case study, the impact of this simplification is small because the refilling time, no matter tank capacity, is much shorter than the travel time between any gate and the refill station.

Complexity and Related Work

The structure of the de-icing vehicle scheduling problem resembles the well-known vehicle routing problem (VRP). VRP amounts to constructing minimum-cost routes for vehicles to visit customers from one or multiple depots. Each vehicle has a capacity that limits how many and which customers it can visit. We refer to (Laporte, 1992) for a survey of VRP. A variant of the problem is VRP with time window (VRPTW). In VRPTW, every customer is associated with a time interval within which the customer must be visited. In (Tan et al, 2001), a range of heuristics, including simulated annealing, tabu search and generic algorithms, are investigated for solving VRPTW. A survey and a comparative study of algorithms for VRPTW is provided in (Bräysy and Gendreau, 2005a) and (Bräysy and Gendreau, 2005b). VRPTW with heterogeneous fleet has been considered in (Liu et al, 1999). The authors applied an improvement of the Clarke-Wright cost-saving procedure as the solution algorithm. For the same problem, (Bräysy et al, 2008) presented a multi-start and multi-phase heuristic.

Instead of having a hard time window constraint, VRP with soft time window (VRPSTW) allows late visits but uses a penalty function to discourage delay. VRPSTW with homogeneous fleet was solved using a tabu search algorithm in (Taillard et al, 1997). The objective is a weighted sum of the travelling and delay costs. This objective makes VRPSTW the most closely related problem to our study of scheduling de-icing vehicles.

The work in (Balakrishnan 1993) presented three heuristics based on greedy construction, Clarke-Wright cost saving, and space-time rules, respectively, for VRPSTW with homogeneous fleet, and reported computational results for up to 100 customers. For VRPSTW with heterogeneous fleet, (Calvete et al, 2007) considered a set of objective functions and formulated a multi-objective mixed-integer goal programming model. Penalty is introduced for deviation from each goal. The solution approach consists in generating all possible routes and solving the resulting integer model. The approach scales up to moderate-sized networks.

There is no doubt that the problem of scheduling de-icing vehicles, as specified in the previous section, is NP-hard. A proof can be obtained from a polynomial-time reduction from the VRPSTW in (Taillard et al, 1997). Given a VRPSTW instance, we construct an instance of the de-icing vehicle scheduling problem by putting the depot and the refill station at the same place (i.e., the depot in VRPSTW), and setting the number of vehicles to be equal to the number of assignments. Thus there always exists an optimal solution to the de-icing scheduling instance such that each vehicle takes no more than one route, that is, the construction of the routes becomes independent from each other in time. The distance, demand and capacity parameters in the de-icing instance are the same as in the VRPSTW instance. Then the two instances become equivalent at optimum. As the VRPSTW generalizes VRPTW (Taillard et al, 1997), and it is well-known that VRPTW is NP-hard (see, for example, (Choi and Tcha, 2007) and (Solomon 1987)), the de-icing vehicle scheduling problem is also NP-hard.

The above proof also sheds lights on a major difference between VRPTW and the de-icing vehicle scheduling problem. In the former, the delay occurred in one route does not affect another route. In the problem that we consider, the routes are often coupled in time, i.e., the arrival times to the assignments in a route is a consequence of the routes taken previously by the same de-icing vehicle.

Solution Algorithms

We present four approaches for the de-icing vehicle scheduling problem. The first one is a reference solution, in which the de-icing vehicles are picked in a round-robin fashion. The next two approaches are based on fast greedy-construction algorithms. The two approaches differ in their ways of treating the objective. The last approach is a greedy randomized adaptive search procedure (GRASP).

First come first served (FCFS)

The first-come-first-served schedule is formed by the intuitive strategy of picking the de-icing vehicle in a round-robin fashion. The strategy represents a typical candidate policy for de-icing operations in practice, and serves as a reference solution for performance comparison. The flights are sorted in their STDs. De-icing vehicle k is allocated the assignments at positions $k, |K|+k, 2|K|+k$, etc., of the sorted sequence. Vehicle capacity is accounted for in the same manner as in optimisation, i.e., the vehicles go to the refill station each time the level of fluid falls below the refill level. Whenever the fluid level of a vehicle becomes less than the refill level, the vehicle goes to the refill station, and the travel distances as well as the new time that the vehicle is available for the next assignment are updated accordingly.

Greedy without availability check (GWOAC)

A greedy heuristic is a constructive algorithm that successively builds up a feasible solution by repeatedly selecting the element that gives the smallest cost increase (Michalewicz and Fogel, 2004). The first algorithm, which we refer to as greedy without availability check (GWOAC), let each de-icing assignment be performed by the currently closest located vehicle, without checking whether or not the vehicle will be available in time. In effect, the algorithm attempts to minimize the total travel distance while disregarding the consequential delay (i.e., setting a to 0). The resulting travel distance is a useful reference value from the de-icing operator's viewpoint.

A summary of the algorithm is given in Table 1. We use $L(k), k \in K$ to denote the most recent assignment performed by vehicle $k \in K$. The algorithm initially puts all vehicles at the depot, which is treated as an artificial assignment of index zero, with zero de-icing time and fluid consumption. For each $i \in N$, the algorithm allocates the vehicle with minimum distance to the location of i to perform the assignment. After the allocation, the algorithm updates the location of the selected vehicle, and calculates the time of completion of the assignment. The latter becomes the starting time of travel, when the vehicle is allocated another assignment. The capacity of the vehicles is accounted for in the same manner as in constructing the reference solution.

Table 1 The GWOAC Algorithm

1. Set $L(k) = 0, k \in K$
2. For all $i \in N$:
   a. Select the closest vehicle, i.e. vehicle $k^* = \arg\min_{k \in K} w_{L(k),i}$ to perform assignment *i*.
   b. Once k^* completes its current assignment, move k^* to the location of assignment *i* and update $L(k)$
   c. Update the travel distance, time of accomplishing assignment *i,* and time delay (if any).
   d. If the remaining fluid of k^* is less than the refill level, set $L(k^*) = 0$ and update the time and distance for refilling.

Greedy with availability check (GWAC)

To account for delaying flight because of de-icing, the second greedy algorithm explicitly checks the time availability of the vehicle before allocating an assignment. Consider assignment *j*, and suppose the closest vehicle is located at the gate of assignment *i* (i.e., assignment *i* is the current or most recent assignment of the vehicle). If this vehicle can perform assignment *j* on time, i.e., if $p_i + T(w_{ij}) + f_j \le STD_j$, where $T(w_{ij})$ is the time of travelling distance w_{ij}, then the vehicle is chosen for *j*. Otherwise the second closest vehicle is considered and its time availability is examined, and so on. If no vehicle can perform assignment *j* on time, the vehicle resulting in lowest delay is selected. The GWAC algorithm is presented in Table 2. In addition to notation $L(k)$, we use $P(k)$ to denote the time vehicle *k* can complete its current assignment (or the ending time of refilling if the vehicle goes to the refill station).

Table 2 The GWAC Algorithm

1. Set $L(k) = P(k) = 0, k \in K$
2. For all $i \in N$:
   a. Let $C = \{k \in K : P(k) + T(w_{L(k),j}) + f_j \le STD_j\}$
   b. If $C \neq \phi$, let $k^* = \arg\min_{k \in C} T(w_{L(k),i})$, otherwise let $k^* = \arg\min_{k \in K} P(k) + T(w_{L(k),i})$
   c. Once k^* completes its current assignment, move k^* to the location of assignment *i* ,and update $L(k)$ and $P(k)$

d. Update the travel distance and delay (if any)

e. If the remaining fluid of k^* is less than the refill level, set $L(k^*) = 0$, and update $P(k)$ as well as the distance travelled for refilling

A GRASP-based Search Algorithm

Greedy randomized adaptive search procedure (GRASP) is an iterative search algorithm (Feo and Resende, 1995). One iteration of GRASP consists in two phases. In the first phase, a feasible solution is constructed by greedy selection with randomization. Instead of picking the element with the smallest cost increase as in regular greedy algorithms, GRASP selects randomly an element from a restricted candidate list (RCL). The RCL consists of a number of the greediest elements restricted by either value (quality) or cardinality (quantity). In the former case, all elements having cost values better than a threshold form the list. The latter refers to RCL containing a fixed number of the best elements (Argüello et al, 1997). The second phase of GRASP applies local search to the solution constructed in the first phase.

Our GRASP algorithm bases its RCL on both quality and quantity. For each assignment, the RCL consists of the three most closely located vehicles, if there are at least three vehicles that are able to perform the assignment on time. If this number is less than three but greater than zero, the RCL contains the (one or two) vehicles that can arrive on time. If no such vehicle exists, the list is formed by the three vehicles with earliest possible arrival time. The vehicle selected to perform the assignment is then randomly picked from the RCL. Doing so for all assignments leads to a complete solution. Refilling is accounted for in the same manner as in the greedy algorithms.

Local search seeks for cost improvement by evaluating a neighbourhood formed by solutions that can be obtained by partially modifying the current solution. If a neighbour solution gives better value, the algorithm moves to this new solution and continues examining the new neighbourhood. This is repeated until no improvement is found, i.e., the current solution is a local optimum in respect of the neighbourhood structure.

For the de-icing vehicle scheduling problem, a solution is represented by a sequence of routes for each vehicle. The local search algorithm for the second phase of GRASP uses two types of neighbourhood structure. The first is a two-exchange procedure that takes two assignments allocated to different vehicles, and swaps their positions in the routes of the two vehicles. Suppose for example vehicle A serves assignment 1-2-3 and vehicle B performs assignments 4-5-6. Two-exchange of assignments 2 and 4 amounts to modifying the two routes to 1-4-3 and 2-5-

6, respectively. The distance, time, and delay values are updated accordingly. In our implementation, the assignments subject to swap are constrained to be spaced by no more than 60 minutes in their STDs.

The second neighbourhood is defined by deleting one assignment of a vehicle and reallocating it to another vehicle. A neighbour solution is generated by considering a pair of assignments of different vehicles, and move the first assignment so that it is performed immediately before the second assignment by the second vehicle. For the above example, the pair of assignments 3 and 6 result in route 1-4 for vehicle A and route 2-5-3-6 for vehicle B. Along with the move of the assignment, the times of completing the assignments are updated, and the two performance metrics (distance and delay) are evaluated.

For the swap operation, the algorithm does not change the positions of the next fluid refill in the two vehicles' routes to save computational effort. In the operation of moving an assignment from one vehicle to another, the two vehicles' subsequent refills are moved forward and backward by one step, respectively; that is, the number of consecutive assignments between the refilling operations remains the same as before the move. This leads to a risk of infeasibility, i.e., the amount of fluid is not enough for the updated assignment before refill. Therefore a check is performed, and the generated solution by swap or move is discarded, if it is infeasible in terms of fluid amount.

Phase two ends when none of the two neighbourhoods gives any better solution. The GRASP search algorithm moves then to its next iteration that repeats phases one and two. The algorithm terminates when a pre-defined number of iterations is reached. A description of one algorithm iteration is given in Table 3.

Table 3 One Iteration of the GRASP Algorithm

1. Set $L(k) = P(k) = 0, k \in K$.
2. For all $i \in N$:
   a. Sort all $k \in K$ by $P(k) + T(w_{L(k),i}) + f_i$ in ascending order and let the sorted sequence be Q
   b. Sort all $k \in K$ by $T(w_{L(k),i})$ in ascending order and let the sorted sequence be D
   c. Let $C = \{k \in K : P(k) + T(w_{L(k),j}) + f_i \le STD_i\}$
   d. If $|C| \ge 3$

      Let RCL be the first three elements of D

      else if $|C| \ge 1$

      Let RCL be the first $|C|$ elements of D

      else

Let RCL be the first three elements of Q

e. Take randomly a vehicle k^* in the RCL.

f. Once k^* completes its current assignment, move k^* to the location of assignment *i* , and update $L(k)$ and $P(k)$

g. Update the travel distance and delay (if any)

h. If the remaining fluid of k^* is less than the refill level, set $L(k^*)=0$, and update $P(k)$ as well as the distance travelled for refilling

3. Let $R=(R_1,R_2,...R_{|K|})$ be the resulting routes of all $k \in K$, and $\boldsymbol{c(R)}$ the total cost of R .

4. Repeat

a. Improvement = false

b. For all assignments $i, j \in N, i < j$ and $\left|STD_i - STD_j\right| \leq$ 1 hour:

Let k and l be the two vehicles performing *i* and *j* in R

If $k \neq l$

Swap *i* and *j* to obtain the two new routes R_k' and R_l', and let R' be the new overall solution

If R_k' and R_l' are feasible in the vehicles' capacity and $c(R') < c(R)$

Let $R = R'$ and Improvement = true

c. For all assignments $i, j \in N, i \neq j$ and $\left|STD_i - STD_j\right| \leq$ 1 hour:

Let k and l be the two vehicles performing *i* and *j* in R

If $k \neq l$

Delete assignment *i* from R_k , move all refill operations one step forward, and let the new route be R_k'

Insert assignment *i* before assignment *j* in R_l , move all refill operations one step backward, let the new route be R_l'

If R_k' and R_l' are feasible in the vehicles' capacity and $c(R') < c(R)$

Let $R = R'$ and Improvement = true

Until Improvement = false

It is worth remarking that the neighbourhood structure in the GRASP algorithm can be used for developing other metaheuristics, such as simulated annealing (SA) and tabu search (TS). An implementation of embedding the move and swap operations into an SA framework has been conducted and evaluated. It turns out that, using the same amount of computing time, the solutions generated from the SA algorithm are comparable to but do not outperform those obtained by GRASP.

Computational Results

We have performed a numerical study using real-life data originating from a one-day passenger flight schedule in February 2007 at Stockholm Arlanda Airport. The number of de-icing assignments is 400. The airport has 105 gates. The average distance between the gates is 2438 m. For de-icing, in average 430 litres of fluid is needed per aircraft. The de-icing fleet consists of 18 vehicles with 6 capacity levels. The vehicle type with largest fluid tank can in average de-ice 12 flights before it needs to go to the refill station. The corresponding number for the vehicle with the smallest tank type is 5.

The de-icing time of the aircraft ranges from 1 minute to 18 minutes, and the average value is 7 minutes. The setup time for each de-icing operation is 20 minutes. The weights in the objective function are a=1.0 and b=0.5. As delay and travel distance are of different units, the latter is converted into time using an average driving speed of 30 km/h. In general, there is no single pair of weight values that are universally most appropriate, as they typically represent the interests of two different actors at the airports. The values used here are based on our expertise and aim at giving a reasonable balance between the two performance metrics. The number of iterations in the GRASP algorithm is 4,000. With a Matlab implementation of the algorithm, the solution time is about 6.5 hours on a moderate-speed (2.53 GHz) dual-core notebook, utilizing both cores. The other algorithms run fast as they are construction heuristics that produce one single solution only.

In Table 4 we report the computational results obtained from the two greedy algorithms and the GRASP algorithm, and compare these to the reference solution (FCFS). The GWOAC algorithm leads to a solution with an unacceptable amount of delay in flight departure. The usefulness of the solution lies only in the perspective of the de-icing operator, as it provides a reference value when one attempts to minimize the total amount of travelling of the vehicles. Indeed, GWOAC gives much lower travelling in comparison to the FCFS reference solution. The solution of the GWAC algorithm is clearly better than the reference solution. The improvements in the two objectives are 17.5% and 53.5%, respectively.

Table 4 Computational results

	Travelling time (minutes)	Delay (minutes)	Objective function
FCFS	1 774	1 409	2 297
GWOAC	1 096	438 610	439 158
GWAC	1 463	654	1 386
GRASP	1 127	246	809

In comparison to GWOAC, the GWAC solution requires the vehicles to travel 33% longer. The GRASP algorithm outperforms substantially the reference solution and the greedy algorithms. The improvement over the GWAC solution is 27% and 62% in the two objectives, respectively. The delay value of this solution is smallest among all solutions. In travel distance, the GRASP solution is just slightly higher than that by GWOAC. In the next section, we will use dynamic simulation to evaluate further the FCFS and GRASP solutions. We also remark that, with other values of *a* and *b*, one can obtain solutions with lower travelling distance using the GRASP algorithm. However, since such solutions give much higher flight delays, we do not present them nor study them by simulation.

Results of Dynamic System Simulation

The results presented in the previous sections are all deterministic. To gain further insights, we have developed and used a simulation model of flight turn-around to evaluate the de-icing routing solutions. The purpose is two-fold. The first is to capture system dynamics by using stochastic de-icing time rather than the deterministic one. In addition, the simulation model puts the de-icing activity into a bigger picture to consider its interaction with the other logistic flows.

The simulation model is developed in Arena (ARENA, 2010). The model represents the turn-around process from touchdown and taxiing into the stand, the turn-around activities, until taxiing out to the runway and takeoff. All the main turn-around activities, including unloading and loading baggage, deboarding and boarding, fuelling, cleaning, catering, water, sanitation and de-icing, are accounted for in the simulation. The process times of the various activities are based on triangular distribution. The average values of the distributions are time estimations made by experts at Stockholm Arlanda Airport. For de-icing, the average value has been used in the optimisation earlier. The optimised de-icing vehicle schedules are used as input to the de-icing process in the simulation model. Note that the other logistic processes are not optimised, but follow the simple first-come-first-serve principle.

Simulations have been performed for two de-icing schedules: the reference FCFS solution, and the solution returned by the GRASP algorithm. As the performance indicator, the simulation returns the time that the aircraft have to wait for the de-icing service. This waiting time is defined as the difference between the time point when an aircraft is ready for de-icing and the time when the de-icing actually starts. The simulation has been run 50 times, and the flight waiting times from all the runs are used to obtain the performance figure. The results are summarized in Table 5. The table shows the percentage of flights that have to wait for de-icing, and the maximum and average waiting times. Note that the average values are calculated for the delayed flights, i.e., flights for which de-icing finishes after the STD. Non-delayed flights are not part of this calculation.

Table 5 Simulation results of the de-icing wait time

	Waiting time for De-icing			
	Percentage	Max	Average	Total
FCFS	24%	60 min	14 min	1330 min
GRASP	20%	39 min	10 min	757 min

The optimised de-icing solutions reduce considerably the waiting time for de-icing, in comparison to the reference solution. Comparing the optimised solution to the reference one, the former requires less flights to wait for the de-icing operation. For flights that waited for de-icing, optimisation leads to approximately 40% less waiting time in average. These results demonstrate the benefit of optimizing de-icing to faster turn-around.

Conclusions

In this paper we have considered optimal scheduling of de-icing vehicles and the resulting performance improvement to aircraft turn-around. The study falls into the research domain of airport logistics involving many concurrent logistic processes of which the resource is shared in the turn-around of aircraft. We have studied the complexity of optimal scheduling of de-icing vehicles, and presented heuristic algorithms for problem solution. The computational results show that the GRASP algorithm generates significantly better solutions than those obtained by simple greedy algorithms, both in terms of travel distance of the de-icing vehicles and flight delay.

Dynamic system simulation has provided the additional insight that, though the optimisation approach is deterministic, its benefit to turn-around with stochastically distributed processing times is apparent, even if the other logistic flows in turn-around are not optimised.

One extension of the current work is to partition a planning instance into several parts using time windows, if the scenario at the airport in question permits this decomposition, for the purpose of speeding up the solution process

of optimisation. For the type of problem we consider, including its special case VRPSTW, heuristic algorithms have been considered thus far. In addition to heuristic solutions, having tight lower bounds is highly valuable for assessing the performance of heuristics as well as for telling the potential amount of room for further improvement. To this end, a very interesting topic for further research is the development of lower-bounding procedures. Better and stronger heuristic algorithms form another line of further investigations. Finally, optimisation of the remaining flows in addition to de-icing, and solution integration by means of CDM, which call for additional research, is expected to enable major improvement in resource efficiency at airports.

Acknowledgement

This work is partially sponsored by the LFV group, which is greatly appreciated. We would also like to thank Stockholm Arlanda Airport, Nordic Aero and Scandinavian Airlines (SAS) for supplying us with the input data. We thank the anonymous referees for their valuable comments and suggestions.

References

ARENA (2010). http://www.rockwellautomation.com/, accessed June 1 2010.

Argüello M F, Bard J F and Yu G (1997). A GRASP for aircraft routing in response to groundings and delays. Journal of Combinatorial Optimization 1:211-228.

Balakrishnan N (1993). Simple heuristics for the vehicle routing problem with soft time windows. Journal of the Operational Research Society 44: 279-287.

Bräysy O, Dullaert W, Hasle G, Mester D and Gendreau M (2008). An effective multirestart deterministic annealing metaheuristic for the fleet size and mix vehicle-routing problem with time windows. Transportation Science 42: 371-386.

Bräysy O and Gendreau M (2005a). Vehicle routing problem with time windows, part I: route construction and local search algorithms. Transportation Science 39: 104-118.

Bräysy O and Gendreau M (2005b). Vehicle routing problem with time windows, part II: metaheuristics. Transportation Science 39: 119-139.

Calvete H I, Galé C, Oliveros M-J and Sánchez-Valverde B (2007). A goal programming approach to vehicle routing problems with soft time windows. European Journal of Operational Research 177: 1720-1733.

Choi E and Tcha D.-W (2007). A column generation approach to the heterogeneous fleet vehicle routing problem. Computers and Operations research 34: 2080-2095.

Euro-CDM (2010). http://www.euro-cdm.org/, accessed May 18 2008.

Feo T A and Resende M G C (1995). Greedy randomized adaptive search procedures. Journal of Global Optimization 6: 109-133.

Laporte G (1992). The vehicle routing problem: an overview of exact and approximate algorithms. European Journal of Operational Research 59: 345-358.

Liu F-H and Shen S-Y (1999). The fleet size and mix vehicle routing problem with time windows. Journal of the Operational Research Society 50: 721-732.

Michalewicz Z and Fogel D B (2004). How to Solve it: Modern Heuristics. Springer-Verlag: Berlin.

Nordic Aero (2007). Written communication with Patrik Johansson. Nordic Aero, Stockholm Arlanda Airport, May 14 2007.

Solomon M M (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research 35: 254-265.

Taillard E, Badeau P, Gendreau M, Guertin F and Portvin J-Y (1997). A tabu search heuristic for the vehicle routing problem with soft time windows. Transportation Science 31: 170-186.

Tan K C, Lee L H, Zhu Q L and Ou K (2001). Heuristic methods for vehicle routing problem with time windows. Artificial Intelligence in Engineering 15: 281-295.