ROADEF 2009 Challenge : Use of a Simulated Annealing-based algorithm in Disruption Management for Commercial Aviation

J. Peekstok[1] and E. Kuipers[2]

[1] johan.peekstok@BeImproved.nl
[2] eelcokuipers@yahoo.com

In this paper we present an algorithm based on simulated annealing used for solving a very complex scheduling problem, which is the ROADEF 2009 Challenge problem that is about minimizing cost as a consequence of the disruption in the schedule of a commercial airline. The algorithm presented here was submitted to the ROADEF 2009 Challenge as one of the finalists.

Initial solution

The starting point of the algorithm, the initial solution, is the original schedule as provided in a set instance files provided by the Challenge organization, including the disruption. The inclusion of the disruption usually results in the initial solution being infeasible, and this does occur in all 20 sets of instances files (Challenge sets A en B).

Infeasibilities

Given the initial solution, the choice is to accept infeasibilities during the iteration process of simulated annealing. We preferred anyway, since restricting the local search algorithm to feasible solutions would in our opinion be too restrictive to achieve good results. Feasibilities we allow in our algorithm are :

- Airport departure or arrival rate violations
- Aircraft time transit or turn-around violations
- Aircraft feasible schedule violations (non-matching arrival and departure airport)
- Aircraft maintenance or unavailability schedule violations
- Passenger maximum capacity on aircraft violations
- Passenger connection time violations
- Passenger feasible schedule violations

Ensuring feasibility

In order to handle infeasibilities, we introduced a second objective value, or a value that quantifies the level of infeasibility of the complete solution. During our tests on test-set B we found that finding a feasible solution is in itself not an easy task, let alone then optimizing it. Secondly, we discovered that passenger infeasibilities are not nearly as hard to fix as any of the first 4 (airport and aircraft) infeasibilities. Therefore we decided that once we have achieved airport and aircraft feasibility, we will not allow the current solution to become

infeasible again for airports or aircraft. After achieving airport and aircraft infeasibility we will still allow the algorithm to have passenger infeasibility. In fact, we allow the algorithm to switch back and forth between attempting to lower the value of the objective function and driving the passenger infeasibility level to zero. This way, if no improvement of the objective value can be found without introducing infeasibility, the algorithm can still improve. After a while the cost of infeasibility is slowly increased so that the algorithm is forced to make the solution feasible again.

Neighbors

In building our algorithm we have tested with the following simulated annealing neighbors :
- Canceling an existing flight
- Adding a new flight
- Moving an existing flight forward or backward in time*
- Change the operating aircraft of an existing flight*
- Change the operating aircraft of a pair of existing (consecutive) flights
- Change the operating aircraft of an existing flight and move it in time
- Exchange the operating aircraft of two existing flights*
- Exchange the time of departure of two existing flights
- Change a flight of a passenger's itinerary*
- Remove a flight from a passenger's itinerary*
- Add a flight to a passenger's itinerary*
- Exchange the itineraries of two passengers

* Neighbors active in the algorithm submitted for the ROADEF 2009 Challenge
It would be a mistake to conclude that none of the neighbors not used in our submitted algorithm has any value. We did see some instances respond favorably to several of these neighbors but determining if they are valid for all instances requires further investigation.

Tuning the final algorithm

The set of neighbors in the final algorithm does not contain any means by which flights are removed from the schedule or added to it, but we realized early on that a mechanism is needed that attempts to discover trade-offs between flights and decide which flights are to be cancelled or added. It proved impossible in the time we had to make this a true part of the annealing process so this is done in a constructive way. A rough outline of the submitted algorithm is :
- Phase 1 : Algorithm start with only aircraft neighbors. Passengers remain on their flights in this phase. If airport and aircraft feasibility is achieved move to phase 3. If not, keep trying until a configurable time is spent then move to phase 2.
- Phase 2 : Periodically delete the cheapest flight that causes infeasibility at an airport or aircraft and attempt to reach feasibility. Insert flights to reach maintenance airports if necessary.
- Phase 3 : Airport and aircraft are feasible. Algorithm continues with all 6 neighbors, now allowing passengers to move between the flights. When completely feasible (i.e. passenger feasibility is achieved as well) move to phase 4a.

- Phase 4a : Periodically attempt to add return trip for an aircraft to relieve highest cumulative cancel, delay and downgrade cost for passengers (cumulative per hour, per origin-destination pair). Airport and aircraft are required to stay feasible. If passenger infeasibility occurs, move to phase 4b.
- Phase 4b : Slowly increase cost of infeasibility to drive algorithm back to feasibility. If feasibility is achieved, return to phase 4a.