

Loading aircrafts for military operations

Christelle GUÉRET^a, Narendra JUSSIEN^a, Olivier LHOMME^a

Claire PAVAGEAU^a and Christian PRINS^b

^a École des Mines de Nantes – BP 20722 – F-44307 Nantes Cedex 03, France

^b Université Technologique de Troyes – BP 2060 – F-10010 Troyes Cedex, France
{gueret, jussien, lhomme, cpava}@emn.fr and prins@univ-troyes.fr

1 Introduction

In the military context, force projection means a fast and massive transportation of military equipment (vehicles, troops, ammunition, etc.) from a set of origins to a set of destinations on the battle-field, with the objective of minimizing the total transportation time and the number of required vectors. The most time-critical projections use air transportation.

Force projections are organized by military logisticians at wartime, but for prospective studies the French Délégation Générale pour l'Armement (DGA) may need projection simulations at any time, for instance to evaluate a new aircraft, to dimension a new vehicle, etc. DGA develops computerized tools to tackle these complex and combinatorial operations. Two sub-problems are considered to reduce complexity : an aircraft loading problem (assigning items to be shipped to aircrafts) and a flight scheduling and routing problem.

We present here the loading software we have developed for DGA. The aim of this work was to speed up the tedious computation of loading patterns (typically one day by hand) and to obtain an interactive program, fast enough to allow a human expert to test various scenarios within a short period of time. We detail the solution method which quickly computes an initial heuristic solution and then improves this solution using local search techniques.

2 Problem description

Initially, a set of aircraft types with rectangular bays and a set of *items* to be shipped are given. Each item has an origin (air base), a destination, a weight, and needs a rectangular area when placed in an aircraft bay. Soldiers are special items because they cannot be located anywhere in an aircraft bay : for safety reasons, they must be seated on fixed or removable benches in the bay. Each air base has its own fleet which does not pick items from other bases. Therefore, each air base with its items can be tackled separately. The problem objective is to minimize the total number of sorties of all aircrafts, subject to the following constraints:

- aircraft-related constraints:
 - an aircraft cannot carry items with different destinations,
 - the number of sorties of each type of aircraft is limited by lower and upper bounds,
 - an aircraft can only carry items which are compatible with it.
- item-related constraints:
 - some items must be loaded in the same plane,
 - some items must not be loaded in the same aircraft,
 - items cannot be stacked up in the aircraft.

- some items may be rotated, some others must keep an initial orientation.

The problem can be viewed as a kind of bidimensional bin-packing, with heterogenous bins and many side constraints. It is clearly NP-hard.

3 Fast computation of a first solution

The first step of our method is to quickly compute a feasible solution. We developed two sets of heuristics for that purpose: a list-based heuristic and a loading pattern generation method.

3.1 List heuristic

The list heuristic first sorts the list of items according to a selected criterion. Then it takes each item in the order of the list to place it into a suitable aircraft. Several sorting criteria are possible. Like in the classical FFD (First Fit Decreasing) or BFD (Best Fit Decreasing) bin packing heuristics, it seems wise to load cumbersome items first. Computational testing shows that the best results are obtained by sorting items in decreasing order of their weight \times area value.

Better results are also obtained by loading several aircrafts in parallel, instead of using an on-line packing which loads one aircraft at a time. The initial set of empty aircrafts reflects the composition of the actual fleet and respects the lower and upper bounds on the number of sorties, to avoid that some aircraft types perform too many sorties whereas some others are not used. Each item of the list is loaded into an aircraft in which it can fit. The final aircraft is selected according to a given criterion like: the least loaded aircraft, the aircraft with the least number of items, etc.

At the beginning, we tried a bottom-left strategy to insert an item into a selected aircraft. This heuristic does not relocate the items already in the plane, so it can deny some insertions which would be possible by relocating items. Since the number of items per aircraft is not too large, we developed an exact, constraint-based algorithm which has not the previous drawback: given an aircraft with its current loading of n items, this algorithm decides if a new object can be added. If yes, the algorithm also yields a new loading of $n + 1$ items in which the original items can have different positions. That approach allows an easy modification of the set of constraints to be satisfied by a given loading: it is there where *exotic* constraints can be handled – obligations, exclusions, etc. Of course, deciding whether a new item can be inserted is NP-complete, this is why we reject an attempt of insertion if no solution is found in a fixed amount of CPU time. This works well because *yes* answers are found quickly in practice, due to a huge number of symmetrical solutions.

Troops are relatively easy to place: they need a small area compared to other items and, as already mentioned, putting them on benches is mandatory. In practice we load them after all other items, first on fixed benches, then by adding removable benches.

3.2 Loading pattern generation method

This method comprises two phases: in the first one, many feasible and attractive loadings are generated. The second phase consists in selecting a subset of loadings to cover all items, with the objective of minimizing the number of sorties.

This first phase randomly generates a given number of loading patterns for each aircraft type and each destination: for each aircraft type and each destination, an item to the destination is randomly selected and inserted in the aircraft bay with the loading algorithm described above.

Once enough loading patterns have been generated, we have to determine a subset of patterns covering all items. We developed for this purpose a greedy algorithm based on Chvatal's heuristic for the set covering problem. This heuristic selects at each iteration a loading not yet used, to increase the coverage while optimizing a given criterion. We have adapted this heuristic to our problem to tackle lower and upper bounds on the number of sorties for each aircraft type. The

heuristic is very fast and is run several times by randomly breaking ties at each iteration among candidate aircrafts. Like in the previous heuristic, soldiers are placed at the end.

4 Improvement procedures

The two heuristics developed quickly build a solution. But, due to the upper bounds on the number of aircrafts, it is sometimes impossible to load all items. We apply several improvement procedures to this first solution to load more items, if not all.

A first improvement method starts from the initial list of items. It applies the list heuristic, moves ahead in the list the items which cannot be loaded, and repeats this process a fixed number of times. This is not really a local search, since the cost of successive solutions does not strictly decrease: in practice the best solution found during the runs must be saved.

The other modification made to the solution seeks to make room in an aircraft for items that could not be loaded. The first idea is to create a non saturated aircraft (its upper bound has not been reached yet) and move items from other aircrafts to it. The remaining items can then be tried in the aircrafts in which room has been made.

We also designed a true local search to reduce the number of sorties needed by the final projection. Its neighborhood considers each aircraft in turn and attempts to empty it by transferring all its items to other aircrafts. Even when no aircraft can be saved, this local search is useful to increase the empty space in aircrafts and to insert unloaded items. All these procedures are repeated until no further improvement can be brought to the current solution.

5 Results on real-world data

We tested our methods on real-world instances provided by DGA, with the following characteristics: average number of item types not greater than 20; number of items per destination up to 5000 (about 3 destinations for each air base); average number of aircraft type around 3, and 10 to 60 aircrafts per air base fleet.

In general the results of the two heuristics are very similar to those manually obtained by DGA. For some instances, there exists a significant improvement. In a few cases, the human experts find a better solution, but they never save more than one aircraft compared to the algorithms.

The main advantage is that a solution is available in a few minutes on average (maximum half an hour for the hardest instances) instead of one day by hand, so that DGA engineers can perform many simulations for determining parameters for new vehicles, or new aircrafts ...

6 Conclusion and further work

The two methods developed are both in use now at DGA. Two kinds of improvements are already planned. In the current version of the software, there are two separate modules for loading the aircrafts and for scheduling the flights. The main goal is to integrate, at least to some extent, the two subproblems although the simultaneous resolution seems very hard. The second goal is to allow a more flexible routing, in which a plane from an air base could pick items at other airbases before flying to its destinations.

Acknowledgements

This work was partially supported by the French Délégation Générale pour l'Armement (DGA). We wish to thank Patrick JOURNÉE and Franck OUARY from DGA for their helpful comments.