



Loading aircraft for military operations

C Guéret^{1,2}, N Jussien^{*1}, O Lhomme³, C Pavageau¹ and C Prins⁴

¹École des Mines de Nantes, Nantes, France; ²IRCCyN—Institut de Recherche en Communications et Cybernétique de Nantes, Nantes, France; ³IPogSA, Valbonne, France; and ⁴Université de Troyes, Troyes, France

In this paper, we describe an aircraft loading problem submitted by the French military agency (DGA) as part of a more general military airlift planning problem. It can be viewed as a kind of bi-dimensional bin-packing problem, with heterogeneous bins and several additional constraints. We introduce two-phase methods for solving this NP-hard problem. The first phase consists in building good initial solutions, thanks to two fast algorithms: a list-based heuristic and a loading pattern generation method. Both algorithms call a constraint-based subroutine, able to determine quickly if the items already loaded can be reshuffled to accommodate a new object. The second phase improves these preliminary solutions using local search techniques. Results obtained on real data sets are presented.

Journal of the Operational Research Society (2003) 54, 458–465. doi:10.1057/palgrave.jors.2601551

Keywords: transportation; military; aircraft loading; airlift; bin packing; constraint programming

Introduction

In military context, *force projection* means fast and massive transportation of military equipment (vehicles, troops, ammunitions, etc) from a set of bases to a set of destinations, with the objective of minimizing both the total transportation time and the number of required *vectors* (aircraft, boats, trains, etc). Since force projection is concerned with the ability of military forces to rapidly alert, mobilise, deploy and operate anywhere in the world, it often takes the form of an airlift.

In military airlifts, a set of requirements must be shipped by plane from a set of departure bases to a set of destination bases. Each requirement specifies a load (goods or passengers), ports of embarkation and debarkation, and time windows. Given a set of aircraft with limited capacities, the goal is to move each requirement within the time windows specified, using the aircraft fleet as efficiently as possible. Airlifts either at wartime or for humanitarian operations are usually scheduled by military logisticians.

The French military agency in charge of prospective studies (the *Délégation Générale pour l'Armement*—DGA) may need to simulate airlifts at any time, for example to estimate deployment times, to evaluate a new aircraft or to find compatible dimensions for a new vehicle. DGA is currently developing computerized tools to perform these complex and combinatorial operations. In order to reduce the complexity of the task and also for project management reasons, they decided to split the problem into two sub-problems: an aircraft loading problem (assigning items to be

shipped to planes) and a flight scheduling and routing problem.

The purpose of this paper is to present the software for the first sub-problem (aircraft loading) that we developed under a contract with DGA. Its aim was to accelerate the tedious computation of loading patterns (typically 1 day by hand for an average size problem). The software had to be fast enough to allow a human expert to test various scenarios within a very short period of time.

The paper is organized as follows. The DGA loading problem is first described and enlightened by one example. Then its links with airlift planning and other loading problems are discussed. After this problem background and statement, we present our two-phase solution methods. They consist in two possible heuristics that quickly compute good initial solutions, followed by local search algorithms to improve these preliminary solutions. Results obtained on real data are finally reported.

The DGA airlift problem

Data

Aircraft and items: The basic information for this problem is composed of two sets: a set of aircraft types and a set of items to be shipped. An aircraft type is characterized by its rectangular bay (depth, width and height) and its maximal *payload*. An item is characterized by its dimensions (depth, width, height and weight) and its route (departure air base and destination).

Troops: Troops are considered as special *items* because they cannot be located just anywhere in the aircraft bay. For safety reasons, they must be seated on fixed seats in the

*Correspondence: N. Jussien, Ecole des Mines de Nantes, 4 Rue Alfred Kastler BP 20722, F-44307 Nantes Cedex 3, France.
E-mail: narendra.jussien@emn.fr

cabin or on removable benches in the bay. Some aircrafts do not contain fixed seats nor removable benches. A bench is characterized by its position and the number of *pax* (ie people) it can handle. Its capacity depends on the type of transported troops (eg a paratrooper needs more room than a tank driver because of his/her) backpack).

Air bases: Air bases have their own fleet which is not shared with other departure bases. Therefore, each air base and its related items can be tackled separately. Note that in our problem, neither fuel restriction nor crew regulations have to be taken into account. Indeed, in war time, crew regulations are ignored and fuel *will* be available (so say operational officers).

Constraints and objective function

The aim of the problem is to minimize the total number of sorties for all aircraft, subject to the following constraints:

- Aircraft-related constraints
 - An aircraft delivers only one destination at a time. Thus, items to different destinations need to fly on separate planes.
 - The number of sorties of each aircraft is limited by lower and upper bounds. This constraint is used to perform prospective studies on the composition of the fleet of each base.
 - An aircraft can only carry items that are compatible with it. For example, international regulations (IATA) forbid ammunition on civilian aircraft (Armies may charter civil aircraft.)
- Item-related constraints
 - Items cannot be stacked up in the aircraft. The problem is bi-dimensional. The heights of items and bays are only given to know if an item will fit in a bay.
 - Some items must be loaded in the same plane (eg a tank driver and its tank, otherwise nobody will be able to get the tank out of the plane in the destination base).
 - Some items cannot be loaded in the same aircraft (eg troops and ammunition).
 - Some items may be rotated (eg pallets), others have a unique orientation (eg consider turning a tank inside a bay!).

This problem is characterized by a small number of item types (but there are a lot of each item). It can be viewed as a kind of bi-dimensional (recall that items cannot be stacked up in a bay) bin packing¹ with heterogeneous bins (the aircraft) and many side constraints. It is clearly an NP-hard problem.

An example

Consider the problem in which the items given in Table 1 must be transported from a given base to three destinations A, B and C. Items are described in Table 2. Note that items I2, I3, and I6 must be transported with one soldier. In this example, there are two types of soldiers (S1 and S2), and items I1 and soldiers (S1 and S2) cannot be transported in the same aircraft.

The fleet associated to the selected base is composed of two aircrafts of type A1, four of type A2 and 44 of type A3. The characteristics of these aircraft are given in Table 3. A2 aircraft can only transport items I1 and/or items I5. A1 aircraft contain 185 fixed seats in the cabin and no removable benches. A3 aircraft contain three removable benches:

- two of length 23.10 and width 41.00 are placed on each side of the bay. The capacity is 24 soldiers S1 and 15 soldiers S2 for both.
- one of length 23.10 and width 1.16 is placed in the middle of the bay. Its capacity is 72 soldiers S1 and 50 soldiers S2.

Finally, Table 4 gives the lower and upper bounds of the number of sorties for each aircraft. Note that the global bounds are not necessarily the sum of the destination bounds: other constraints may apply (this is the case for aircraft A3 in the example). In Table 4, when no figure is given, no aircraft of the considered type can transport items to the considered destination.

Table 1 List of items to transport (this is a real example for which, for confidentiality reasons, data have been modified)

	<i>Destination A</i>	<i>Destination B</i>	<i>Destination C</i>
Soldiers S1	122	201	371
Soldiers S2	100	380	0
Item I1	14	27	3
Item I2	14	26	—
Item I3	51	100	—
Item I4	2	—	—
Item I5	32	—	—
Item I6	9	20	—
Item I7	5	—	—

Table 2 Item characteristics

<i>Items</i>	<i>Length</i>	<i>Width</i>	<i>Height</i>	<i>Weight</i>	<i>Spec. Req.</i>
S1	—	—	—	0.135	—
S2	—	—	—	0.250	—
I1	2.64	2.74	2.43	4.00	—
I2	4.50	1.72	2.00	2.50	1 S1
I3	6.45	2.07	2.00	4.50	1 S1
I4	16.10	3.04	2.70	5.00	—
I5	10.00	3.50	2.50	25.00	—
I6	7.80	2.40	2.00	8.00	1 S1
I7	14.60	3.00	3.76	6.00	—

Table 3 Aircraft characteristics

Aircraft	A1	A2	A3
Number in the fleet	2	4	44
Maximum payload	25.00	110.00	24.00
Length of the bay	11.20	15.00	23.10
Width of the bay	3.18	7.00	3.57
Height of the bay	2.44	2.50	3.85

Table 4 Lower and upper bounds of the number of sorties

	Dest. A	Dest. B	Dest. C	Global bounds
A1 aircraft	—	—	3–4	3–4
A2 aircraft	0–8	—	—	0–8
A3 aircraft	38–46	39–47	—	80–93

Background and links with related problems

Position in the global airlift planning problem

Planning an airlift is a very complex task that involves a huge amount of data and combines scheduling, routing and packing decisions. This is why most armed forces recognize today that they need a computer-aided system to assist planning officers. Owing to problem complexity, only heuristics can provide solutions within acceptable running times. Anyway, an optimal solution computed *a priori* would be of limited interest in practice, because last-minute updates are frequent in contingency situations.

The first system implemented and successfully applied is based on the airlift planning heuristic (APH), developed by Rappoport *et al.*^{2,3} for the Military Airlift Command (MAC) of the US Air Force. To simplify the problem, these authors decompose it into two sub-problems: (a) assignment of requirements to planes and (b) mission routing and scheduling (the mission of an aircraft is a sequence of flights from its current base to each on-load base and then to the off-load bases. It corresponds to a *sortie* in the DGA problem).

Sub-problem (a) is described in more detail in another paper by Rappoport *et al.*⁴ As underlined by the authors, the goods composing any requirement vary in size, shape, weight and volume, and can be packed in a plane in many different ways. At the MAC level, data are not always available on the individual items. This is why each requirement is defined by aggregate measures of total weight, volume and footprint (area of floor space), without going down to the item level. Packing the plane (determining the exact layout of items) is left to the air wing conducting the operations.

Rappoport *et al.* describe two assignment methods called *enumerative plane minimization approach* (EPMA) and *hierarchical payload matching procedure* (HPMP). Both methods order the requirements list by giving priority to

the requirements with tight time windows and assign a small subset of requirements at a time.

The first heuristic tested, EPMA, is designed to minimize the number of planes. Each assignment decision tries to fill a plane in terms of weight and prefers the largest plane among those that could be fully utilized. This algorithm may have a bias towards generating higher weight flights at the expense of volume and square feet. The heuristic finally selected, HPMP, gives better results by partitioning each subset according to three cargo types (outsize, oversize and bulk) and by trying to move as much weight, volume and square footage as possible, given the fleet availability.

Compared to Rappoport's study, the problem submitted to us by DGA corresponds to sub-problem (a). The decision to ignore provisionally the mission scheduling problem was imposed by the system specifications. Several differences with the MAC problem must be underlined.

Concerning simplifications, the DGA problem defines one independent sub-problem for each departure base: there is no distinction between the current base and on-load bases because France is small enough to enable to bring requirements by train or truck to each departure base. Moreover, each aircraft delivers one destination at a time: a mission simply is a round trip between the departure base and one destination.

Concerning complications, the DGA problem, yet separated from sub-problem (b), implicitly tackles some scheduling constraints by defining lower and upper bounds on the number of missions for each aircraft. The goal is now to minimize the total number of missions for moving all requirements, a harder objective because of lower and upper bounds. However, the most important complication resides in plane packing, which is now part of system requirements: the input data describe all individual items and the solutions must provide the exact layout of items loaded in each aircraft.

Note that other optimization problems are raised by airlift operations. Solanki and Southworth⁵ describe, for instance, a heuristic to update efficiently an initial operations plan during the execution phase. Hong and Chandra⁶ insist on the staffing aspects, with an algorithm to minimize the number of crews.

Links with other loading problems

Apart from its relations to airlift planning, the DGA problem can be considered as aircraft loading problem (ALP), which belongs in turn to the broad class of cutting and packing problems.⁷ Given a fleet of aircraft and a set of items and persons, the goal of the ALP is to determine a feasible loading plan that minimizes the number of aircraft loads used.

Even the one-dimensional packing problems (knapsack and bin-packing problems) are NP-hard. 2-D and 3-D problems are much more combinatorial and cannot be

solved exactly for large instances. All studies reporting optimal results and/or based on mathematical programming models concern small instances or simplify the problem. For instance, the palletization problems,⁸ yet three-dimensional, are relatively well solved because all items are identical. Several container loading heuristics⁹ build successive vertical or horizontal layers of objects, corresponding to 2-D sub-problems.

The study presented by Vasko *et al*¹⁰ for Bethlehem Steel is a typical 3-D case, where a 0–1 linear program is made possible thanks to problem properties. The problem consists of loading structural steel shapes (like the I-beams used in construction) onto a minimal number of rail cars. The I-beams are cut into order lengths and nested to form drafts. A draft is a stack of nested beams all of the same width, height and length dimensions. The problem becomes bi-dimensional because drafts cannot be stacked on rail cars. The LP formulation is made possible because the draft lengths are such that at most two of them can be put end to end in a car. Moreover, this case is quite rare since the average draft length is well over half the usable car length.

Papers about the ALP assume similar simplifications. As a rule, small items are already packed on pallets or trolleys: the input data contain only pallets, trolleys or cumbersome items that cannot be stacked on board the aircraft. Cochard and Yost¹¹ reduce the problem to one dimension, by building blocks of items as wide as the aircraft bay in a first phase. In a second phase, the blocks may be placed switched or shifted to satisfy gravity center constraints. Larsen and Mikkelsen¹² simplify the problem by dividing the cargo bay into six or 12 compartments and by interactively guiding the heuristics by the load planner.

Chauny *et al*¹³ have recently proposed the first exact approach for the ALP, a column generation technique. However, their problem is still simplified because items are wide enough to forbid side-by-side placements. Thanks to this assumption, these authors reduce the problems to two sequence-dependent knapsack problems (one for the ramp and one for the bay), in order to keep the center of gravity in a given interval.

Even if the items cannot be stacked, the DGA problem is a true 2-D loading problem: its solutions must indicate not only the objects assigned to each aircraft, but also the exact position and orientation of each object in its aircraft. Because of instance size (hundreds or thousands of objects) and to several exotic constraints, the problem is clearly beyond the capabilities of exact approaches.

Fast computation of a first solution

For solving the DGA problem, we developed two-phase solution methods. They consist in two possible heuristics that quickly compute good initial solutions, followed by local search algorithms to improve these preliminary

solutions. We developed two sets of heuristics for the quick computation of initial solutions: a list-based heuristic and a loading pattern generation method.

List-based heuristic

The list heuristic sorts all the items according to a given criterion, and takes them one at a time to place them in a suitable aircraft. Such a heuristic is therefore composed of three phases:

1. sorting the items;
2. selecting for each item a suitable aircraft;
3. placing the items into the aircraft's bay.

Sorting the items: Several sorting criteria come to mind for sorting the items. Like in the classical first fit decreasing (FFD) or best fit decreasing (BFD) bin-packing heuristics, it seems wise to load cumbersome items first. Thus, we tried several orders: decreasing weights, decreasing areas, etc.

Computational experiments have shown that the best results are obtained when sorting the items in decreasing order of their $weight \times area$ value.

The item-related constraint stating that some items need to be transported with others is handled in this phase by using *macro-items*: a *fictional* item that groups together all the items to be shipped in the same aircraft. This macro-item has a weight equal to the sum of the weights of the grouped items. When its dimensions have to be used, the macro-item is automatically expanded into the real items.

Selecting an aircraft for each item: After the sorting phase, the list heuristic selects each item in the ordered list and assigns it to an aircraft. It is possible to either load one aircraft at a time or several ones at the same time (equivalent to *sequential* and *parallel* methods in vehicle routing problems). These two options were tested but the latter gave much better results. The algorithm works as follows:

- The initial set of empty aircraft is given by the lower bounds on the number of sorties.
In our example (see above), the initial set of empty aircraft only contains three aircraft of type A1 to destination C, respectively, 38 and 39 aircraft of type A3 to destinations A and B. We add to this set three aircraft of type A3 without preassigned destination (because the global lower bound on the number of aircraft of this type is 80 whereas the sum of lower bounds for each destination is only $38 + 39 = 77$).
- Once these aircraft are full (or more precisely once the current item in the list does not fit in any of the aircraft), another set of aircraft is added. This set contains $\min(UB_i - n_i, F_i)$ aircraft of type i where UB_i is the global upper bound for aircraft of type A_i , n_i the number of

aircraft of type A_i selected so far and F_i the number of aircraft of this type in the fleet.

In our example, the second set of aircraft considered is composed of $\min(4-3,2)=1$ aircraft of type A1 to destination C, $\min(8-0,4)=4$ aircraft of type A2 to destination A and $\min(93-80,44)=13$ aircraft of type A3, which will be shared between destinations A and B.

- That generation is repeated until all items are loaded or until upper bounds are reached for all aircraft types.

Each item is loaded into one aircraft of the current set in which it can fit. The selected aircraft is chosen according to a given criterion (eg the most loaded aircraft, the aircraft with the most number of items, etc).

Placing items into an aircraft: In order to make sure that an item fits into its aircraft, we developed a *checking* procedure. This is where *exotic* constraints are handled: incompatibility constraints, macro-items, etc. Clearly, fitting a set of n items into an aircraft bay is a difficult problem. We thus chose to focus on effective heuristics to solve that problem. We first tried a bottom-left strategy¹⁴ but this heuristic does not relocate the items already in the bay. Thus, it can deny some insertions which would be possible by relocating items.

Thus, we chose to develop a constraint-based approach for solving that placement problem. Constraint-based programming uses variables with an assigned domain (set of possible values for the variables) upon which constraints are declared.¹⁵ Solving such a set of constraints is done through an enumeration process helped with constraint filtering (setting aside parts of the search space proved to lead to non-solutions).

Once *exotic* constraints are verified, an actual positioning of the items needs to be computed. Our variables will be the position of each item (ie the coordinates of its bottom-left corner in the bay). The aim is to find a set of non-overlapping positions in the bay for the items.

Without loss of generality, we can only consider positions in which items are bottom-left stacked when considering the bay as an oriented plane. Obviously, not all points in the bay correspond to valid item positions. One can only consider positions that are sums of item dimensions.¹⁶ Therefore, two *subset-sum problems* are solved (one for each dimension of the bay: length and width).

A subset-sum problem is a particular case of the 0–1 knapsack problem: given a set $S = \{a_1, a_2, \dots, a_n\}$ of n integers and another integer B (the knapsack capacity), find a subset Y of S whose cumulative sum is maximal without exceeding B .

This problem is binary NP-hard, but can be solved efficiently in practice by a dynamic programming method in $O(n.B)$.¹⁷ This method has an interesting property: for the

same complexity, it obtains as by-products all sum values that can be achieved by subsets of S .

For example, consider the subset-sum problem composed of n lengths of items (without forgetting the width of the items that can turn), with capacity L where L is the maximum length of the bays. Solving this subset-sum problem will provide all the possible x -coordinates.

Possible positions for each item are deduced from those two subset-sum problems by taking an x -coordinate and a y -coordinate in the two answers. After removing non-compatible ones, the remaining set of positions becomes the *domain* of an item.

The search process is quite simple, variables are dynamically ordered regarding the current size of their domain (choosing first highly constrained variables: the well-known *first-fail principle*) and enumerated this way. After each decision, the remaining domains are cleaned by removing all the positions that are covered by placing the related item to its newly assigned position (*forward checking*).

Owing to the huge number of symmetries in such a problem, the resolution is halted after a given number of backtracks if no solution is found. In such a case (after 100 backtracks), the answer of the procedure is *no*.

This is clearly not a complete method, but our experiments compared to the bottom-left heuristic show that our technique:

- answers *yes* whenever bottom-left says *yes*,
- and answers much more often *yes* when bottom left falsely says *no*.

Handling troops: Troops are relatively easy to handle. They need a small area compared to other items and, as already mentioned, have little possibilities for placement; they need to be seated on seats or benches. In practice, we load them after all the other items: first on fixed seats in the cabins, then on removable benches in the bays. In the first case, we do not take their area into account but only their weight. Indeed, if they are placed on fixed seats in the cabin, the only limitations are the number of seats (an information given for each type of aircraft) and the maximum payload of the aircraft. If they are seated on removable benches, in addition to their weights, we take the area of the bench into account.

A loading pattern generation method

This method consists of two phases:

1. generating a great number of feasible and *attractive* loadings;
2. selecting a subset from those loadings to cover all the items to be shipped with the final objective of minimizing the number of sorties.

Generating loadings

The first phase randomly generates a given number of loading patterns for each aircraft type and each destination: items to a destination are randomly selected and inserted in the aircraft bay with the loading algorithm described above until completion of that bay.

For each type of aircraft, the number of generated loadings for each pair (origin, destination) is proportional to the upper bound of that type. Experiments show that $20 \times UB_{id}$ (where UB_{id} is the upper bound of aircraft type A_i for destination d) is enough for good results. Actually, as the number of different items in our problem is quite small, generating more loading patterns leads to several occurrences of the same one.

Like in the list heuristic previously described, troops are loaded after all other items. They are not considered through the generation process.

Selecting a subset of loadings: Once enough loading patterns have been generated, we have to determine a subset of all those patterns *covering* the items to be shipped. We developed for this purpose a greedy algorithm based on Chvatal's heuristic for the set covering problem.¹⁸

In the set covering problem, the data consist of a set of objects V and a collection E of subsets of V . To each set E_i of E is associated a cost c_i . A subset J of E is called a *cover* if each object of V is covered by (ie belongs to) at least one of the subsets in J . The cost of the cover is $\sum(c_i; E_i \in J)$. The problem is to find a cover of minimum cost.

Chvatal's heuristic consists at each iteration in selecting the set E_i having the maximum ratio $|E_i|/c_i$. Then, each object selected in E_i is removed from all the other sets of E . The procedure is repeated until all the objects are covered (ie, all the remaining sets of E are empty).

In the unit cost version (our problem), this heuristic picks up, at each iteration, the set (ie, the loading) that contains the greatest number of items.

That heuristic was meant for problems in which each item is unique. Our problem is a little bit different since several items of the same type have to be transported. Furthermore, there exist lower and upper bounds on the number of aircraft of each type. Thus, we adapted Chvatal's heuristic to tackle these additional constraints:

At the beginning, our heuristic selects at each iteration a loading not yet chosen, among the aircraft for which the lower bound is not yet reached. Once enough aircraft are selected to reach all lower bounds, the heuristics chooses a loading among the aircraft for which the upper bound has not been reached.

Ties among candidate aircraft are broken using a different criterion from Chvatal's heuristic. Indeed, with Chvatal's original criterion, aircraft that contain very small items would be preferred to the ones that contain only one big item. Our experiments showed that early loading of cumbersome items is much more beneficial for the search.

Therefore, we chose to select in priority the aircraft for which the sum of $weight \times area$ value is maximal.

Furthermore, an object type is removed from all remaining loadings only when all the objects of this type are covered, unlike Chvatal's version which performs this removal at each iteration. Our heuristic is very fast and thus can be run several times by randomly selecting a loading among equivalent candidates at each iteration. Troops are handled at the end of the process like in the list heuristic.

Note that once the selection is done, it may happen that the set of selected aircraft carries too many items. A last phase consists in removing from the bay all the items in excess.

Local search algorithms

The two initial heuristics (the list heuristic and the pattern generation method) provide a first solution very quickly. Sometimes, because of the limited fleet and the upper bounds on the number of sorties per aircraft, it is impossible to ship all items or the number of sorties is excessive. We briefly describe hereafter improvement procedures (local searches) for such situations. They are executed in turn until the solution can no longer be improved.

Procedures for loading more items

It is important to note that, in some cases, some items cannot be loaded whereas the upper bound UB_i of an aircraft type A_i is not reached. This is the case when the remaining items are not compatible with this aircraft type. Two improvement algorithms have been designed.

The first algorithm consists in running the list-based heuristic a fixed number of times. After each run, the solution is analyzed and the items that could not be loaded are moved one step towards the beginning of the ordered list. The idea is to increase progressively the priority of such items. This algorithm is not a descent method because the cost of successive solutions may not strictly decrease. Therefore, the best solution found during the runs must be recorded to be returned at the end.

The second algorithm makes room in compatible planes to load remaining items. The principle is to allocate a non-saturated aircraft (ie whose upper bound is not yet reached) and to move items from other planes to it. As soon as there is enough free space in a partly unloaded plane, one of the remaining items is inserted.

Procedure for reducing the number of sorties

We also developed a local search to reduce the number of sorties needed in the final airlift plan. Its neighborhood considers each aircraft load in turn and attempts to empty it by transferring all its items to other planes. Even when no

Table 5 Results on benchmarks—LH (resp. LH w/LS) and LPG (resp. LPG w/LS) give results obtained with our list heuristic and our loading pattern generation method (resp. followed by a local search). (Results in brackets represent the remaining items and pax)

Set	Items	Pax	LH		LPG		LH w/LS		LPG w/LS	
1	1082	0	257	—	250	(1/0)	252	—	248	—
2	1082	0	309	(83/0)	233	(118/0)	296	(28/0)	256	(11/0)
3	1085	4877	272	(72/0)	280	(76/314)	277	(40/0)	279	(59/314)
4	367	1772	69	(0/180)	66	(0/367)	65	—	64	(0/164)
5	368	1405	72	—	66	(1/0)	68	—	64	—
6	368	1405	115	(0/17)	109	(1/416)	110	—	114	(0/416)
7	1658	0	275	(1/0)	268	—	273	—	261	—
8	1700	0	174	—	177	(15/0)	169	—	171	—

aircraft can be saved, this method is useful to increase the empty space in some planes, thus helping the previous procedures in loading more items.

Experiments

We tested our methods on real-world instances provided by DGA. They presented the following characteristics:

- average number of items types no greater than 20;
- number of items per destination up to 5000 (about three destinations for each air base);
- three aircraft types in average;
- A total of 10–60 aircraft per air base fleet.

Human experts from DGA solved these problems but their solutions usually did not verify all the constraints. For example, constraints on pax (handling the benches, etc) are not strictly enforced. Moreover, only the weight is taken into account for pallets, leading to loadings in which no feasible position exists for some pallets. Our results enforce all the declared constraints and thus may lead to apparently worse solutions.

However, in general, the results of our two heuristics are very similar to those manually obtained by DGA. For some instances, there exists a significant improvement. In a few cases, manually obtained solutions are better, but they never save more than one aircraft compared to the automated algorithms (which is negligible). Moreover, remember that human solutions are not feasible since they do not always verify all the given constraints (They are only used to compute a rough evaluation of the optimal solution.).

The two base heuristics (the list-based and the set covering-based) give similar results. Comparing the results before and after the local search show that it is well worth using it. Usually, quite all the items are loaded while even lowering the number of sorties. Table 5 presents our results obtained on benchmarks provided by DGA. The results were obtained in a matter of seconds for each data set.

The main advantage of our set of algorithms is that a solution is available in a matter of seconds (the very worst case took approximatively half an hour to be solved). This is to be compared with the full day requirements of each instance by the human experts. DGA engineers can now perform many simulations by varying the parameters of the projections.

Conclusion

In this paper we have presented two methods for solving a real aircraft loading problem brought to us by DGA. These methods quickly compute an initial heuristic solution before applying local search heuristics. Results obtained on real data sets are satisfactory and the two developed methods are both currently used at DGA.

Our loading pattern generation method suggests that the problem could be solved by a column generation approach (such an approach is proposed in Chauny *et al*¹³ to solve a somehow similar problem). We are currently investigating this idea.

Moreover, two kinds of improvements are planned:

- In the current version of the projection software, loading the aircraft is separated from actually scheduling the flights. It would be interesting to integrate the two processes, but solving simultaneously the two sub-problems seems very hard.
- The second improvement concerns allowing a more flexible routing, in which a plane from a base could pick items at other bases before flying to its destinations (as often happens in real operations).

Acknowledgments—We wish to thank Patrick Journée and Franck Ouary for their helpful comments in writing this paper. This work was partially supported by the French *Délégation Générale pour l'Armement* (DGA) under contract 98 1107A000.

References

- 1 Dyckhoff H, Scheithauer G and Terno J (1997). Cutting and packing. In: Dell'Amico M, Maffioli F and Martello S (eds.).

- Annotated Bibliographies in Combinatorial Optimization*. Wiley: New York.
- 2 Rappoport HK, Levy LS, Golden BL and Toussaint KJ (1992). A planning heuristic for military airlift. *Interfaces* **23**(3): 73–87.
 - 3 Rappoport HK, Levy LS and Toussaint KJ (1994). A transportation problem formulation for the MAC airlift planning problem. *Ann Opns Res* **50**: 505–523.
 - 4 Rappoport HK, Levy LS, Golden BL and Feshbach DS (1991). Estimating loads of aircraft in planning for the military airlift command. *Interfaces* **21**(4): 63–78.
 - 5 Solanki RS and Southworth F (1991). An execution planning algorithm for military airlift. *Interfaces* **21**(4): 121–131.
 - 6 Hong JD and Chandra MJ (1990). A heuristic algorithm to minimize the number of crews allocated in an airlift operation. *Comput Opns Res* **17**: 389–396.
 - 7 Sweeney PE and Paternoster ER (1992). Cutting and packing problems: a categorized, application-oriented research bibliography. *J Opl Res Soc* **43**: 691–706.
 - 8 Gehring H, Menshner K and Meyer M (1990). A computer-based heuristic for packing pooled shipment containers. *Eur J Opl Res* **44**: 277–288.
 - 9 Bischoff EE and Ratcliff MSW (1995). Issues in the development of approaches to container loading. *OMEGA* **23**: 377–390.
 - 10 Vasko FJ, McNamara JA, Newhart DD and Wolf FE (1994). A practical solution to a cargo loading problem at Bethlehem Steel. *J Opl Res Soc* **45**: 1285–1292.
 - 11 Cochard DD and Yost KA (1985). Improving utilization of air force cargo aircraft. *Interfaces* **15**(1): 53–68.
 - 12 Larsen O and Mikkelsen G (1980). An interactive system for the loading of cargo aircraft. *Eur J Opl Res* **4**: 367–373.
 - 13 Chauny F, Ratsirahonana L and Savard G (2000). A model and column generation algorithm for the aircraft loading problem. Report G-2000-68, GERAD.
 - 14 Liu D and Teng H (1999). An improved bl-algorithm for genetic algorithm of the orthogonal packing of rectangles. *Eur J Opl Res* **112**: 413–420.
 - 15 Tsang E (1993). *Foundations of Constraint Satisfaction*. Academic Press: New York.
 - 16 Christofides N and Whitlock C (1977). An algorithm for two-dimensional cutting problems. *Ops Res* **25**: 30–44.
 - 17 Martello S and Toth P (1990). *Knapsack Problems*. Wiley: New York.
 - 18 Chvatal V (1979). A greedy heuristic for the set-covering problem. *Math Opns Res* **4**: 233–235.

*Received April 2001;
accepted January 2002 after one revision*