A Model and Column Generation Algorithm for the Aircraft Loading Problem

Fabien Chauny
GERAD and MQG
École des Hautes Études Commerciales
3000 chemin de la Côte-Sainte-Catherine
Montréal, Qc, Canada, H3T 2A7

Léandre Ratsirahonana
École Polytechnique de Montréal
C.P. 6079, Succursale Centre-ville
Montréal, Qc, Canada, H3C 3A7

Gilles Savard
GERAD and Département de mathématiques et génie industriel
École Polytechnique de Montréal
C.P. 6079, Succursale Centre-ville
Montréal, Qc, Canada, H3C 3A7

December, 2000

Les Cahiers du GERAD
G–2000–68

Abstract

In this paper, we study the aircraft loading problem (ALP) which consists of transporting a series of items between two different points by aircraft at a minimum cost. The problem is formulated as an integer nonlinear multi-commodity network flow model with resource constraints. It is solved by using a column generation approach embedded within a branch-and-bound framework. Lower bounds are provided via a Dantzig-Wolfe decomposition scheme and branching decisions are taken on aggregated flow variables. The algorithm has been tested on real instance and on randomly generated problems. The results show that this approach can solve the ALP to optimality within a reasonable amount of time.

Keywords: Aircraft Loading, Multi-Commodity Network, Column generation.

Résumé

Dans ce rapport, on examine le problème qui consiste à transporter à coût minimum, une série d'items entre deux points, par avion-cargo. Le problème est d'abord modélisé comme un problème de flots non-linéaires multi-commodités avec contraintes de ressources dans un réseau. La solution de ce modèle est obtenue par un algorithme de génération de colonnes emboîtée dans un algorithme de séparation et évaluation progressive. À chaque nœud de l'arbre de branchement, une borne inférieure est obtenue par l'algorithme de décompostion de Dantzig-Wolfe et les décisions de branchement sont prises sur les variables agrégées de flots. L'algorithme a été testé sur des jeux de données réelles et générées aléatoirement. On obtient une solution optimale dans des temps tout à fait raisonnables.

Mots clés: Chargement d'avions-cargos, Flots multi-commodités, Génération de colonnes.

1 Introduction

The aim of this paper is to present a new model for the **Aircraft Loading Problem** (ALP) and an exact algorithm to solve it. This problem arises when there exists a need to transport a wide variety of items and possibly some passengers by aircraft between two given points. Both civilian and military operations are concerned with this problem; the continuous growth in courier activities together with the recent increase of the observed number of deployments (e.g. assistance after natural disasters, peace contingency) make the ALP a recurring problem. The ALP can be defined as follows: given a number of items and persons to be moved by aircraft from one location to another, determine a feasible loading plan that minimizes the total cost consisting of the number of aircraft loads used. A load is feasible if it respects the physical operating constraints of the aircraft; these relate to the total weight of the load, the height of each stacked item in the load and the specific weight and height limits for the ramp location. The center of gravity of the aircraft must also fall within a predefined range of acceptable fuselage stations as near as possible to an optimal location (in terms of fuel efficiency). The challenge is to balance the load fore and aft: in general, balancing the load from side to side is not significant. We will therefore concentrate our effort on the fore and aft dimension. The fact that transportation is carried out between only two points implies that there is no constraint on the sequence of loading and unloading items. In some situations, especially during war time when the items are unloaded in areas where no airport is present, some unloading may be made by parachute at intermediate locations. The sequence used for loading the items must then consider the sites where they will be delivered and the flight path of the aircraft. Additionally, exclusive or pre-order conditions on the items can also be required for security or logistical reasons. A more general description of the problem has been given by Martin-Vega [13].

In this paper, we consider the class of problems faced by the loadmasters of a transport group who have to prepare the aircraft loads, called chalks, for a deployment. This normally means that they have to consider the loading of passengers, large items (like containers, cars, trucks, and trailers), medium items, and small items. The small items are normally already packed onto the large ones while the medium ones are packed on pallets before being loaded on board the aircraft. This last problem, known as the palletization problem (see e.g. Bischoff and Ratcliff [2], Dyckhoff [9] and Gehring, Menshner and Meyer [10]) is difficult in its own right but will not be considered here. Hence we only consider the loading of large items and pallets. For each item to be transported, the loadmasters know the dimensions (height and length) of the item, its weight and its priority which indicates the precedence of loading. The pallets and large items are normally such that they can not be loaded side by side on the aircraft. When there is a need to transport persons, they must be located in the fore section of the aircraft and seats must be installed.

The literature on the ALP is mostly concerned with the construction of load plans which respects all the physical constraints. This construction is sometimes done in an interactive way. This allows the user to suggest good moves without making many computations to verify the feasibility of the constraints. It is also possible for the computer to suggest or complete a plan. Huebner [11] presents the advantages of this type of approach (speed, accuracy, high performance and simplicity) over the manual procedure currently used for

the loading problem. It is the case for the methods proposed by Cochard and Yost [4], Larsen and Mikkelsen [12]. In Cochard and Yost, a *load first–balance after* approach is developed to solve a two-dimensional cutting stock problem which appears when one tries to optimize the utilization of the space inside the aircraft. Blocks of items respecting space constraints are built in the first phase. In the second phase, the blocks may be placed, switched or shifted to satisfy the constraints on the center of gravity. Larsen and Mikkelsen have proposed an interactive procedure that presents load plans to a load planner. The system was developed for a civil airline company. The cargo bay of the aircraft is divided into six or twelve compartments and the cargo is assigned to these compartments according to a set of constraints on weight, balance, specific position for some kinds of item (magnetic, crushable,etc.), and each position must be filled to avoid accumulation of containers at some airports. The number of alterations is first minimized (moving some containers at intermediate airports for loading, unloading and balance constraints), and the total load is next balanced such that the center of gravity is in a feasible region. Two heuristics are proposed, one for constructing plans, and the other for improving them. The load planner override the solutions proposed by the heuristics if other constraints (specific position for different kinds of item, number of passengers, fuel weight) are not satisfied. The procedure is then repeated.

Brosh [3] has examined the case where the loading area is divided into five bays. The problem consists of loading each bay with a quantity of cargo (homogeneous or not) in such a way that the values of the total weight and the center of gravity fall within a predetermined zone. The boundaries of the zone are first linearized, and the nonlinear constraints are then replaced by linear approximations. A sequential linear programming approach is then used until two successive optimal values are sufficiently close.

Amiouny, Barthodi, Van de Vate and Zhang [1] have proposed a heuristic to position the items that are in the same loading plan in such a way that the center of gravity will be as near as possible to an optimal point. The items are first sorted according to density. They are next placed front or aft depending on whether the center of gravity of the loaded items have to be moved forward or backward. They show that the center of gravity computed is at most at $l_{max}/2$ (where l_{max} stands for the longest item) from the target point if such a solution exits; otherwise, it will be as close as possible to the target point.

Ng [14] presents a multicriteria optimization model which considers a predetermined set of feasible load plans. The selected load plans are those that ensure that all the items are loaded, the total number of chalks is minimized and the excess capability's cargo area is maximized. The model is solved by an integer linear programming algorithm. The small number of hand made proposed plans limits considerably the quality of the solution in term of the number of chalks used.

In this article we formulate the problem as a nonlinear multi-commodity network flow model where all constraints on weight, length, ramp, priority and, for the first time, center of gravity, are taken into account. We propose to solve the problem exactly using a column generation scheme embedded within a branch-and-bound algorithm. To our knowledge, this is the first exact algorithm for this problem. Numerical results show that the proposed approach can solve optimally, within an acceptable time frame, a recent deployment of

about one hundred Hercule aircraft of the Air Transport Group of the Canadian National Defence.

The remainder of this article is organized as follows. In the next section we present the model used to describe all the components of the problem. Section 3 describes the algorithm. Numerical results are presented in Section 4 followed by a short conclusion.

2 The Model Formulation

The Aircraft Loading Problem amounts to selecting from all feasible chalks a subset that will minimize total cost, i.e. the number of chalks, in such a way that all items are transported. We formulate this problem as a nonlinear multi-commodity network flow model, where chalks correspond to feasible paths in appropriated constrained networks. A path in a network gives the selected items and the order in which to load them. The commodities correspond to aircraft. This model is solved using Dantzig-Wolfe decomposition, the details of which are presented in the next section.

A set partitioning formulation can also be developed for the ALP. In this formulation, a partitioning constraint is associated with each item and a column to each feasible chalk. An entry a_{ij} of the matrix is equal to 1 if and only if the item i is included in chalk j. However the huge number of feasible chalks prevents an explicit representation of the problem. A column generation scheme is then required to solve the problem. These two formulations are closely related since applying Dantzig-Wolfe decomposition to our nonlinear multi-commodity network flow model leads to a particular column generation scheme for the set partitioning formulation.

2.1 The Underlying Networks

The network $G^k(N^k, A^k)$ corresponding to chalk k, illustrated in figures 1 and 2, is composed of six types of node. The first node O^k is the source node indicating the start of the load. The load is completed at the sink node D^k. Since the loading area is divided into two parts, the bay and the ramp with different physical constraints, there is a *Bay-Ramp* node BR^k which splits the chalk into two parts: the items that go in the bay and those that go on the ramp. The other nodes are grouped in series that represent the relative position on the aircraft, i.e. there is a first series for the first item to put in front of the aircraft, a second series for the second item, and so on (see Figure 1). Since only large items are loaded on the aircraft, the number of series will be small. It is equal to the maximum number of items that can be loaded in any single chalk. Let S be the set of the series, and s an element of this set. Each item that can eventually be loaded in a chalk k, is represented by a node in each series $s \in S$. The items of the same type are grouped together (see Figure 2). In each group, there is a node B^k_{is} that indicates the start of the group of items of type i in series s for commodity k, and another one E^k_{is} for the end of the group. Within the group associated with type i, there is a node I^k_{ijs} for each item $j, j = 1, ..., q_i$, where q_i is the number of items of type i to be transported.

The set of arcs A^k is composed of six types of arcs.

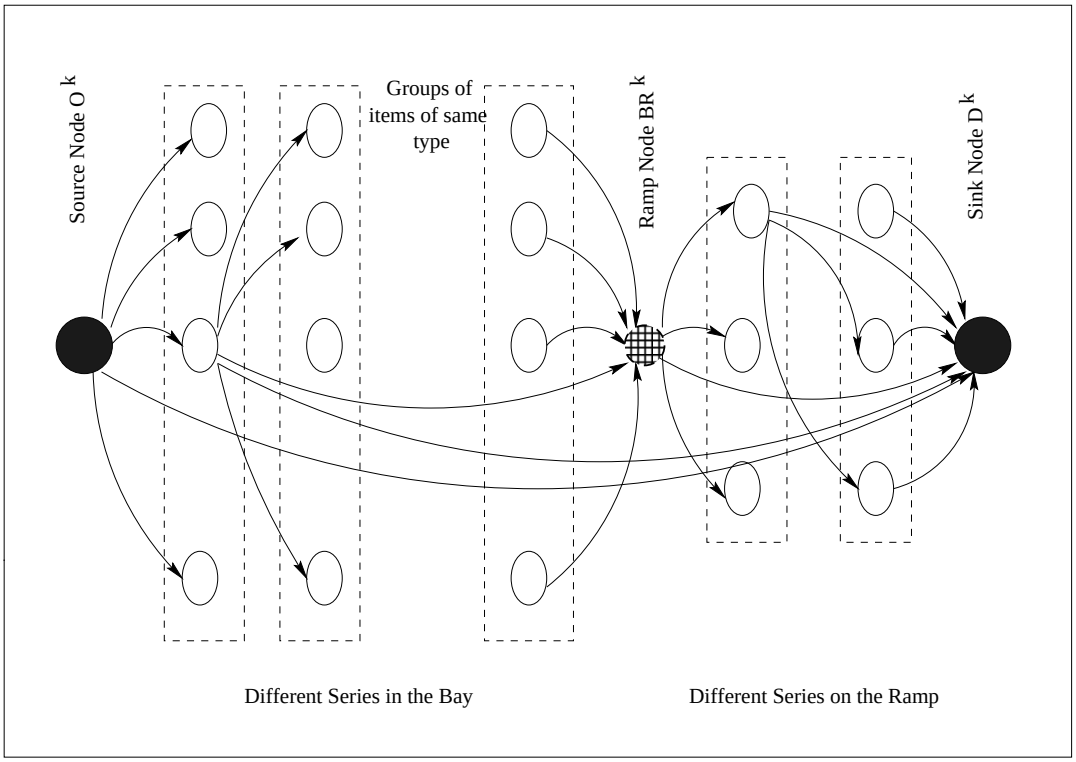

Figure 1: The network for the aircraft loading problem

1. The arcs that select the next type of item to load correspond to three different situations in the network:
   - The arcs $(O^k, B^k_{i1}), i = 1, ..., n$ select the type of item to be put in first position.
   - The arcs $(E^k_{is}, B^k_{js+1}), i, j = 1, ..., n, i \neq j$, select the loading of item of type j in series $s + 1$ immediately after the loading of an item of type i in series s. Here we impose that $i \neq j$ (the case $i = j$ is considered by the third set of arcs).
   - The arcs $(BR^k, B^k_{is}), i = 1, ..., n$ select the first type of item to be loaded on the ramp.
2. The arcs $(B^k_{is}, I^k_{ijs}), j = 1, ..., q_i$, select which item of type i will be loaded first in series s.
3. The arcs $(I^k_{ijs}, I^k_{ij+1s}), j = 1, ..., q_i - 1$, select two items of type i to be loaded consecutively.
4. The arcs $(I^k_{ijs}, E^k_{is}), j = 1, ..., q_i$, indicate that no more item of type i is loaded in series s.
5. The arcs (E^k_{is}, BR^k) indicate that series s is the last in the bay, and the arcs (E^k_{is}, D^k), that series s is the last of the chalk.

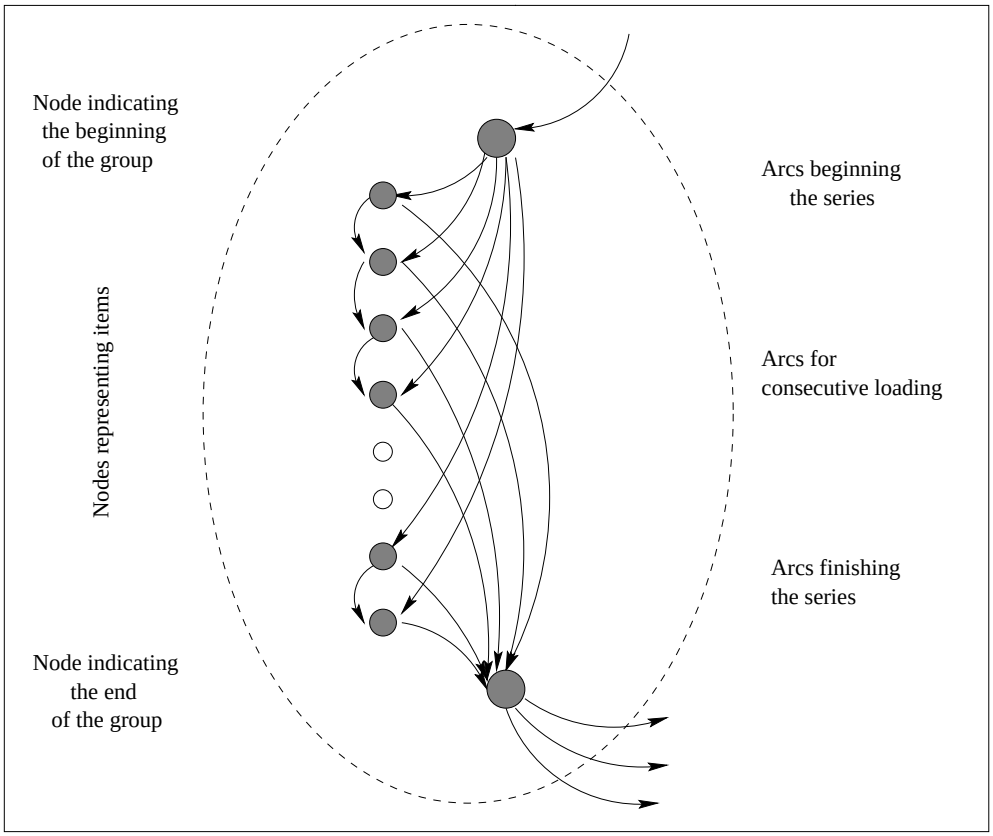

Figure 2: Group of items of the same type

6. The arc (BR^k, D^k) indicates that there is no item on the ramp, and the arc (O^k, D^k) indicates that the aircraft is empty.

A cost of 1 is associated to the arcs $(O^k, B^k_{i1}), i = 1, ..., n$. It corresponds to the creation of a new chalk. The costs on every other arcs is 0.

2.2 The multi-commodity formulation

For item $i, i = 1, ..., n$ we denote by l_i its length, w_i its weight, h_i its height and p_i its priority. A value $\alpha_i \in [0, 1]$ is used to locate the center of gravity (CG) of item i, the center being at position $\alpha_i l_i$. If the item is perfectly balanced, then $\alpha_i = 1/2$. The length of the bay is L_B and the total length of the chalk cannot be larger than L. The maximum value for the weight of the chalk is W and the maximum weight for the items on the ramp is W_R. The center of gravity must lie in the interval $[\underline{g}, \overline{g}]$

A flow variable X^k_a (or X^k_{ij}) is associated with each arc $a = (i, j) \in A^k$. If j is a node corresponding to an item, a flow of one indicates that item j is loaded next in the chalk.

If j corresponds to another type of nodes (e.g. closing the bay, closing the chalk, selecting a new kind of item, etc.), a positive flow indicates that the associated action is selected.

A path from the source O^k to the sink D^k in each network corresponds to a feasible chalk if physical constraints on length, weight, center of gravity, etc., are respected. We model these constraints by the use of resource variables. To each physical constraint, we associate a resource. Let R be the set of all the resources. The resource variable t_j^{kr} represents the amount of resource $r \in R$ accumulated on the path from O^k to j. Let $\mathbf{T}_j^k$ be the $|R|$-dimensioned vector of the resource variables at node j of network G^k. The level of resource r at node j arriving from node i is given by a function $f_{ij}^{kr}(\mathbf{T}_i^k)$ called an *extension function*. At each node the value of t_j^{kr} must be in the interval $[a_j^{kr}, b_j^{kr}]$.

Using the notations $\Gamma(j) = \{(i,j) : i \in N^k\}$ and $\Gamma^{-1}(j) = \{(j,i) : i \in N^k\}$ for the sets of arcs adjacent to the node j, we obtain the ALP model:

$$(ALP) \qquad \min \sum_{k \in K} \sum_{a \in \Gamma^{-1}(O^k) \setminus (O^k, D^k)} X_a^k \tag{1}$$

$$\text{subject to} \quad \sum_{k \in K} \sum_{s \in S} \sum_{a \in \Gamma(I_{ijs}^k)} X_a^k = 1 \quad i = 1, ..., n, j = 1, ..., q_i \tag{2}$$

$$\sum_{a \in \Gamma^{-1}(O^k)} X_a^k = \sum_{a \in \Gamma(D^k)} X_a^k = 1 \quad \forall k \in K \tag{3}$$

$$\sum_{a \in \Gamma(\alpha)} X_a^k - \sum_{a \in \Gamma^{-1}(\alpha)} X_a^k = 0 \quad \forall \alpha \in N^k, \forall k \in K \tag{4}$$

$$X_{ij}^k (f_{ij}^{kr}(\mathbf{T}_i^k) - t_j^{kr}) \leq 0 \quad \forall k \in K, \forall r \in R, \forall (i,j) \in A^k \tag{5}$$

$$a_i^{kr} \leq t_i^{kr} \leq b_i^{kr} \quad \forall k \in K, \forall r \in R^k, \forall i \in V^k \tag{6}$$

$$X_{ij}^k \in \{0,1\} \quad \forall k \in K, \forall (i,j) \in A^k \tag{7}$$

The objective function (1) represents the number of chalks (or aircraft) to be used to transport every items. Constraint (2) imposes that the j^{th} item of type i must be loaded exactly once. Constraints (3) and (4) correspond to the classical flow conservation constraints in the networks. The values of resource variables are updated through constraint (5). Constraints (6) and (7) correspond to bounds on variables.

We now specify the exact resource variables used together with their respective bounds and extension functions corresponding to constraints (5) and (6).

Total length

The first resource variables t_i^{k1} correspond to the total length of items already loaded at node i for chalk k. The extension function f_{ij}^{k1} is therefore:

$$f_{ij}^{k1}(\mathbf{T}_i^k) = \begin{cases} t_i^{k1} + l_j & \text{if } j \text{ corresponds to an item and } i \neq BR^k \\ L_B & \text{if } i = BR^k \\ t_i^{k1} & \text{otherwise} \end{cases} \tag{8}$$

For all nodes j on the left hand side of node BR^k, the value of t_j^{k1} must be less than or equal to the length of the bay. For the nodes on the right hand side, its value must be less than or equal to the length of the aircraft.

Total weight

The second resource variables t_i^{k2} correspond to the total weight of items already loaded at node i for chalk k. The extension function is similar to equation (8), except that the length l_j is substituted by the weight w_j and there is no setup at the ramp node. The corresponding extension function is:

$$f_{ij}^{k2}(\mathbf{T}_i^k) = \begin{cases} t_i^{k2} + w_j & \text{if } j \text{ corresponds to an item} \\ t_i^{k2} & \text{otherwise.} \end{cases} \tag{9}$$

The value of t_j^{k2} must be less than or equal to the maximum allowable weight of a chalk.

Weight on the ramp

A specific resource variable t_i^{k3} is needed to limit the weight of the items loaded on the ramp. The extension function $f_{ij}^{k3}(\mathbf{T}_i^k)$ is similar to equation (9) except that the incrementation of the variable is only done for items on the ramp. The value of t_j^{k3} must not exceed the maximum allowable weight on the ramp.

Center of gravity

We need two resource variables to model the center of gravity. Let x_j^k be the position of the center of gravity of item j in aircraft k, l_j its length, and w_j its weight. For a new item to be loaded, we must have:

$$x_j^k = t_i^{k1} + \alpha_j l_j$$

where t_i^{k1} corresponds to the total length of the items previously loaded, i being the last one. If we define I^k as the set of items loaded in the aircraft k, then the center of gravity of the aircraft CG^k is:

$$CG^k = \sum_{i \in I^k} w_i x_i^k / \sum_{i \in I^k} w_i \tag{10}$$

This suggests the following resource variable:

$$f_{ij}^{k4}(\mathbf{T}_i^k) = \begin{cases} \frac{t_i^{k2} t_i^{k4} + w_j (t_i^{k1} + \alpha_j l_j)}{t_i^{k2} + w_j} & \text{if } j \text{ corresponds to an item} \\ t_i^{k4} & \text{otherwise.} \end{cases} \tag{11}$$

Since the items are loaded as near to the front as possible, the position of the CG will necessarily increase as new items are loaded. Then at each node we have to verify if its value does not exceed the upper limit $\overline{g}$:

$$t_j^{k4} \leq \overline{g}$$

If at some node the CG is below the lower limit $\underline{g}$, there are two ways to increase it: add more items or move aft the already loaded items. At this point, it is sufficient to verify whether a translation of the loaded items will satisfy the constraint. The new value of the

CG is given by:

$$f_{ij}^{k5}(\mathbf{T}_i^k) = \begin{cases} \frac{t_i^{k2}(t_i^{k5}-l_j)+w_j(L_B-(1-\alpha_j)l_j)}{t_i^{k2}+w_j} & \text{if } j \text{ corresponds to an item in the bay} \\ \frac{(t_i^{k2}-t_i^{k3})t_i^{k5}+t_i^{k3}(t_i^{k5}-l_j)+w_j(L-(1-\alpha_j)l_j)}{t_i^{k2}+w_j} & \text{if } j \text{ corresponds to an item in the ramp} \\ t_i^{k5} & \text{otherwise.} \end{cases} \tag{12}$$

This expression is the result of the displacement of the already loaded items to the rear of the compartments (bay or ramp). For all nodes j of network k, the value of t_j^{k5} is only constrained to be positive, except for the last node D^k. At node D^k, no more item will be loaded. So, the value of t_j^{k5} is restricted to be at least the lower limit of the CG, i.e.,

$$\underline{g} \leq t_j^{k5}$$

2.3 Treatment of Special Cases

If passengers are to be transported, they must be sit in the front of the aircraft and, therefore, the first series of items consists only of one group of nodes. In this group, there is one node for each row of seats, one node for the beginning and one node for the end of the group.

When priorities on precedence of loading are associated with the items, we construct one network for each priority group of items that can be transported together. For example, if there are ten priority levels and it is forbidden to transport one item of priority p until all the items of priority less than $p-1$ are transported, we construct nine networks. The first network contains the items of priority 1 and 2, the second one, the items of priority 2 and 3, and so on. With this structure, one item may be represented by several nodes (one in each series), at different positions (in the bay or on the ramp) and in different networks (in all networks containing its priority).

3 The Algorithm

We now briefly describe the algorithm used to solve the model. We refer the reader to Desaulniers *et al* [5] and Desrosiers *et al* [6] for more information on this approach. The model defined by (1)-(7) has a block-diagonal structure, which suggests the utilization of the Dantzig-Wolfe decomposition algorithm or, equivalently a column generation approach. Without the coupling constraints (2), the problem decomposes in $\mid K \mid$ subproblems. The subproblem for the commodity $k \in K$ corresponds to constraints (3)-(7) for a given k. These constraints define a path structure used to send one flow unit between O^k and D^k. A feasible path is a path that respects the resource constraints of a feasible chalk. At the start of the algorithm, we consider only the trivial paths corresponding to the loading of a single item. At any stage of the decomposition algorithm, the master problem consists of determining the best combination of paths that satisfies the coupling constraints (2). After solving this linear program, an optimal marginal value is associated with each item. These values are then used to obtain profitable chalks by solving the subproblem. The

objective function of a subproblem is the sum of the marginal values of the items included in the corresponding chalk. This subproblem is then a shortest path problem with resource constraints (SPPRC). A set of good feasible paths are sent to the master problem. This procedure is repeated untill no more good path can be found. If the solution to the original problem is fractional, cut and branch-and-bound (B&B) strategies are used. The optimal value Z_{LP} of the linear relaxation of the problem is rounded up to the next integer to form the following cut that is added to the linear relaxation:

$$\sum_{k \in K} \sum_{a \in \Gamma^{-1}(O^k) \setminus (O^k, D^k)} X_a^k \geq \lceil Z_{LP} \rceil.$$

This constraint has the same form as a branching constraint and is transferred to the subproblem simply by subtracking from the reduced cost a constant equal to the corresponding dual variable. At each node of the B&B tree of the B&B algorithm, the linear relaxation defined by (1)-(7) is solved by the column generation procedure. We used the branching rule developed in Desrochers and Soumis [6], a special case of the Ryan and Foster [16] branching scheme for master problems with a set partitioning structure. Given a fractional linear relaxation, branching decisions are taken on the values of the flows between two consecutive items that represent the quantity loaded. Since all items must be loaded exactly once as imposed by constraints (2), these values are subject to binary requirements. Therefore, two branching nodes are created: one for which the flow value is set to 1, and another for which it is set to 0. The search process first explores the branch where the flow is set to 1 and search the successor of that node until it encounters a node where an extra aircraft would be required. It then backtracks one level, moves to the branch where the flow is set to 0 and goes depth first from there. In both branches, the decision is imposed at the subproblem level. The shortest path algorithm has to be modified to ensure that the dominance procedure leads to an optimal solution. As described in Dumas, Desrosiers and Soumis [8], this is done by keeping track of the last item loaded for each label of the dynamic programming algorithm, and by comparing labels only if they have the same last item. Thus, all subproblems keep a shortest path structure with the same number of resource constraints.

The procedure used to solve the SPPRC is based on a labelling algorithm that embeds dominance tests to reduce the state space as early as possible. The labels require the cost component as well as additional components for each resource. Each label corresponds to a partial path from the origin node O^k to the current node and the algorithm requires recording at each node all multi-dimensional labels. Desaulniers *et al* [7] prove, in the special case where all extension functions are nondecreasing including the cost extension function, that state s dominates state t if each component of s does not exceed the corresponding component of state t. Then, the elimination of that state t is possible since this label cannot be present in any optimal shortest path solution. In our model, the resource extension functions associated to the CG do not possess the nondecreasing property with respect to the second resource. We propose the following rules for testing the dominance relationship between two states.

Proposition 1 *Let s, t be two states at a given node $i \in N$ with an associated vector of resource values and a cost. State s dominates state t if:*

$$\begin{aligned} c_i^s &\leq c_i^t \\ t_i^{k1,s} &\leq t_i^{k1,t} \\ t_i^{k2,s} &= t_i^{k2,t} \\ t_i^{k3,s} &\leq t_i^{k3,t} \\ t_i^{k4,s} &\leq t_i^{k4,t} \\ t_i^{k5,t} &\leq t_i^{k5,s}. \end{aligned}$$

Proof: Suppose that the extension of state t using arc $(i,j) \in A^k$ is a feasible state t' at node $j \in N^k$. We show that state s can also be extended using arc (i,j) to yield a feasible state s' with cost $c_j^{s'} \leq c_j^{t'}$. For resources 1, 2 and 3, it is trivial that state s' is feasible. For the fourth resource, we have:

$$\begin{aligned} f_{ij}^{k4}(\mathbf{T}_i^s) &= \frac{t_i^{k4,s} t_i^{k2,s} + (t_i^{k1,s} + \alpha_j l_j) w_j}{t_i^{k2,s} + w_j} \\ &\leq \frac{t_i^{k4,t} t_i^{k2,t} + (t_i^{k1,t} + \alpha_j l_j) w_j}{t_i^{k2,t} + w_j} \\ &= f_{ij}^{k4}(\mathbf{T}_i^t) \leq t_j^{k4,t'} \leq \overline{g}, \end{aligned}$$

if j corresponds to an item. Otherwise, we have

$$\begin{aligned} f_{ij}^{k4}(\mathbf{T}_i^s) = t_i^{k4,s} &\leq t_i^{k4,t} \\ &= f_{ij}^{k4}(\mathbf{T}_i^t) \\ &\leq t_j^{k4,t'} \leq \overline{g}. \end{aligned}$$

For the fifth resource, we have:

$$\begin{aligned} \underline{g} \leq t_j^{k5,t'} = f_{ij}^{k5}(\mathbf{T}_i^t) &= \frac{t_i^{k2,t}(t_i^{k5,t} - l_j) + w_j(L_B - (1-\alpha_j) l_j)}{t_i^{k2,t} + w_j} \\ &\leq \frac{t_i^{k2,s}(t_i^{k5,s} - l_j) + w_j(L_B - (1-\alpha_j) l_j)}{t_i^{k2,s} + w_j} \\ &= f_{ij}^{k5}(\mathbf{T}_i^s), \end{aligned}$$

if j corresponds to an item in the bay, and

$$\begin{aligned} \underline{g} \leq t_j^{k5,t'} = f_{ij}^{k5,t}(\mathbf{T}_i^t) &= \frac{t_i^{k2,t} t_i^{k5,t} - t_i^{k3,t} l_j + w_j(L - (1-\alpha_j) l_j)}{t_i^{k2,t} + w_j} \\ &\leq \frac{t_i^{k2,s} t_i^{k5,s} - t_i^{k3,s} l_j + w_j(L - (1-\alpha_j) l_j)}{t_i^{k2,s} + w_j} \\ &= f_{ij}^{k5,s}(\mathbf{T}_i^s), \end{aligned}$$

if j corresponds to an item in the ramp.

Then if it is possible to extend state t to t', it is also possible to extend state s to s' with cost $c_j^{s'} \leq c_j^{t'}$ and the label associated with t can be eliminated. □

We emphasize that the size of the network is reduced by grouping together the nodes corresponding to the same type of item in each series. This simplification removes the possibility of taking the same item twice successively in the same chalk. Moreover, the number of chalks that include two or more items of the same type successively is substantially reduced since only the arcs $(j, j+1)$, where j is a node corresponding to an item, are presents. This is referred to as the symmetry of the problem. We can build examples where the linear relaxation is tightened by using these strategies.

4 Computational Results

The algorithm just described was tested on real life and on randomly generated problems. The real life problem comes from Ng (1992) and is reported in Table 1. The characteristics of each item are those given by the Department of National Defence of Canada (DND). To assess the impact of priorities on algorithmic performance, we have added for some scenarios arbitrary priority numbers between 1 and 4 for each item. These 322 items must be transported in a Hercules C130 for which the length of the bay is 492 inches and the length of the ramp is 132 inches. The maximum allowed weight is 195900 lbs in total, and 5000 lbs on the ramp. Items 2, 7, 11, 15 and 16 are the only ones that can be loaded on the ramp.

All tests were performed on an Enterprise 10000 computer(400 Mhz, 64Go, unix, gcc compiler). To solve the problems, we have used the GENCOL software developed by Desrosiers, Dumas, Solomon and Soumis [7].

4.1 Number of series

The number of series that we consider in each network has an obvious impact on the difficulty of the problem. We can easily compute a priori the largest number of series needed in the model to optimally solve the ALP; it corresponds to the number of items in a chalk containing an alternate load of the smallest and the next smallest item still available. For the data of Table 1, we need 6 series for the bay and 1 for the ramp (obtained from items number 15 and 13). Table 2 presents the results on Ng's data when we let the number of series vary from 2 to 6 for the bay and keeping one series for the ramp. The optimal solution is guaranteed only when the number of series considered is equal to 6, even if it is obtained when 3 series or more are used.

For these tests, the window for the CG is fixed to the medium size (551,564) which corresponds to the real constraint for an Hercules C130 (Richardson (1993)) and no priority is imposed for the loading. For each problem, we give the resolution time (CPU in seconds), the resolution time of the linear relaxation (TiRel in seconds), the optimal solution (Sol), the value of the linear relaxation (Relax), the integrality gap (Gap), the number of nodes explored (B&B), the number of column generation iterations (ItColGen), the number of shortest paths called (SP), the number of column generated (ColGen), the number of arcs

i	Qty	l_i (in.)	w_i ($\approx \times$100lbs)	p_i	Item Description
1	40	158	35	1	0.25 ton utility truck
2	48	124	40	1	1.25 ton cargo truck
3	3	279	195	2	2.5 ton bowser
4	47	267	170	3	2.5 ton medium logistics vehicle wheeled
5	2	204	213	3	5.0 ton truck
6	11	232	248	4	armored personnel carrier M113
7	34	109	11	3	0.25 ton trailer
8	18	147	29	2	0.75 ton trailer
9	19	166	54	1	1.5 ton trailer
10	2	158	120	2	rough terrain fork lift
11	4	109	10	3	Herman Nelson heater
12	2	271	225	1	front head loader
13	22	104	205	4	armored vehicle general purpose
14	6	166	51	2	howitzer 105
15	9	58	30	2	pallet 54" $\times$ 88"
16	11	109	10	3	triwalls
17	6	252	80	1	twin huey CH135
18	3	388	20	3	kiowa CH136
19	34	224	94	1	1.25 ton command post
20	1	310	338	4	5.0 ton wrecker

Table 1: Data set for real-life instance

and nodes in the network (Arcs and Nodes) and the maximum memory required to solve the problem (Mem in meg).

These first results show that we can solve the DND problem optimally in less than 6 minutes, which is clearly acceptable for a planning decision. The CPU time, the size of the network and the memory required to solve the problem all increase with the number of series considered. The results show that three series are sufficient to find the optimal number of chalks (92) for that set of items. This solution requires 18 trips fewer than Ng's solution. Using the costs estimated by Ng, (around \$110000/trip) this solution costs \$2 millions less than Ng's solution and \$3 millions less than the first solution proposed by the loadmaster. Ng solves the covering problem by considering 38 different selected good patterns. The column generation approach allows us to consider implicitly all feasible patterns (9383 patterns were generated), which explains the quality of the solution.

We have observed that the "mixed integer rounding up property" holds for all problems solved with the Ng and randomly generated problems. However, counter-examples show that this property is not in general valid for the ALP.

model	2 Layers	3 Layers	4 Layers	5 Layers	6 Layers
CPU	89	204	215	271	319
TiRel	76	143	151	188	190
Sol	94	92	92	92	92
Relax	93.13	92.00	92.00	92.00	92.00
Gap %	0.93	0.00	0.00	0.00	0.00
B&B	190	208	202	181	210
ItColGen	1149	1519	1475	1616	1958
SP	959	1311	1273	1435	1748
ColGen	6065	10100	9001	9128	9383
Arcs	2695	4061	5427	6793	8159
Nodes	843	1205	1567	1929	2291
Mem	8.42	28.71	51.37	57.06	57.81

Table 2: Different number of series for the bay and one fixed layer for the ramp

4.2 Number of items and CG constraint

The second test (Tables 3,4,5) measures the impact of the quantity of items to be moved, the tightness of the CG constraint and the use of priorities. Based on Ng's data and using two priority scenarios (without and with 4 priorities as indicated in Table 2), we have solved 4 problems corresponding to the number of items considered. The first problem, labelled "Single", consists of Ng's problem, the second, labelled "Double", has twice the number of items of each type, and so on for the column labelled "Triple" and "Fourfold ". The first window CG is fixed to (245,737), which is equivalent to relaxing the CG constraint (the center of gravity can then be anywhere within the bay), the second (551,564) is the actual accepted CG for the Hercules, while the third one (559,563) is artificially tight.

We observe that CPU time increases exponentially with the number of items to be transported even if the other parameters (number of iteration, columns generated, number of nodes, etc.) indicate that problem difficulty seems to *increase* linearly. This is due to the algorithms used to solve the restricted master problem and the subproblems. Observe that we can adapt our algorithm to solve problems for which the center of gravity of each item is not necessarily located at the middle of the item ($\alpha = 1/2$). In such a case, each item has to be duplicated in the network to model the two possible positions (in front or behind) of the loading. Hence, the columns labelled "Double" give an estimate on the amount of CPU time and memory required if we such a situation is modelled with the same type of data.

The instances including priorities on items are always faster to solve because of their smaller size. The impact of priorities on the minimum number of chalks found is clearly illustrated in Tables 3,4 and 5. For example, with a medium window on the CG, 14 additional chalks are needed to transport the 322 items. The impact of the CG constraint is also not negligible. Tightening the CG constraint from (551,564) to (559,563) requires 5 additional chalks if no priority is considered and 2 otherwise.

Priority	1 Priority				4 Priorities			
model	Single	Double	Triple	Fourfold	Single	Double	Triple	Fourfold
CPU	202	1963	8715	22590	77	655	2963	8646
TiRel	71	544	1979	5247	19	121	472	1241
Sol	92	184	275	367	101	201	302	402
Relax	91.55	183.09	274.64	366.18	101.00	201.00	302.00	402.00
Gap %	0.49	0.49	0.13	0.22	0.00	0.00	0.00	0.00
B&B	289	564	864	1149	242	476	743	959
ItColGen	1292	2639	4206	5624	763	1541	2345	3155
SP	1003	2075	3342	4475	1563	3195	4806	6588
ColGen	4293	9356	14150	19316	3026	6422	9515	13039
Arcs	8159	14273	20387	26501	10476	19266	28056	36846
Nodes	2291	4329	6367	8405	2291	4329	6367	8405
Mem	76.36	154.95	241.00	334.27	11.02	26.05	46.22	70.74

Table 3: Large window for the CG: (245,737)

Priority	1 Priority				4 Priorities			
model	Single	Double	Triple	Fourfold	Single	Double	Triple	Fourfold
CPU	319	2410	9589	23680	83	805	3588	8335
TiRel	191	1218	4366	9719	22	134	451	1078
Sol	92	184	276	368	106	212	318	424
Relax	92.00	184.00	276.00	368.00	106.00	212.00	318.00	424.00
Gap	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
B&B	210	443	631	914	198	468	702	932
ItColGen	1958	4219	7992	10063	747	1702	2547	3346
SP	1748	3776	7361	9149	1647	3702	5535	7242
ColGen	9383	26127	56548	59151	3077	6771	9828	13899
Arcs	8159	14273	20387	26501	10476	19266	28056	36846
Nodes	2291	4329	6367	8405	2291	4329	6367	8405
Mem	57.81	119.72	189.23	263.16	10.74	25.94	45.93	76.24

Table 4: Medium window for the CG: (551,564)

Priority	1 Priority				4 Priorities			
model	Single	Double	Triple	Fourfold	Single	Double	Triple	Fourfold
CPU	216	1503	5860	12770	93	869	3776	9747
TiRel	164	876	2893	6235	27	137	406	1072
Sol	97	194	291	388	108	216	324	432
Relax	96.86	193.71	290.57	387.43	108.00	216.00	324.00	432.00
Gap	0.14	0.14	0.14	0.14	0.00	0.00	0.00	0.00
B&B	199	491	695	901	185	395	603	787
ItColGen	1175	2614	4595	5911	809	1665	2772	3590
SP	976	2123	3900	5010	1872	3810	6507	8409
ColGen	7567	17863	33949	46319	3276	6735	10002	13665
Arcs	8159	14273	20387	26501	10476	19266	28056	36846
Nodes	2291	4329	6367	8405	2291	4329	6367	8405
Mem	46.71	97.95	157.70	222.95	10.70	25.18	44.04	68.42

Table 5: Tight window for the CG: (559,563)

4.3 Random instances

We conclude the numerical results by solving the ALP on 30 randomly generated instances. The CG window is set to (551,564) and no priority is imposed. The instances are generated so as to approximate real life situations. The number of different types is set to 20 for all the 30 problems. The quantity for each type is a random number between 1 and 50, with a higher probability in the interval [10;30]. The length of each type is between 50 and 300 with a higher probability in [100;150], and the weight of each type is between 10 and 350 with a higher probability in [10;100].

The results are detailed in Tables 6 to 9. The average and standard deviation are given in the first two columns (AVE and SD) of Table 6. Four additional lines are given in these tables: the number of items to be transported (Items), the average of items per chalk (Itm/chalk), the number of series used in the model (Series) and the maximum number of items found in one chalk in the optimal loading (NbMaxIt). We note that the number of items to be transported is always larger than the 322 items considered in Ng and the average number of items per chalk is about the same. This is certainly the most determinant factor of the difficulty of a problem, as confirmed by the three most difficult problems (N18, N24 and N29). The results indicate that our approach is efficient when the number of items per chalk remains relatively small (less than five, which was the case for the DND problem). When this number increases, the memory required to solve the problem becomes prohibitive (mainly due to the exponential increase of labels that can not be dominated in the shortest path problem with resources algorithm). Recall that each chalk is represented as a suitably constrained path in a network and a multilabel shortest path algorithm is used to evaluate the cost. Solving exactly the ALP problem with small items would probably require the aggregation of small items into larger ones.

	AVE	SD	N1	N2	N3	N4	N5	N6
CPU	1628	2015	340	224	763	4748	2197	552
TiRel	1120	1716	111	152	359	4266	1099	300
Sol	144.53	25.73	135	120	151	103	151	140
Relax	144.18	25.70	135.00	119.39	151.00	102.58	150.83	139.30
Gap	0.25	0.19	0.00	0.51	0.00	0.40	0.11	0.50
B&B	377	74	297	266	366	398	441	355
ItColGen	2290	707	1736	1620	2430	2130	4274	2209
SP	1946	737	1439	1354	2064	1732	3833	1854
ColGen	9959	7873	4261	4237	6681	13511	9025	5445
Arcs	11059	2281	8249	7460	9331	10922	13048	9303
Nodes	3272	698	2411	2143	2770	3212	3919	2764
Mem	270.75	346.38	59.15	18.98	191.88	913.81	291.70	55.19
Items	507	70	426	388	504	469	587	487
Itm/chalks	3.50	2.73	3.16	3.23	3.34	4.55	3.89	3.48
Series			5+1	5+0	5+1	6+1	6+1	5+1
NbMaxIt	5.04	0.84	5	4	5	6	7	4

Table 6: Random instances (part 1)

	N7	N8	N9	N10	N11	N12	N13	N14
CPU	510	1282	1145	704	1614	1949	227	1066
TiRel	192	950	385	318	1058	1685	78	516
Sol	157	146	200	133	195	120	131	146
Relax	156.87	145.92	199.83	132.50	194.27	119.41	131.00	145.64
Gap	0.08	0.05	0.08	0.37	0.37	0.49	0.00	0.24
B&B	324	299	462	392	422	323	282	420
ItColGen	1458	1919	2597	1904	3114	1878	1150	2774
SP	1134	1620	2135	1512	2692	1555	868	2354
ColGen	5475	5549	7092	6277	10722	9790	3976	6972
Arcs	10686	14035	11407	10858	15527	10183	8089	12073
Nodes	3130	4058	3462	3184	4647	2959	2361	3594
Mem	165.74	358.10	95.59	223.12	243.66	417.07	48.88	89.23
Items	470	463	640	483	617	445	410	534
Itm/chalks	2.99	3.17	3.20	3.63	3.16	3.71	3.13	3.66
Series	6+1	8+1	5+1	6+1	7+1	6+1	5+1	6+1
NbMaxIt	5	5	6	6	5	5	5	5

Table 7: Random instances (part 2)

	N15	N16	N17	N18	N19	N20	N21	N22
CPU	1144	823	1394	4370	1212	458	350	892
TiRel	780	522	815	3540	724	141	155	615
Sol	137	112	152	124	111	162	159	118
Relax	136.54	111.96	151.18	123.48	110.78	161.75	158.40	117.46
Gap	0.33	0.03	0.54	0.42	0.20	0.15	0.37	0.46
B&B	367	315	460	470	407	379	349	353
ItColGen	2342	1870	2751	3634	2293	1910	1509	2314
SP	1975	1555	2291	3164	1886	1531	1160	1961
ColGen	9523	19645	9934	30464	11951	4867	4329	10027
Arcs	13094	8379	12667	10692	10726	9461	10848	8940
Nodes	3836	2451	3792	3227	3145	2815	3184	2643
Mem	191.88	113.38	167.00	751.98	273.58	32.15	57.52	196.71
Items	504	440	564	571	462	505	483	467
Itm/chalks	3.68	3.93	3.71	4.60	4.16	3.12	3.04	3.96
Series	7+1	5+1	6+1	5+1	6+1	5+1	6+1	5+1
NbMaxIt	5	5	4	6	5	4	4	5

Table 8: Random instances (part 3)

	N23	N24	N25	N26	N27	N28	N29	N30
CPU	175	9782	1021	338	472	1581	5781	1736
TiRel	43	7859	596	171	276	437	4802	664
Sol	188	114	154	128	174	166	123	186
Relax	188.00	113.94	154.00	127.42	173.17	165.50	122.88	185.33
Gap	0.00	0.05	0.00	0.45	0.48	0.30	0.10	0.36
B&B	250	429	313	267	384	492	498	526
ItColGen	1212	2869	2733	1708	1809	2201	3050	3292
SP	962	2440	2420	1441	1425	1709	2552	2766
ColGen	3142	27489	8068	4177	5209	7089	35321	8516
Arcs	8821	14471	13017	8966	9917	15556	13027	14048
Nodes	2600	4300	3812	2555	2967	4660	3912	4254
Mem	22.06	1708.05	145.49	81.09	38.96	332.77	740.32	97.36
Items	467	557	499	382	545	607	585	641
Itm/chalks	2.47	4.89	3.24	2.96	3.13	3.66	4.76	3.45
Series	5+1	7+1	7+1	6+1	5+1	7+1	6+1	4+1
NbMaxIt	4	6	5	4	4	6	6	5

Table 9: Random instances (part 4)

5 Conclusions

We have proposed a model to solve the aircraft loading problem to optimality. This method is based on a column generation approach and has proven to be efficient for real life problems when the items to load are large. We should now explore aggregation techniques to deal with small items.

References

[1] Amiouny S.V., Bartholdi III J.J., Van de Vate J.H., Zhang J., (1992), "Balanced Loading", *Operations Research* , 40, 238-246.

[2] Bischoff, E.E., Ratcliff, M.S.W., (1995), "Issues in the Development of Approaches to Container Loading", *Omega*, 23, 377-390.

[3] Brosh, I., (1981), "Optimal Cargo Allocation on Board a Plane: A Sequential Linear Programming Approach", *European Journal of Operational Research* , 8, 40-46.

[4] Cochard, D.D., Yost, K.A., (1985), "Improving Utilization of Air Force Cargo Aircraft", *Interfaces*, 15, 53-68.

[5] Desaulniers, G., Desrosiers, J., Ioachim, I., Solomon, M.M., Soumis, F., Villeneuve, D., (1998), "A Unified Framework for Determistic Time Constrained Vehicle Routing and Crew Scheduling Problems", in *Fleet Management and Logistics*, T. Crainic and G. Laporte (eds), Kluwer, Boston.

[6] Desrochers, M., Soumis, F., (1989),"A Column Generation Approach to the Urban Transit Crew Scheduling Problem", *Transportation Science*, 23, 1-13.

[7] Desrosiers, J., Dumas, Y., Solomon, M.M., Soumis, F., (1995), "Time Constrained Routing and Scheduling", in *Network Routing*, M.O. Ball, T.L. Magnanti, C.L. Monma, and G.L. Nemhauser (eds), Handbooks in Operations Research and Management Science, 8, 35-139.

[8] Dumas, Y., Desrosiers, J., Soumis, F., (1991),"The Pickup and Delivery Problem with Time Windows", *European Journal of Operational Research*, 54, 7-22.

[9] Dyckhoff, H., (1981),"A New Linear Programming Approach to the Cutting Stock Problem", *Operations Research*, 29, 1092-1104.

[10] Gehring, H., Menshner, K., Meyer, M., (1990), "A computer-based Heuristic for Packing Pooled Shipment Containers", *European Journal of Operational Research*, 44, 277-288.

[11] Huebner, W.F., (1982), "Load Planning, Rapid Mobilization and the Computer", *Air Force Journal of Logistics*, 6, 22-24.

[12] Larsen, O., Mikkelsen, G., (1980), "An Interactive System for the Loading of Cargo Aircraft", *European Journal of Operational Research* , 4, 367-373.

[13] Martin-Vega, L.A., (1985), "Aircraft Load Planning and the Computer: Description and Review", *Computers & Industrial Engineering* , 9, 357-369.

[14] Ng, K.Y.K., (1992), "A Multicriteria Optimization Approach to Aircraft Loading", *Operations Research* , 40, 1200-1205.

[15] Richardson, J., (1993), An Expert System for Military Aircraft Load Planning. M.Sc. Thesis, Collège Militaire Royal de St-Jean, Canada.

[16] Ryan, D.M., Foster, B.A, (1981), "An Integer Programming Approach to Scheduling", in *Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling*, A. Wren (ed.), North-Holland, Amsterdam, 269-280.