# Load Packing Algorithm

Tesla Motors

19 August 2015

Nikita Korolko

Contacts:
Before 8/28/2015: nkorolko@teslamotors.com
After 8/28/2015: korolko@mit.edu

Tesla Motors works with hundreds of suppliers all over the world and each day thousands of different auto parts arrive to Fremont facility. In order to run manufacturing smoothly and efficiently it is essential to schedule deliveries accurately and at minimum possible costs. In this memo we consider a subproblem of load packing into trucks trying to minimize shipping costs.

## 1. Description of the algorithm

Let's assume that we are given a list of boxes that we need to ship. Dimensions, weights and stackability properties of the boxes are assumed to be known. We also know the properties of trucks, both physical (dimensions, weight capacity) and financial (freight cost). The main goal of the suggested algorithm is to place all the boxes into trucks in a way that minimizes the total shipping costs.

There are two different types of shipping method: full truck load (FTL) and less than truck load (LTL). FTL containers have dimensions $W = 96, D = 630, H = 105$ inches, maximum weight capacity $44,000$ lbs and a fixed rental cost that does not depend on items loaded, but does depend on a pickup location. LTL containers have dimensions $W = 100, D = 240, H = 105$ inches, maximum weight capacity $20,000$ lbs and a rental cost that can be approximated as

$$\text{LTL cost} = c_0 + c_1 w,$$

where $w$ is a total weight of the shipment, $c_0$ and $c_1$ are some fixed constant values that depend on a class of the shipment (65, 150, etc, which is basically a function of density of the shipment) and a distance from pickup location to a delivery point.

The algorithm takes all the data mentioned above into consideration and splits the entire shipment into two groups: FTL subset and LTL subset, saying what is the optimal method of shipment for each of the boxes. Moreover, the algorithm will say what is the minimum number of FTL and LTL containers necessary to ship all the boxes/pallets. Finally, it will output the placement and orientation of boxes inside containers.

## 2. Assumptions of the algorithm

1. LTL container is assumed to have dimensions $W = 100, D = 240, H = 105$ due to Christopher Drossulis from ODFL ($W = 100$) and "20 feet constraint" ($D = 240$).

2. Algorithm allows horizontal rotation of the pallets inside container. A vertical rotation is prohibited, i.e. a height of the pallet must be always parallel to a height of the container.

3. The algorithm consists of 2 stages.

   The first stage (Fig.1) tries to find *a feasible* physical placement of boxes inside a container. Due to the specific design of the algorithm some gaps between containers are possible. (Different colors of boxes correspond to different types of them.)

   The second stage (Fig.2) squeezes all pallets together trying to put them side by side and delete any gaps.

   In most of the cases even the output of the 1st stage will not have any gaps. But if the total number of different pallet types is large, gaps are possible. Therefore the 2nd stage is sometimes essential.

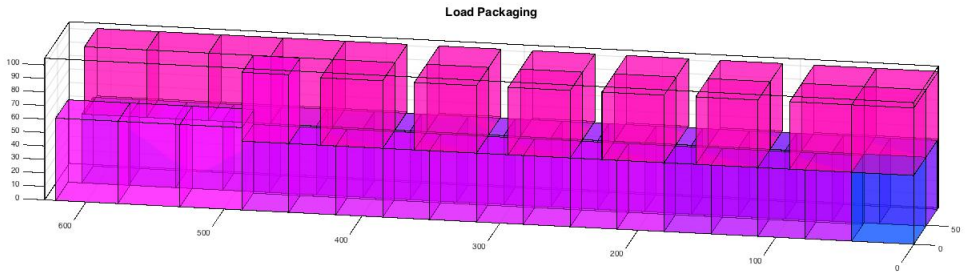


Figure 1: Stage 1 of the algorithm

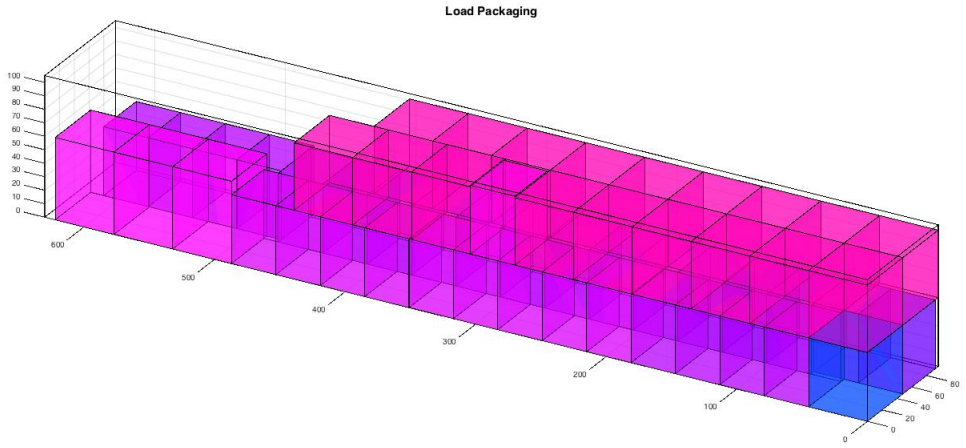


Figure 2: Stage 2 of the algorithm

### 3. Input data description

An example of input data for the algorithm is stored in a file "Bin_data.csv". Each line corresponds to a different type of the pallet. The description of the columns:

- A: pallet width
- B: pallet depth
- C: pallet height
- D: weight of the pallet
- E: number of pallets of this type in the shipment
- F: stackability indicator. (By default this value is equal to 1 and it means that this type of pallets is stackable. If it is equal to 0, then we are not allowed to stack anything on top of pallets of this type).

It is also necessary to indicate in a source file "BinLTL-H3.jl" the shipping costs:

- Parameter *ship_cost* for FTL cost (Line 18)
- Parameters *c0* and *c1* for LTL cost (Line 19-20)

### 4. Output data description

Examples of output data for the algorithm is stored in files "boxes_1.csv" and "2boxes_1.csv". The name of the file is meaningful. The files that start with "boxes..." and "2boxes.." correspond to the output of the 1st and 2nd stage of the algorithm, respectively.

The number after the underscore symbol in the name of the file is a number of a container. For instance, if shipment is placed inside 3 LTL trucks and 4 FTL trucks, the algorithm will generate $2(3+4) = 14$ files:

7 labeled as "boxes_1.csv", ..., "boxes_7.csv"; and

7 labeled as "2boxes_1.csv", ..., "2boxes_7.csv", where LTL containers always go first.

Each line in a file "(2)boxes_ j.csv" corresponds to a different pallet (not a type of pallet as in the input) placed inside container $j$.

The description of the columns:

- A: Coordinate $x$ of left front bottom corner of the pallet (Axis $x$ is parallel to width $W = 96/100$ of the container)
- B: Coordinate $y$ of left front bottom corner of the pallet (Axis $y$ is parallel to depth $W = 630/240$ of the container)
- C: Coordinate $z$ of left front bottom corner of the pallet (Axis $z$ is parallel to height $H = 105$ of the container)
- D: pallet width (can be changed wrt to initial width if pallet is rotated)
- E: pallet depth (can be changed wrt to initial depth if pallet is rotated)
- F: pallet height
- G: stackability indicator of the pallet
- H: Auxiliary. Number of type of the pallet (necessary to paint pallets of the same type in the same color)
- I: Auxiliary. Indicator if this pallet is shipped by FTL method (1) or LTL method (2).

## 5. Algorithm's implementation

The current version (8/19/2015) of the algorithm is implemented in a programming language Julia (http://julialang.org/).

The main reason is that the connection with optimization package JuMP is efficient and easy. (https://jump.readthedocs.org/en/latest/)

The source file is "BinLTL–H3.jl".

The 1st stage of the algorithm (Lines 1-1441) does not use any specific operators, functions or datastructures. It can be rewritten in C# without loss of functionality.

The 2nd stage of the algorithm (Lines 1442-1676) formulates and solves an auxiliary mixed-integer linear optimization problem. The problem is formulated in optimization package JuMP and is solved using a commercial solver Gurobi (http://www.gurobi.com/).

The visualization of recommended solution is implemented in MATLAB using a function "voxel.m":
(http://www.mathworks.com/products/matlab/),
(http://www.mathworks.com/matlabcentral/fileexchange/46564-voxel-m).

### 5.1 Major parameters of the code

1. *max_LTL* is the maximum number of LTL containers that can be used for a given shipment (Line 22)

2. *nu_min* is a minimum fraction of surface square that should have support. In other words, pallets are allowed to hang a little without support, but not more than $100(1 - nu\_min)\%$ of surface square can hang. (Line 25)

3. *W,D,H* dimensions of FTL containers (Lines 44-46); *W2,D2,H2* dimensions of LTL containers (Lines 721-723).