CORA Cork Open Research Archive
Cartlann Taighde Oscailte Chorcaí

Title	Modelling choices for the Roadef 2022 challenge
Authors	Simonis, Helmut
Publication date	2024-09-02
Original Citation	Balogh, A., Dev Gupta, S., Argiro, J., Restrepo, L., O'Sullivan, B., Simonis, H. and Souza, F. (2024) 'Modelling Choices for the Roadef 2022 Challenge', ModRef 2024, The 23rd workshop on Constraint Modelling and Reformulation, Part of CP 2024, 2-6 Sept, Girona, Catalonia.
Type of publication	Conference item
Link to publisher's version	https://modref.github.io/ModRef2024
Rights	© 2024 theAuthors. This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant number 12/RC/2289-P2 at Insight the SFI Research Centre for Data Analytics at UCC, which is co-funded under the European Regional Development Fund. For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission. - http://creativecommons.org/licenses/by/4.0/
Download date	2026-10-30 11:23:37
Item downloaded from	https://hdl.handle.net/10468/16408

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Modelling Choices for the Roadef 2022 Challenge

Andrea Balogh
Insight SFI Centre for Data Analytics, School of Computer Science and Information Technology, University College Cork, Cork, Ireland

Sharmi Dev Gupta
Insight SFI Centre for Data Analytics, School of Computer Science and Information Technology, University College Cork, Cork, Ireland

Jheisson Lopez
Insight SFI Centre for Data Analytics, School of Computer Science and Information Technology, University College Cork, Cork, Ireland

Barry O'Sullivan
Insight SFI Centre for Data Analytics, School of Computer Science and Information Technology, University College Cork, Cork, Ireland

Helmut Simonis
Insight SFI Centre for Data Analytics, School of Computer Science and Information Technology, University College Cork, Cork, Ireland

Filipe Souza
Insight SFI Centre for Data Analytics, School of Computer Science and Information Technology, University College Cork, Cork, Ireland

Abstract

This paper describes our approach to modelling and solving the Roadef 2022 Challenge, a transportation planning problem introduced by Renault. We describe a high-level decomposition of the problem, and the models for different stages, focussing on a MIP main problem which decides when in time stacks of items should be transported. We present a lower bound on the number of trucks required to deliver all stacks in time, and show how the lower bounds on individual placement problems can be incorporated as cuts in the main MIP model.

2012 ACM Subject Classification Applied computing → Industry and manufacturing

Keywords and phrases Transportation, Scheduling, Logistics, Truck Loading

Funding This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant number 12/RC/2289-P2 at Insight the SFI Research Centre for Data Analytics at UCC, which is co-funded under the European Regional Development Fund.
For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission.

1 Introduction

This paper describes our model for solving the Roadef 2022 Challenge, and explains some of the reasoning which lead to the choice of the method presented. The problem consists of organizing the transport of car components from multiple suppliers to multiple factories over a period of several months. The items to be transported are packed in rectangular boxes of different sizes and weights, which need to be placed in trucks of given types to be delivered on-time. A set of planned trucks is given, each truck used incurs a resource cost, which, together with an inventory cost for early delivery form the cost function of the problem. Items can either be placed on the truck floor or on top of compatible items with the same footprint, forming stacks.

The approach presented in this paper extends the model used by our team S43 in the final stage of the competition by adding a cut generation method to the master MIP model, which is responsible for assigning stacks in time. These cuts are created from a lower bound on the placement, which is based purely on the size and orientation of the stacks to be placed. We ignore load stage and axle weight constraints when creating the bound, and do not allow to change the stacks built at an earlier stage in the problem decomposition. While these restrictions limit the quality of the placements that can be achieved, the approach was taken to de-risk the development of a solution given

- the relatively short time for development,
- the limits of our own efforts to participate in the Challenge,
- and the run-time limits imposed on the solution process.

Overall, our approach tries to decompose the problem into multiple smaller sub-problems. At its core is a master MIP model, which assigns stacks, created earlier in the workflow, to trucks, while minimizing the overall objective, a mix of resource cost and earliness cost of items delivered before their due date. Similar to a Logical Benders Decomposition [15, 9], we create a set of sub-problems for placement, which use the assignment of stacks to trucks created in the master problem. If we find that a placement of the assigned stacks to a truck and created copies is not possible, a cut is generated in form of a linear constraint, or a disjunction, which is then added to the master problem. We use a heuristic lower bound calculation to check feasibility of the assignment, as relying on the enumeration of potential placement routines was not deemed practical given the run-time limits of the competition. While Constraint Programming can handle tight rectangle placement problems [19, 20], the time needed to check for infeasibility of a placement with such models prohibits their use for cut generation. Overall, we were forced to replace optimization based methods for a number of sub-problems by heuristics, in order to solve the required number of sub-problems we encounter for the larger problem instances.

2 Problem Description and Related Work

We can only give an overview of the problem posed in the Challenge, the details can be found on the website [1, 2], together with datasets, a checker for solutions, and an overview of the results achieved in the different stages of the competition.

The problem is part of the logistics operation of Renault, transporting components from different suppliers to factories worldwide. Each item, represented as a rectangular box, has a definition of its properties (origin, destination, size, quantity, weight, product, potential forced orientation, time window of transport, and earliness cost per day). We are given the schedule of planned trucks, which are defined by their route (order of pickup at suppliers), their destination, arrival time, internal size of trailer, weight limits, and axle weight parameters. We have to decide which quantity of specific items we want to transport at which time point with which trucks, and how many copies of the planned truck we need at each time point. Not all planned trucks need to be used, we pay a resource cost only for the used planned trucks, and an extra cost for additional truck copies. For each item, we pay no inventory cost if the item is delivered on its due-date, otherwise we pay a inventory cost per day multiplied by the number of earliness days. Items cannot be delivered either too early or too late.

For each truck used, we have to describe the loading of the items in the truck, by giving the x, y, and z coordinates of the items placed. Items must be placed on top of each other in stacks, where all items in a stack have the same width and length, and compatible forced

orientation. The height of a stack is the sum of the heights of the items minus potential overlap of items, which depends on the item type. More complex constraints restrict how many of which items can be stacked together, and what maximal weight is allowed for the stack, or for the items above the bottom piece of the stack.

Stacks must be placed in parallel to the sides of the trailer. The loading must respect the load order, i.e. items from suppliers visited earlier by the truck must be loaded in front, while items from later suppliers can only be placed either behind or to the right of stacks from earlier supplier locations. Obviously, stacks must fit completely inside the trailer, and cannot overlap each other. On the other hand, stacks must be supported in front by either the front side of the trailer, or by another stack.

Besides a total weight limit for all items loaded in a trailer, and the weight limits on individual stacks, we must also satisfy the axle weight constraints, i.e. the weight limits imposed on each of the axles of the semi-truck. A non-linear model to compute the axle weights is given as part of the problem specification. The axle weight limits must not only hold for the fully loaded truck, but also for each of the partial loads when leaving each supplier site. For the trucks given, loading too many stacks in the front of the truck quickly exceeds the axle weight constraint of the second axle, while only filling the truck from the back is not possible as it leaves the stacks unsupported. An acceptable solution must place some stacks in the front of the truck to provide support for additional, heavier stacks placed towards the rear of the truck.

While there is a multitude of papers on truck loading and the underlying two- or three dimensional bin packing problem [13, 12, 10, 14, 6, 17, 22, 4, 23, 8, 18, 7, 16, 5, 21, 3, 11], the Roadef problem defines many very specific side constraints that are not covered by the existing literature. The closest work to our approach seems to be [11], which proposes the use of a Benders Decomposition for the two-dimensional bin-packing problem. The method is only evaluated on instances of several hundred items, and would not generalize to the larger and more restricted problem instances of the competition. At the current time, very little is known about the methods used by the individual teams participating in the competition, only the cost value results of each team in the different stages of the competition have been published so far.

3 High-Level Decomposition

We decided early on that we should use a problem decomposition to split the overall problem into smaller, more easily managed sub-problems. Figure 1 shows the overall design. The first step is a clustering method, which splits the overall transport problem into independent sub-problems finding connected components in a graph built from trucks as nodes, and an edge connecting two nodes if some item can be transported on either truck. Trucks in different connected components cannot interact with each other, we can therefore solve the placement problem for each connected component independently.

As a second step, we build stacks from all items that must be transported at the same time. Once we have decided on the stacks to create, we only need to decide which truck will contain which stack, which determines when the items will be transported, while leaving the problem of how to pack the stacks into the trucks. Using stacks eliminates the third dimension of the placement problem, and we have to consider a sequence of 2D placement problems. The main master problem is assigning the stacks in time, deciding at the same time whether we use a planned truck, and/or if we want to use additional copies of planned trucks. In the master problem, we only use an approximation of the 2D placement, considering the

overall area to be used, and the total weight allowed for each truck. The solution of the master problem allocates each stack to a planned truck, but it is not guaranteed that the assigned set of stacks actually fits within the truck and any allocated copies. We use a lower bound estimate on the placement to detect infeasible assignments, and create cuts for the master problem to avoid those assignments. The lower bound calculation only considers the size, number and orientation of different stack types, and thus does not guarantee that only feasible assignments are generated.

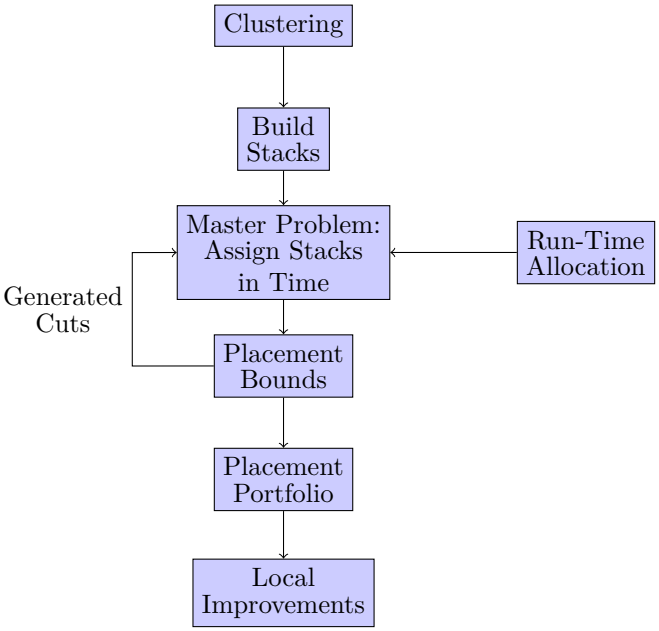


Figure 1 Overall Problem Decomposition

When no more cuts can be added, we try to create an actual placement for each planned truck, allowing the use of additional trucks if we cannot find a solution with the number of trucks suggested by the master problem. We use a portfolio of placement routines to generate placements, most of them heuristics to limit the amount of time needed for each placement. We also check each placement for the axle weight constraints, and reject solutions that are not acceptable.

At this point we have a feasible solution, each item belongs to a stack, each stack is placed in a truck, and all constraints are satisfied. But we may be left with many partially filled trucks. We now apply a local search improvement routine that tries to improve the overall objective value, and we continue to do this until we no longer find improving moves of stacks, or we run out of time.

One of the issues of decomposing the overall problem into multiple independent sub-problems, and solving each by a sequence of process steps is that we have to carefully consider the total time given to make sure that we find a feasible solution for all sub-problems, and we spend the time given on the right sub-problem. In the competition each instance is given a time limit of 30 minutes or one hour, depending on problem size. We use a run-time allocation module to estimate the complexity of each sub-problem, and allocate time accordingly. We assign more time for large clusters, and limit the time for each iteration of the master problem, as well as the time spent in cut generation and placement. We also solve sub problems starting with smaller instances, and reallocate any time left over for the

remaining, more complex sub problems.

This decomposition chosen will typically produce sub-optimal solutions, as we commit to certain decisions before understanding their impact at a later stage. We selected the given approach based on the development time frame for the competition, our own, limited development time, and the rather tight run-time limits for the problem instances. The results indicate that good, if not exceptional results can be achieved by this method.

We now briefly describe the clustering and stack building modules, before starting on the description of the master problem and the cut generation.

3.1 Clustering

Each problem instance contains items of different sizes and product types, coming from potentially many different vendors, and being delivered to different parts of a factory. For each of the items we can decide from the input data which planned trucks can potentially carry it. We can then decompose the overall problem into independent clusters by exploiting this information. We introduce an undirected graph with vertices being the planned trucks. Two nodes (trucks) are connected by an edge, if there is an item that can be transported on either of the trucks. We find all connected components of this graph, each of the components is an independent sub-problem, which consists of a set of planned trucks that need to be studied together. In the competition data, we note there often are trucks which cannot transport any of the given items, these trucks can be ignored in the solution process. We typically find a large number of very small clusters (often of only one planned truck), and relatively few large clusters (some containing more than half of all trucks).

3.2 Stack Building

For each cluster, we generate a set of stacks by considering all items that need to arrive at the factory at the same time. Currently we place items together on a stack if they have the same arrival time and the same "stackability". This is a property defined in the input data to decide if two items with the same footprint (length and width) can be placed together in the same stack. This mainly depends on the products used. Building stacks basically consists in solving a series of bin-packing problems, one for each stackability and orientation, with side constraints. We have defined an optimization model to find the optimal assignment, but found that due to the large number of instances we cannot afford to run this model. Instead, we use a greedy heuristic, based on a first-fit decreasing bin packing strategy, which includes the required side constraints. We use some trivial lower bounds to check the quality of the solution, which seems to indicate that we produce solutions with a minimal number of stacks in most cases. Note that we often create multiple, identical stacks using the same items, in order to carry the required quantity of all items.

Once the stacks are generated, we calculate which trucks can transport each stack. This is determined by the intersection of the time-windows of the all items in the stack, but also considers possible differences in the size (especially height) of the trucks, and constraints on the weight on the bottom of the truck, which can vary between different truck types. While we need to check these constraints to ensure we only allow feasible assignments, in the given problem instances these extra constraints only affect a very small number of stacks.

4 MIP Master Problem: Assigning Stack Types in Time

We now consider the problem of assigning stacks of different types to planned trucks, within one cluster defined earlier. We group the stacks by type, so that we avoid the potential symmetries when we encounter multiple, equivalent stacks with the same properties. Stacks which have the same size, height, forced orientation, load stage, total weight, inventory cost, and list of feasible trucks can be combined into multiple copies of the same stack type.

4.1 Input Data

We denote the set of stack types by I, and use index i to denote a specific stack type. We use the set J for the set of all potential planned trucks in the cluster, using index j for specific trucks. We can also interpret set J as the collection of possible arrival time points. We use the set S for the different size combinations of the stacks to be placed, and the set O of values *none*, *widthwise* and *lengthwise* for the possible forced orientations of the stacks. We use the following notation for data items, most of these values are themselves computed from other input data of the competition.

n_i number of stacks of stack type i
u_j upper bound on the number of extra trucks that can be used at time point j
T_i the set of possible trucks on which stacks of type i can be placed
o_i forced orientation of stack i, value *none* if not constrained
s_i the size type of stack i
w_i the total weight of stack i, which is the sum of the item weights in the stack
f_i the area of the footprint of the stack, e.g. $1200 * 1000$ for a stack of size $1200x1000$ (sizes are expressed in mm).
t_j the truck type of truck j, given by the combination of length and width
g_j the area of the load-bed of truck j, e.g. $13500 * 2440$ for a truck of type $13500x2440$
v_j the total allowed weight of items placed in truck j
a_j the cost of using the planned truck for time point j
b_j the cost of using an additional truck for timepoint j
c_{ij} the earliness cost of transporting one stack of type i with truck j

The earliness cost of a stack is the sum of the earliness costs of its items. The set of possible trucks for a stack is based on the possible trucks for each item on the stack, together with additional constraints that apply on the stack (height, density).

4.2 Variables

We express the problem in terms of stack types, and not individual stacks, to avoid the symmetries introduced by multiple stacks with the same properties

We introduce integer variables $x_{ij} \in [0, n_i]$ to indicate how many stacks of type i we transport at time j. We have 0/1 variables $p_j \in \{0, 1\}$ which states if we use the planned truck at time j. We add integer variables $q_j \in [0, u_j]$ which indicate how many extra trucks we use at time j. The upper bound u_j must be large enough to allow for enough extra trucks when they are needed for a given time point.

While we can express the basic constraints of the model using only these variables, we also introduce some integer counting variables y_{jso} which tell us how many stacks of size s and forced orientation o we transport at time j.

There will be additional, implicit 0/1 variables introduced in the model when we use disjunctions as cuts to remove infeasible size combinations later on.

4.3 Constraints

We now state the constraints of the MIP model.

The first constraint states that the number of stacks of a given type allocated at the different time points must be equal to the total number n_i of stacks of that type.

$$\forall_{i \in I}: \quad \sum_{j \in J} x_{ij} = n_i \tag{1}$$

Stacks can only be placed on compatible trucks, so many of the x_{ij} values will be zero.

$$\forall_{i \in I} \forall_{j \in J \text{ s.t. } j \notin T_i}: \quad x_{ij} = 0 \tag{2}$$

We can only use additional trucks at time j if we use the planned truck for that time-point as well. As the q_j variables are integer and not just 0/1, we need to express this condition by a weighted inequality. We can use the upper bound on their domain u_j as an appropriate *big-M* value.

$$\forall_{j \in J}: \quad q_j \leq u_j p_j \tag{3}$$

For any time point, the capacities of the selected trucks must be respected. We have the constraint for weight

$$\forall_{j \in J} \quad \sum_{i \in I} x_{ij} w_i \leq (p_j + q_j) v_j \tag{4}$$

and another set of constraints for the area used by the stacks

$$\forall_{j \in J}: \quad \sum_{i \in I} x_{ij} f_i \leq (p_j + q_j) g_j \tag{5}$$

Note that we use the effective floor area of the truck for g_j in order to strengthen the constraint. The effective length (width) of a truck is defined by considering all possible lengths (widths) of stacks that can be loaded on the truck, and finding the largest integer value below the truck length (width) that can be reached by combination of those values. This is computed by dynamic programming.

We need to link the x and y decision variables, by counting the number of stacks of a given size and orientation, using the notation s_i for the size type of stack type i, and o_i for the forced orientation of stack type i.

$$\forall_{j \in J} \forall_{s \in S} \forall_{o \in O}: \quad y_{jso} = \sum_{i \in I \text{ s.t. } s_i = s \wedge o_i = o} x_{ij} \tag{6}$$

4.4 Cuts

Given only the constraints stated so far, the MIP solution will tend to try and place too many stacks on a truck, as it ignores that placing stacks of certain sizes and orientation will often result in unusable space that cannot be filled by the other stack types. We can use the counting variables to express cuts that remove infeasible size combinations. These take two forms, a linear combination, or a disjunctive constraint.

The linear cuts take the form

$$\forall_{j \in J}: \quad \sum_{s \in S} \sum_{o \in O} d_{so}^{t_j} y_{jso} \leq e^{t_j} (p_j + q_j) \tag{7}$$

We describe later how we obtain valid coefficients $(d_{so}^{t_j}, e^{t_j})$ for the different cuts generated. Note that while in principle we can allow any linear combination of the counting variables, in practice we only generate such cuts for a limited number (up to four) of non-zero size and orientation combinations. It is important to note that we only introduce valid cuts, which do not eliminate feasible size combinations.

Unfortunately, some infeasible size combinations are inside the convex hull of the feasible combinations. In order to also remove these points, we introduce disjunctive constraints. If we know that some size combination $\bar{r}$ is infeasible for a given truck type t, then increasing any of its component values, adding an extra stack of that type, is also infeasible. We can therefore exclude the point by a disjunction stating that at least one of the types occurs in a quantity smaller than r^{t_j}, or we use additional trucks. This leads to disjunction of the form:

$$\forall_{j \in J}: \quad \bigvee_{s \in S, o \in O} (y_{jso} \leq r^{t_j} - 1) \vee (q_j >= 1) \tag{8}$$

Note that our selected MIP solver will automatically convert the disjunctions into a set of new binary variables, and modified inequalities.

4.5 Objective

The cost function consists of three elements, the cost of the chosen planned trucks, the cost of any created additional trucks, and the inventory cost of delivering some stacks before their latest delivery date.

$$\min \sum_{j \in J} a_j p_j + \sum_{j \in J} b_j q_j + \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} \tag{9}$$

Note that the placement routine may increase the number of additional trucks required (and therefore the cost), if the placement was not able to fill all assigned stacks in the provided planned and addition trucks.

5 Lower Bound

We now consider a lower bound on the number of trucks (of a given length and width) needed to fit a set of stacks of given sizes and orientation. The lower bound only considers the area and orientation of the stacks to be placed, and ignores stacks considered too small. We use the following idea: Some stacks are large enough to use more than half the width of the given truck, regardless of their orientation. These stacks can be placed in any order, but must be sequentially in the truck, this requires at least the sum of their minimal lengths. Other stacks are large enough so that we cannot place three of them at the same x-position in the truck. These stacks are those whose minimal width is between one-third and one half of the truck width, and which cannot be placed together with the stacks of the first group. Any placement of the stacks of the second type can use at most two stacks next to each other. We can therefore split these stacks into two groups, the left and the right group. We can use dynamic programming to find the best distribution of all these stacks to the two groups, the total space taken by the stacks is the space required by the items in the first group, and the smallest maximum of the lengths required for the left and right groups. When we compare this to the total length of the truck, we see if the stacks will fit into one truck or not.

6 Deriving Cuts

We apply the lower bound calculation for each planned truck and its assigned stacks in the solution of the master problem. If the lower bound finds that we cannot pack all required stacks in the given trucks, then we need to generate a cut in the master problem to exclude this choice in the future. While we could possibly just generate no-good cuts to exclude a specific assignment, we can produce stronger cuts that reject more infeasible assignments. If, for example, we find that we cannot pack k items of some type, then we also know that we cannot pack $k+1$ items of the type. Alternatively, if we exclude stacks of a given size, then all stacks of a larger size also do not fit in the given space. The idea is to allow all feasible combinations of stack sizes, while excluding the infeasible combination that was flagged by the lower bound calculation.

We now describe how the cuts are generated. We first tackle the case of two size types, leading to a two-dimensional cut. In the 2D case, we produce the set of facets of the polyhedron which is spanned by the feasible points. In higher dimensions, this might be too complex, and we only produce separating cuts which split the feasible from the given infeasible point.

6.1 Two Dimensional Cuts

The procedure for two dimensions is as follows. We first generate all feasible combinations of the two types to fill the given truck size, then we produce the convex hull of the set of all feasible points. We can convert the edges of the convex hull into a set of inequalities of the form $ax + by \leq c$, and add these cuts to the master model for the instances where the constraint is not already satisfied. We use an example to illustrate the procedure.

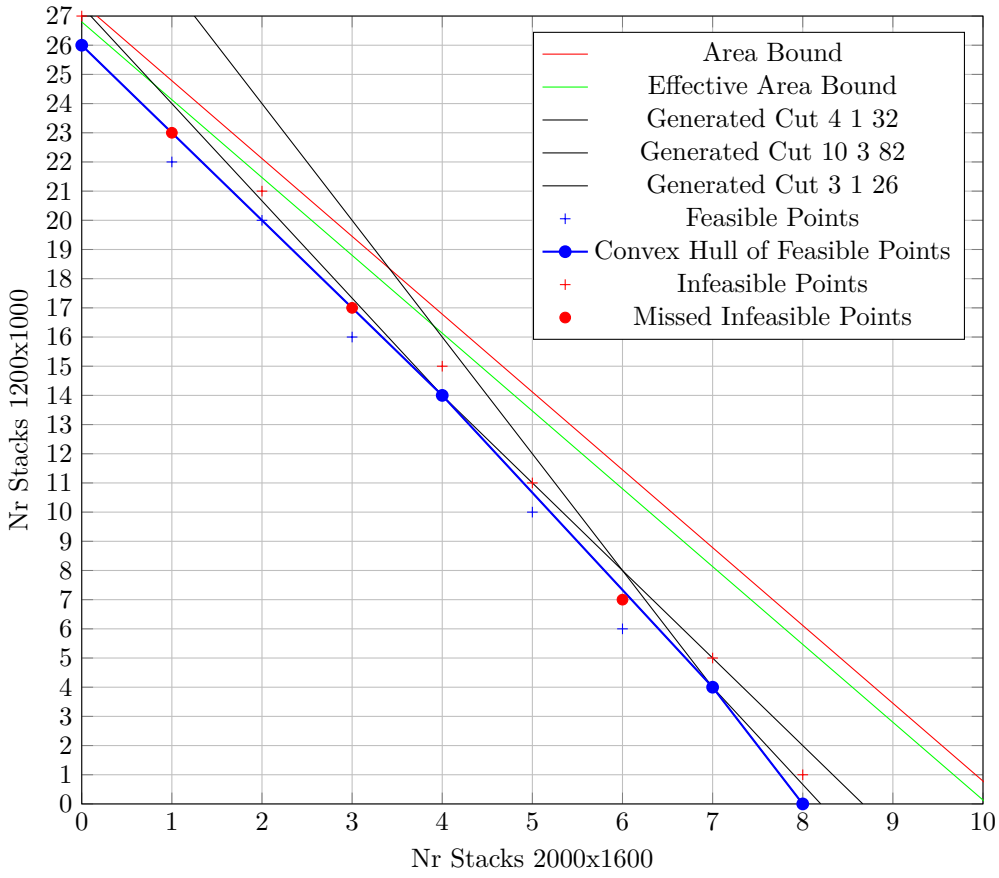
Figure 2 shows the situation for an example problem of packing stacks of sizes $2000x1600$ and $1200x1000$ into a truck of size $13500x2440$. If we just consider the available area, we would consider all integer point to the left of the red area bound line. If we strengthen this constraint by considering the effective size of the truck for the given sizes, we obtain the green effective area bound. But this includes many infeasible combinations, for example (9,0) and (7,7). If we compute all (maximal) feasible points, we obtain the blue points. Taking the convex hull of these points (and (0,0)), we find four corners, denoted by the blue circles. The linking vertices of the convex hull define three hyperplanes of the form $ax + by \leq c$, with integer coefficients (4,1,32), (10,3,82), and (3,1,26). These cuts eliminate many, but not all infeasible combinations, while retaining all feasible size combinations. When we consider the minimal infeasible combinations that are left, marked as red points, we find that three points, marked as red circles, are on or inside the convex space defined by the linear cuts. In order to also remove these points, we need to use disjunctions. Any point above or to the right of a red point is also infeasible, so the constraint for the infeasible point (6, 7) is $x \leq 5 \vee y \leq 6$.

6.2 Higher Dimensions

While in two dimensions we can compute the convex hull in polynomial time, and generate the facet-defining cuts based on the segments of the convex hull, the number of facets in higher dimensions may be too high to compute exhaustively. But we can define a simpler cut which separates an infeasible point from all feasible points by an inequality. We still need the set of (maximal) feasible points, which may already be expensive to generate for higher dimensions, e.g. combining five or more sizes in a single cut.

Given a set F of feasible points $\bar{f}$ of dimension $|D|$, and an infeasible point $\bar{i}$, also of

Figure 2 Cuts for Packing Stacks of Sizes 2000x1600, 1200x1000 in Truck 13500x2440



dimension $|D|$, find positive integer coefficients a_d and a right-hand side r with the following optimization model.

$$\min \sum_{d \in D} a_d + r \tag{10}$$

such that

$$\forall_{\bar{f} \in F} : \quad \sum_{d \in D} a_d \bar{f}_d \leq r \tag{11}$$

$$\sum_{d \in D} : \quad a_d \bar{i}_d \geq r + 1 \tag{12}$$

We try to find the smallest integer coefficients for the cut that separates the feasible points from the infeasible point. As all feasible and infeasible points are on integer grid points, we can ask for the parameters of the cut to be integral as well. This model may not have a solution, if the infeasible point lies inside the convex hull spanned by the feasible points. In that case we can state a disjunctive constraint

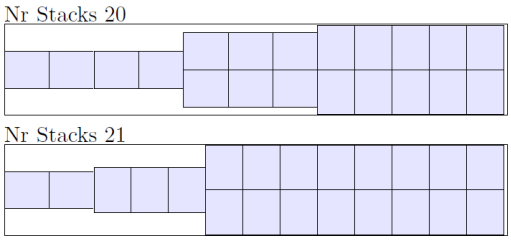
$$\bigvee_{d \in D} (x_i \leq i_d - 1) \tag{13}$$

which eliminates the infeasible point at the cost of introducing implied new binary variables.

7 Placement

We use a number of heuristics to create actual placements for a set of assigned stacks. There are different variants that try to solve specific sub-problems, for example the case when all assigned stacks have the same size and orientation. In that case, we can use precomputed placement solutions to optimally place the stacks in the available space, trying to use any empty space to limit the mid-axle weight of the placement. Figure 3 gives an example of placing 20 or 21 stacks of size $1200x1000$ into a truck of size $13500x2440$. Different load stages will be handled by assigning earlier stages to the left, while the axle weight is reduced by pushing heavier items to the right.

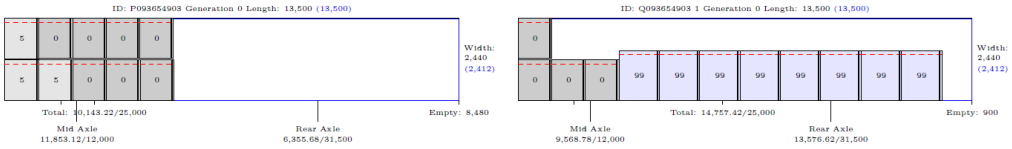
Figure 3 Placement Template of Stacks of Size $1200x1000$ in Truck of Size $13500x2440$



A more generic placement routine is used when there are multiple stack sizes and load stages, the heuristic tries to fill the truck from left to right, providing support for each stack placed, while respecting the load order, and trying to reduce the mid-axle weight by placing heavier items to the right. Different combinations of options are used to create a

set of potential placements. Each of the placements is then evaluated for the axle-weight constraints, and the best resulting solution is kept. For debugging and visualization we also keep so-called counter-factuals, which show which better placement solutions have not been selected. Figure 4 shows an example from a generated report on the solution, colours of stacks indicate load stages. The selected placement requires two trucks, the counter-factual shows a solution using a single truck, but which violates the mid-axle weight constraint.

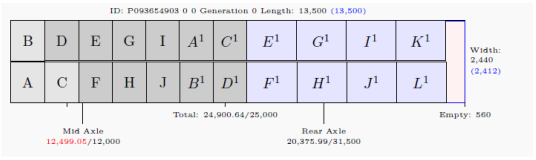
Figure 4 Counter-factual Example: Placement below violates axle-weight constraint, but uses one less truck



2.3.1 Counter Factuals

Table 10: Counter Factuals for P093654903

Name	Truck	Cost	Nr Errors	Excess Weight	Assigned Trucks	Excess Trucks	Assignment Strategy	Orientation Strategy	Order Strategy
P093654903 0	P093654903	1500.00	2	1984.47	1	0	Corners	Widthwise	Size
P093654903 1	P093654903	1500.00	2	1984.47	1	0	Corners	Preference	Size
P093654903 2	P093654903	1500.00	2	1984.47	1	0	Corners	Widthwise	Weight



8 Post-Optimization

When we have run the master problem and the placement problems for each truck, we have a feasible solution. If the placement did not require any additional trucks, we have an optimal solution for our decomposition, but usually not for the overall problem. But in most cases we need additional trucks to complete the placement, as our lower bound calculation only considers the stack sizes, but not load stages, nor axle weight constraints. In addition, it is not complete, as it does not handle all size combinations. Most of the extra trucks created will be nearly empty, with only some overflow placed in the extra truck. On the other hand, the cost of these extra trucks is quite high, we should therefore aim to remove them. The easiest way to achieve this is to move the stacks placed in these trucks to earlier or later trucks, where we still have some spare capacity. We have defined a local search which identifies such moves, and then picks the best subset of all possible moves to reduce the overall cost. This is implemented as a set partitioning model, solved with the MIP solver. This is very likely an overly complex approach, and a simple, greedy local search working at the stack or item level might produce better results. Time constraints stopped us from implementing such a solution.

9 Experimental Evaluation

The instances provided by the Roadef Challenge are grouped by the different competition stages (datasets A, B, C, and X) and consist of different problem families, which correspond to different plants of Renault around the world. We use the instances of family AS in our

evaluation. Table 1 gives an overview of the instances, the full data are available on the competition website. Datasets A and B consist of smaller problem instances, from 8,000 to 15,000 pieces to be transported, while the instances in datasets C and X contain around 50,000 items. There are up to 2,000 planned trucks available. We also show the number of stacks that were created by our stack building method, and the number of clusters computed by our clustering method, together with the number of trucks and stacks in the largest cluster of each instances. This defines the number and the size of the master MIP problems that we need to solve to obtain a solution. In the last column, we show the ratio of inventory cost to transport cost required for the instance, this controls the weight factors in the Objective 9.

Table 1 Instance Data Overview

Dataset	Instance	Pieces	Planned Trucks	Stacks	Clusters	Largest Nr Trucks in Cluster	Largest Nr Stacks in Cluster	Inventory Transport Cost Factor
A	AS	8,402	656	3,658	81	236	1,354	10:1
B	AS	15,707	2,037	7,203	207	642	2,972	1:1
B	AS2	15,707	2,037	7,203	207	642	2,972	5:1
C	AS	52,264	1,593	19,588	129	1,352	11,064	1:1
C	AS2	48,193	1,490	17,910	119	1,256	10,708	5:1
C	AS3	48,193	1,490	17,910	119	1,256	10,708	1:5
X	AS	51,025	1,474	19,279	375	1,453	10,310	1:1
X	AS2	51,025	1,474	19,279	375	1,453	10,310	5:1
X	AS3	51,025	1,474	19,279	375	1,453	10,310	1:5
X	AS4	51,923	1,408	20,206	367	1,484	10,535	1:1
X	AS5	51,923	1,408	20,206	367	1,484	10,535	5:1
X	AS6	51,923	1,408	20,206	367	1,484	10,535	1:5
X	AS7	51,923	1,408	20,206	367	1,484	10,535	1:1

For time and space reasons we did not perform a complete evaluation of the proposed method against all problem instances of the competition, but show only instances of the AS family. In Table 2 we present the results obtained by the current method of this paper, the results obtained by our submission to the final stage of the competition, and the best known result for each instance from the Roadef website. Please note that the best known results for dataset A and B only reflect earlier stages of the competition. We also show the gap between the submitted version and the best known result, the gap between the current version and the best known result, and the improvement of the current result compared to our submitted version.

For our current results, we show the aggregated lower bound of the MIP problems, the total cost of the MIP solutions, and the cost of the resulting placement. We also note the number of excess trucks, i.e. how many more trucks the placement needs to find a feasible solution, compared to the assignment given by the master problem. All results include all generated cuts. We see that for the smaller problems we are able to add enough cuts to find a master problem solution which requires very few excess trucks, leading to an overall solution better than the best published values. The results are not quite as good for the larger problem instances, where our solutions are still inferior to the best known results, while providing a significant improvement over our submitted results for the competition. This comes at the cost of a significantly increased run-time, caused by the cut generation.

Table 3 illustrates the impact of the cut generation on instance X/AS. We compare results when no cuts are generated, when only the convex, linear cuts are produced, and when all cuts, including the disjunctive cuts, are generated. Adding the linear cuts increases both the lower bound and the solution cost for the master MIP problems, but results in a much lower cost of the placement. This gap cannot be fully made up by the local search, so that the end result is a significantly improved solution, at a moderate run-time increase. Adding all cuts

Table 2 Results for Instance Family AS in Datasets A, B, C, and X

Dataset/ Instance	MIP Bound	MIP Solution	Placement Solution	Excess Trucks	After Local Improvement	Submitted to Final Stage	Best Known	Gap Submitted to Best	Gap Current to Best	Improvement over Submitted	Time (s)
A/AS	392,176	393,060	393,060	0	393,060	400,860	395,040	1.47%	-0.50%	-1.95%	10
B/AS	896,895	900,404	909,404	5	907,265	974,505	912,146	6.84%	-0.54%	-6.90%	1,687
B/AS2	973,520	975,580	979,180	2	977,430	1,036,525	988,795	4.83%	-1.15%	-5.70%	79
C/AS	1,732,078	1,814,173	1,915,063	87	1,814,173	1,982,627	1,737,148	14.13%	4.43%	-8.50%	24,184
C/AS2	1,657,848	1,689,215	1,863,815	97	1,765,660	1,885,970	1,667,920	13.07%	5.86%	-6.38%	17,985
C/AS3	7,803,344	7,960,178	8,635,178	75	8,174,542	8,997,823	7,797,902	15.39%	4.83%	-9.15%	17,673
X/AS	1,918,919	1,930,494	2,049,294	66	2,002,947	2,147,194	1,912,475	12.27%	4.73%	-6.72%	7,237
X/AS2	2,082,317	2,095,440	2,176,440	45	2,134,420	2,274,020	2,083,300	9.15%	2.45%	-6.14%	5,238
X/AS3	9,243,839	9,314,719	9,971,719	72	9,698,956	10,357,996	9,196,363	12.63%	5.47%	-6.36%	7,923
X/AS4	1,944,754	1,958,440	2,077,240	65	2,017,296	2,153,864	1,940,261	11.01%	3.97%	-6.34%	8,590
X/AS5	2,116,552	2,129,875	2,230,675	56	2,171,620	2,299,150	2,113,325	8.79%	2.76%	-5.55%	5,330
X/AS6	9,355,990	9,433,229	9,973,229	59	9,708,738	10,419,636	9,297,347	12.07%	4.42%	-6.82%	8,728
X/AS7	1,912,888	1,929,533	2,067,533	92	2,003,258	2,109,281	1,914,552	10.17%	4.63%	-5.03%	9,406

increases the MIP bound and solution only marginally, but still leads to a better placement, and final result. This comes at the cost of a significantly increased run-time, which no longer fits in the competition time limit. This increase is caused by both an increasing complexity of the master problem, as more disjunctive cuts add more and more 0/1 variables, and by an increased number of iterations needed to reach a fix point.

Table 3 Impact of Cut Level on Solution Quality for Instance X/AS

Cut Level	MIP Bound	MIP Solution	Max Iterations	Placement	Excess Trucks	Result	Time (s)
No Cuts	1,880,938	1,894,162	-	2,455,762	312	2,136,968	1,379
Convex Only	1,918,031	1,929,144	7	2,103,777	97	2,017,089	2,521
All Cuts	1,918,919	1,930,494	35	2,049,294	66	2,002,947	7,237

Our submitted version used basically the same approach, but did not implement the cut generation, so that the master model only used the area and weight constraints to assign stacks to trucks. If the placement did not find a solution with the number of trucks computed, additional trucks would be used in the placement, and the local search routine was attempting to redistribute the nearly empty trucks that were created. The current cut generation provides up to 9% improvement over this earlier model.

10 Conclusion

This paper describes our current model for the Roadef Challenge 2022, which adds a cut generation based on a lower bound calculation to the master MIP model. This results in a significant improvement of the cost values, while not quite reaching the best results obtained in the competition. There are probably multiple reasons for this. Our decomposition chosen may exclude some solutions that can be found with a more flexible technique which avoids fixing the stack structure from the start. The time allocation could be improved to more accurately predict the time needed to find high-quality solutions to sub-problem, and spend the limited run-time for better effect. The cuts generated are purely based on size consideration, and ignore load stage and axle weight constraint completely. This has the advantage that the cuts generated can be applied to any truck of the same size, and can be reused across instances. But more placement specific cuts could be found that help to improve the master solution quality and lead to fewer excess trucks in the placement. On the other hand, the lower bound calculation ignores smaller stacks completely, and therefore is not guaranteed to be exact. Further work would be required to improve the lower bound reasoning. Finally, our placement portfolio does not always find the best possible placement, we could study improvements especially for problems with many load stages and/or high item weights.

References

1 Christian Serrano Alain Nguyen, Mohamed-Amine Khatouf. Challenge euro/roadef 2022 truck loading, 2022. https://www.roadef.org/challenge/2022/en/index.php.

2 Christian Serrano Alain Nguyen, Mohamed-Amine Khatouf. Challenge euro/roadef 2022 truck loading rules, 2022. https://www.roadef.org/challenge/2022/en/index.php.

3 Sara Ali, António Galrão Ramos, Maria Antónia Carravilla, and José Fernando Oliveira. On-line three-dimensional packing problems: A review of off-line and on-line solution approaches.

Computers and Industrial Engineering, 168:108122, 2022. URL: https://www.sciencedirect.com/science/article/pii/S0360835222001929, doi:https://doi.org/10.1016/j.cie.2022.108122.

4 Maria Teresa Alonso, Ramon Alvarez-Valdes, Francisco Parreño, Jose Manuel Tamarit, et al. Algorithms for pallet building and truck loading in an interdepot transportation problem. *Mathematical Problems in Engineering*, 2016, 2016.

5 Alireza Amini, Reza Tavakkoli-Moghaddam, and Aschkan Omidvar. Cross-docking truck scheduling with the arrival times for inbound trucks and the learning effect for unloading/-loading processes. *Production & Manufacturing Research*, 2(1):784–804, 2014.

6 Andreas Bortfeldt and Jörg Homberger. Packing first, routing second—a heuristic for the vehicle routing and loading problem. *Computers & Operations Research*, 40(3):873–885, 2013.

7 Andreas Bortfeldt and Gerhard Wäscher. Container loading problems: A state-of-the-art review. *Working Paper Series*, 2012.

8 Andreas Bortfeldt and Gerhard Wäscher. Constraints in container loading–a state-of-the-art review. *European Journal of Operational Research*, 229(1):1–20, 2013.

9 André Augusto Ciré, Elvin Coban, and John N. Hooker. Logic-based benders decomposition for planning and scheduling: a computational analysis. *Knowl. Eng. Rev.*, 31(5):440–451, 2016. doi:10.1017/S0269888916000254.

10 Gianluca Costa, Maxence Delorme, Manuel Iori, Enrico Malaguti, and Silvano Martello. Training software for orthogonal packing problems. *Comput. Ind. Eng.*, 111:139–147, 2017. URL: https://doi.org/10.1016/j.cie.2017.06.036, doi:10.1016/J.CIE.2017.06.036.

11 Jean-François Côté, Mohamed Haouari, and Manuel Iori. Combinatorial benders decomposition for the two-dimensional bin packing problem. *INFORMS Journal on Computing*, 33, 01 2021. doi:10.1287/ijoc.2020.1014.

12 Maxence Delorme, Manuel Iori, and Silvano Martello. Bin packing and cutting stock problems: Mathematical models and exact algorithms. *Eur. J. Oper. Res.*, 255(1):1–20, 2016. URL: https://doi.org/10.1016/j.ejor.2016.04.030, doi:10.1016/J.EJOR.2016.04.030.

13 Maxence Delorme, Manuel Iori, and Silvano Martello. Logic based benders' decomposition for orthogonal stock cutting problems. *Comput. Oper. Res.*, 78:290–298, 2017. URL: https://doi.org/10.1016/j.cor.2016.09.009, doi:10.1016/J.COR.2016.09.009.

14 Maxence Delorme, Manuel Iori, and Silvano Martello. BPPLIB: a library for bin packing and cutting stock problems. *Optim. Lett.*, 12(2):235–250, 2018. URL: https://doi.org/10.1007/s11590-017-1192-z, doi:10.1007/S11590-017-1192-Z.

15 J. N. Hooker. Logic-based benders decomposition for large-scale optimization, 2019. arXiv:1910.11944.

16 Leonardo Junqueira, Reinaldo Morabito, and Denise Sato Yamashita. Mip-based approaches for the container loading problem with multi-drop constraints. *Annals of Operations Research*, 199:51–75, 2012.

17 António G Ramos, Elsa Silva, and José F Oliveira. A new load balance methodology for container loading problem in road transportation. *European Journal of Operational Research*, 266(3):1140–1152, 2018.

18 Christian Serrano, Xavier Delorme, and Alexandre Dolgui. Scheduling of truck arrivals, truck departures and shop-floor operation in a cross-dock platform, based on trucks loading plans. *International Journal of Production Economics*, 194:102–112, 2017.

19 Helmut Simonis and Barry O'Sullivan. Search strategies for rectangle packing. In Peter J. Stuckey, editor, *Principles and Practice of Constraint Programming, 14th International Conference, CP 2008, Sydney, Australia, September 14-18, 2008. Proceedings*, volume 5202 of *Lecture Notes in Computer Science*, pages 52–66. Springer, 2008. doi:10.1007/978-3-540-85958-1_4.

20 Helmut Simonis and Barry O'Sullivan. Almost square packing. In Tobias Achterberg and J. Christopher Beck, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems - 8th International Conference, CPAIOR 2011,*

Berlin, Germany, May 23-27, 2011. Proceedings, volume 6697 of *Lecture Notes in Computer Science*, pages 196–209. Springer, 2011. doi:10.1007/978-3-642-21311-3_19.

21 Túlio A.M. Toffolo, Eline Esprit, Tony Wauters, and Greet Vanden Berghe. A two-dimensional heuristic decomposition approach to a three-dimensional multiple container loading problem. *European Journal of Operational Research*, 257(2):526–538, 2017. URL: https://www.sciencedirect.com/science/article/pii/S0377221716305707, doi:https://doi.org/10.1016/j.ejor.2016.07.033.

22 Mario C Vélez-Gallego, Alejandro Teran-Somohano, and Alice E Smith. Minimizing late deliveries in a truck loading problem. *European Journal of Operational Research*, 286(3):919–928, 2020.

23 Xiubin Wang and Amelia C Regan. Local truckload pickup and delivery with hard time window constraints. *Transportation Research Part B: Methodological*, 36(2):97–112, 2002.