Model	Original size ineqs (eqs) / vars / nzs. / dens.	Original, presolved rows / cols / nzs. / dens.	Projected size ineqs / vars / nzs. / dens.
No weights	774 (12) / 1142 / 6662 / 8.61	554 / 657 / 2784 / 5.03	20 / 12 / 155 / 7.75
No hydro.	806 (43) / 1173 / 7854 / 9.74	555 / 657 / 3441 / 6.20	18 / 12 / 144 / 8.00
2 parts	810 (43) / 1173 / 7860 / 9.70	556 / 661 / 3447 / 6.20	96 / 12 / 1113 / 11.59
4 parts	824 (49) / 1179 / 7886 / 9.57	564 / 671 / 3471 / 6.15	64 / 12 / 731 / 11.42
6 parts	838 (55) / 1185 / 7916 / 9.44	570 / 679 / 3496 / 6.13	80 / 12 / 888 / 11.10
8 parts	852 (61) / 1191 / 7950 / 9.33	576 / 685 / 3522 / 6.11	52 / 12 / 582 / 11.19
Naive model	3 / 12 / 36 / 12.00	3 / 9 / 24 / 8.00	

Table 1: Size of systems (models), before and after projections. Size of naive model for comparison.

1 Results

1.1 What has been done

We have considered a number of different models, where the weight and hydrostatics are taken into account to various degrees. The first model does not consider the weight of the containers at all, the second model only considers the total weight-limit, and the subsequent models consider the hydrostatic constraints at 2, 4, 6 and 8 measure points, respectively.

Each model has been projected, such that only the variables denoting the number of each type of containers is left, that is, all variables but $\{ y_\tau \mid \tau \in T \}$ [check name] are eliminated. Projections have been done in two different ways, *flat* and *decomposed*. For the decomposed projections, a tree structure has been used as described in Section **??**, while the flat projection does not use any decomposition at all. The particular used decompositions can be found in the appendix.

The original and the projected models have been optimized for revenue, and objective value as well as time taken has been compared. The latter is measured in both iterations and *ticks* as it appears when solved with cplex version... Ticks is a deterministic time measure that according to the software provider "yields the same level of performance for repeated solving of the same model with the same parameter settings, on the same computing platform"[1]. The revenue is dependent on the container type, so that "limiting" container types yields higher revenue.

The projections were done on a ... computer with ... .

1.2 Size

The results of the mentioned projections are summarized in Table 1. The table shows the size of the original (unprojected) models and the projected model (found by decomposition), given in terms of the number of inequalities (ineqs), variables (vars), non-zero entries (nzs.) and density (dens.). The size of the original model is given both as they appear as input to our algorithm, but also after a competitive preprocessing, namely after cplex has preprocessed it.

The numbers show that the projected models have much fewer inequalities and variables, namely app. 6 - 30 times fewer inequalities and 54-57 times fewer variables than the presolved systems. The projected systems also have fewer non-zero entries (between 3 and 24 times fewer), though, the projected models are more dense (app. 1.3 to 1.9 more dense than the presolved systems).

The results reveal no apparent [/direct] relationship between the size of the original model and the size of the projection, which probably have to do with both the actual position of the hydrostatic measure points and the undeterministic behaviour of the removal of almost redundant inequalities.

[1] https://www.ibm.com/support/knowledgecenter/SSSA5P_12.5.0/ilog.odms.studio.help/CPLEX/ReleaseNotes/topics/releasenotes125/newDetTime.html

Model	Time			
	decomp.	vars left	flat	vars left
No weights	24.5m	-	2.5m	-
No hydro.	14.5m	-	1.8m	-
2 parts	7h 18m	-	(TO) 32h	551
4 parts	8h 4m	-	(TO) 61h	557
6 parts	3h 7m	-	(TO) 18h	577
8 parts	3h 19m	-	(TO) 65h	566

Table 2: Size of projections

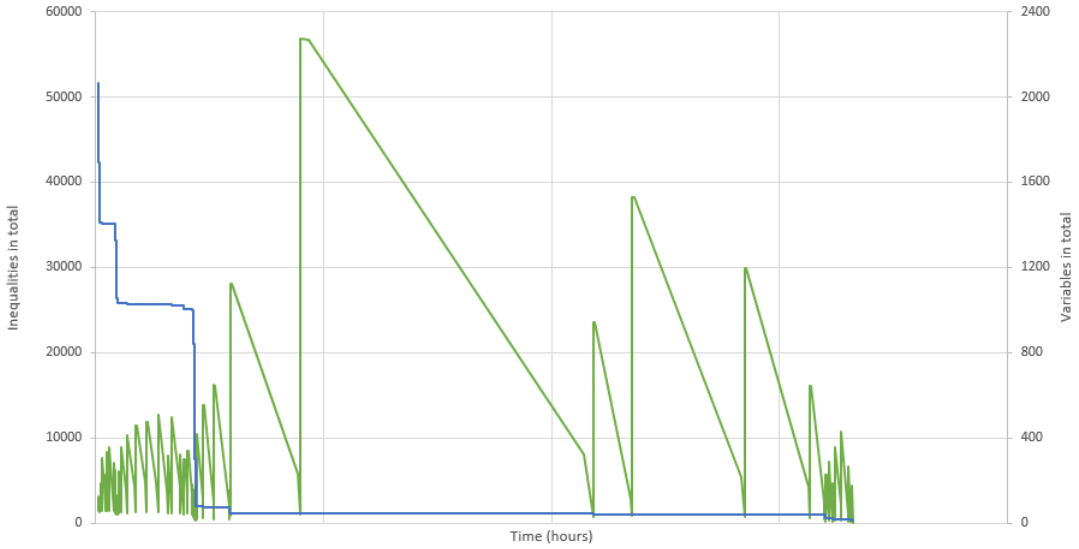

Figure 1: The inequalities growth and variable decrease for the 8 part-model with decomposition.

1.3 Decomposition

Table 2 shows the time taken for the algorithm to do the projection, both decomposed and flat. For most models, the flat projection timed out (TO), in which case the time limit and the variables left to be projected is given. Figure 1 shows the progression of the number of inequalities and variables, respectively, as the algorithm runs on the decomposed 8 part-model. Likewise, Figure 2 shows the progression for the flat projection of the same model; this figure includes the number of inequalities for the decomposed projection for comparison. Each graph shows the number of inequalities and variables after the preprocessing step, each Gauss-elimination, and each FM-elimination followed by some preprocessing, parallel redundancy removal and sequential removal of almost redundant inequalities.

When considering each subsystem in a decomposition as a system in itself, in generel, the number of inequalities after the FM step grows, as well as the number of inequalities before the FM step, until there are a few variables left, where both these numbers decrease. Notice, that the graph in Figure 1 shows the total run of the projection algorithm, causing this pattern to be repeated.

For the decomposed algorithm, though the number of inequalities grow after each FM-step, most of them are redundant or almost redundant. The same does not hold for the flat-version for the cases where hydrostatics are taken into consideration. Instead, many of the produced inequalities are non-redundant which increases the likelyhood that even more inequalities will be produced in the next elimination, but also that the redundancy removal will take longer time - not only are there more inequalities to check, each check will also take longer. Though this just speculations, it appears that the flat projection of the total system on a “global” scale behaves as a “enlarged” version of described pattern for the subsystems.

This behaviour can be explained by the occurence of several “interacting” transverse/global inequalities (as described previously), since the algorithm projects a few variables from one subsystem (resulting in a little denser

Figure 2: The inequalities growth for the 8 part-model with and without decomposition.

global constraints) before moving on to the next subsystem from where it removes another few variables etc., all the while it postpones dealing with the variables in the global inequalities which get more and more dense.

It is clear that the decomposition has a huge impact on the success of the projection algorithm when the model/system has a none-negligible number of "interacting" global inequalities.

It should be mentioned that it is possible, that other orderings of the variables (caused by other heuristics then the used greedy heuristic) in these cases could potentially lead to manageable flat projections, but testing this is outside the scope of this paper. [It should likewise be mentioned that the runtime, even for the decomposed projections, are not exactly small. The main part of the execution time is spend doing redundancy removal.]

1.4 Revenue optimization

The original and the projected models have been optimized for revenue using cplex. Each transported container yields a revenue which is based on its type, i.e. on the size, weight and refer-property of the container types. More specifically, for our tests, reefer containers yields the double revenue as a similar non-reefer container, 40' containers have a revenue which is 1.5 times higher as a similar 20' container, while containers are more expensive the heavier it is. [For a 20', non-reefer container, with a weight, respectively of 6, 21 and 27 t, the revenue is, respectively set to 100, 600 and 700 $.]

Table 3 shows the number of iterations, the deterministic time and the objective value found by cplex when optimizing revenue for the original and projected models. It likewise how many times faster, the projections are w.r.t. iteration and ticks, as well as the difference in objective value in percentage. For comparison, te number of iterations, deterministic time and objective vaule is shown for the naive model too. [I should compare with the naive model as well, but I don't see how I should do that]

As can be seen from the numbers in Table 3, in general, projections are much faster than the unprojected models. More specifically there are between app. 17 and 33 times fewer iterations and 20-137 times fewer ticks, which corresponds to a difference between 94 and 97 % of the number of iterations, and 96 and 99.5 % cplex ticks, respectively. Meanwhile the difference in objective value is only modest; for the models including hydrostatic constraints, the difference is at most 0.5 %, while the other two models have a difference of 6.8 and 26.5 %, respectively.

For most of the models, the objective is bigger - allowing more than it should, except for the model with a 6 part division, which allows less [There is not that big a difference between the objectives for many versus few parts, which is probably due to the fact that we fill an empty vessel - but let's not get into that]

I should also do some comparisons with the naive model, but which? When comparing to the naive model, we see that this model of course is even faster (between 41-97 times (iterations) and 132-294 (ticks)), but the difference in objective is also between 72 and 76 % for the last 5 models, while it is 32 % for the model without any weights.

Model	Projected			Original			Difference		
	Iter.	Ticks	Objective	Iter.	Ticks	Objective	Iter. (times)	Ticks(times)	Obj.(%)
No weights	11	0.05	$8.63 \cdot 10^6$	363	2.64	$8.08 \cdot 10^6$	33.0	52.8	6.8
No hydro.	9	0.04	$7.87 \cdot 10^6$	188	5.48	$6.22 \cdot 10^6$	20.9	137	26.5
2 parts	14	0.29	$6.09 \cdot 10^6$	251	5.88	$6.07 \cdot 10^6$	17.9	20.3	0.196
4 parts	13	0.18	$6.17 \cdot 10^6$	228	4.95	$6.16 \cdot 10^6$	17.5	27.5	0.153
6 parts	9	0.20	$6.17 \cdot 10^6$	227	5.02	$6.18 \cdot 10^6$	25.2	25.1	0.202
8 parts	12	0.14	$6.21 \cdot 10^6$	233	4.79	$6.18 \cdot 10^6$	19.4	34.2	0.490
Naive model	4	0.02	$1.07 \cdot 10^7$						

Table 3: Iterations, time and objective values for projected and unprojected models, as well as for the naive model.

1.5 Projection of multi commodity flow graphs

It is not only for cargo models/the domain of vessel stowage that it is relevant/useful to eliminate variables irrelevant for analysis of a model at a "higher level" than where it is modelled. As another example, consider e.g. a multi-commodity flow graph $G = (V, E)$, where commodities c_1 to c_k flow through the graph from sources ($S \subseteq V$) to sinks ($T \subseteq V$), and each edge have limits for each commodity as well as a joint limit. For a number of applications it would be of interest to know how much can flow from the source-nodes to the sinks, or more specifically, how the amount/number of each commodity at the sinks (x_{t,c_i} for all $t \in T$ and $1 \leq i \leq k$) depends on the amount/number of the commodities at the sources (x_{s,c_i} for all $s \in S$ and $1 \leq i \leq k$). Notice that in this scenario, the demand of each commodity is not fixed/given.. For this purpose, we are uninterested in the amount/number of each commodity that flows at each edge of the graph, and to get a description of the direct relationship between the inflow-variables and the outflow-variables, the other variables should therefore be eliminated from the system describing the multi-commodity flow problem.

A multi-commodity flow problem is naturally block-structured, though, there are usually many global constraints – corresponding to the common upper limits for each edge. Instead of using these blocks to decompose the system, it is also possible to divide the graph into smaller subgraphs. We then treat the ingoing edges in a subgraph G' as sources ($S_{G'}$) and the outgoing edges as sinks ($T_{G'}$), and eliminate all variables but $\{ x_{v,c_i} \mid v \in S_{G'} \cup T_{G'} \}$ from the inequality systems describing the flow problem in G'. Afterwards, these subgraphs are successively combined into larger graphs $\tilde{G}$ from which all but $\{ x_{v,c_i} \mid v \in S_{\tilde{G}} \cup T_{\tilde{G}} \}$ are eliminated.

Results What/how much to write about how I made the graph?] To show this point, we have generated a flow graph inspired by the problems that can be found in.... The graphs consists of 7 "layers" consisting of 3 nodes each, and there are 2 commodities. From each node at a given level, there is a (directed) edge to each node at the next level. Sources are composed of the nodes at the first level, while the nodes at the last level constitutes the sinks. The capacity for each commodity c_i and edge e is 0 with a probability of 5% and otherwise drawn from a uniform distribution between 5 and 15, while the common capacity of the edge e is 0 with probability 25% and otherwise a number drawn from the uniform distribution between s-10 and s, where s is the sum of the individual cpacities on that edge.

Similarly to Table **??**, Table 4 shows the size of the original model (both as modelled and presolved) and the projections resulting from a flat and decomposed projection, respectively. Both projections finished and the time taken to make them is also shown. Figure 3 shows the progression over time of the number of inequalities and number of variables left to be projected, for both the flat and decomposed projection algorithm.

We see a reduction in the number of inequalities, variables and non-zero entries, of 3.5, 6.6, and 3.3/3.8 times, respectively (in both cases) compared to the presolved model. The density stays almost the same; for the decomposed model, the density increases with 5.3 %, while the density decreases with 8.5% for the flat projection.

Model	Size ineqs (eqs) / vars / non-zeros / density	Time
Original	204 (42) / 120 / 444 / 2.17	-
Presolved	59 / 79 / 201 / 3.41	-
Projected, decomposed	17 (2) / 12 / 61 / 3.59	2h 9m
Projected, flat	17 (2) / 12 / 53 / 3.12	17h 41 m

Table 4: Size of projection of multicommodity flow model

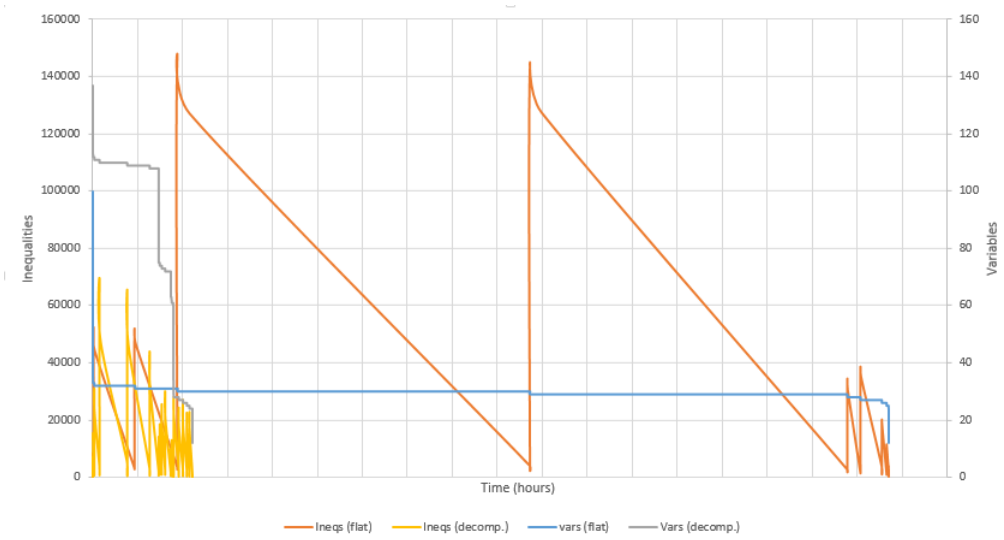

Figure 3: Number of inequalities and variables during flat and decomposed projection of a multi commodity flow problem