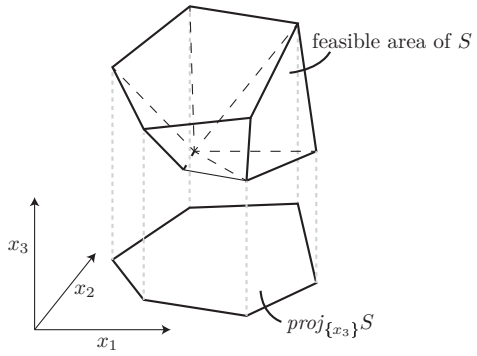

Figure 1: The projection of the S with respect to $\{x_3\}$.

1 Definitions and notation

(In)equality systems In the following, an (in)equality system S is a set of equalities and inequalities over the same set of variables, $VAR(S) = \{x_1, \ldots, x_n\}$. Each (in)equality c is written as either $a_1x_1 + \ldots + a_nx_n = b$ or $a'_1x_1 + \ldots + a'_nx_n \leq b'$, though the left-hand-side is also written using a dot-product. We let $var(c)$ denote the variables whose coefficient in c is nonzero and say that c *uses* x if $x \in var(c)$.

The set of points in $\mathbb{R}^{|VAR(S)|}$ that satisfies all (in)equalities in S is called S's *feasible area*. An (in)equality $c \in S$ is *redundant* if it does not influence the feasible area for S. In other words the inequality $c : \mathbf{a} \cdot \mathbf{x} \leq b$ is redundant iff $\max \mathbf{a} \cdot \mathbf{x}$ w.r.t. $S \setminus \{c\}$ is less or equal to b. An equality is redundant iff both corresponding inequalities are redundant. If the (in)equality c is *not* redundant, it is called *non-redundant*.

Projection As described, the feasible area of S describes the combination of values for the variables in $VAR(S)$ that satisfy all (in)equalities in S. However, there are some variables $Y \subseteq VAR(S)$ whose value in a feasible point we are not interested in; we just want to know that a satisfying value exists. This information is captured by the *projection* of the feasible area of S w.r.t. Y, $proj_Y S \in \mathbb{R}^{|VAR(S) \setminus Y|}$. This is the largest set consisting of values for $VAR(S) \setminus Y$ that can be extended with values for Y such that all (in)equalities in S are satisfied (see Figure 1).

The projection of a system S is a set of points in Euclidian space, and it is the feasible region of (another) inequality system S' (see e.g. [Zie95]). However, many (in)equality systems determine the same feasible area, and when we say e.g. "S' is the projection of S w.r.t. Y" we mean that "S' is one of the (in)equality systems whose feasible area equals the projection of the feasible area of S w.r.t Y".

A note on $VAR(S)$ In this paper, we are mainly interested in the original system S and its projection S', while the associated feasible areas of S and S' are the "mediator" between the two systems. However, since the feasible area of a system is just a set of points in a multi-dimensional Euclidian space, the number and the order of the variables are important and needs to be given (explicitly or implicitly). Though, the inequality $c : a_1x_1 \leq b$ can be considered both as an inequality over the set $\{x_1\}$ as well as over any set X where $x_1 \in X$, and we will not specify $VAR(S')$ formally for every considered (in)equality system S. Intuitively, we just make sure that the dimensions (and order of variables) "match". For example, when two systems S and S' are joined to form another system S'', we consider S and S' as (in)equalities over the same variable set, namely the variables used in either S or S', i.e. $VAR(S'') = var(S) \cup var(S')$. A more stringent exposition keeping track of the variable sets and ordering can be found in [AJ17].

2 Method

Overall procedure

Our *Fourier-Motzkin-based projection framework* for eliminating the variables Y from the (in)equality system S consists of several steps. To begin with, we preprocess the (in)equality system, i.e. we reduce it by removing easily identifiable redundant (in)equalities and assign necessary bounds and values to variables. Then we use the equalities in S to isolate variables from Y and substitute in the rest of the system. This clearly eliminates

variables from the system, and we refer to it as Gauss-elimination. Subsequently, we successively eliminate one variable from Y and remove redundant inequalities from the system afterward, until no more variables remain to be eliminated. Eliminating a variable from S causes some of its (in)equalities to be removed while others new inequalities are added. All of the staying (in)equalities are non-redundant, and hence we only check the added inequalities for redundancy.

At the top-level, the pseudocode for our projection framework is thus as described in the pseudocode below. Each step is detailed further below.

function FM-PROJECTIONFRAMEWORK(S,Y)
  $(S, Y) \leftarrow$ PREPROCESS(S, Y)
  $(S, Y) \leftarrow$ GAUSS-ELIM(S, Y)
  while $Y \neq \emptyset$ **do**
    $(S, Y, New) \leftarrow$ FME-SINGLEVAR(S, Y)
    $S \leftarrow$ REMOVEREDUNDANCY(S, New)
  return S

Projection

Naturally, the goal of our projection framework is projection, and we will start by describing the two types of eliminations used, namely Fourier-Motzkin-elimination and Gauss-elimination.

Fourier-Motzkin-elimination Fourier-Motzkin is a classical algorithm for producing the projection of a set of variables from an inequality system. The method successively eliminates one variable $x \in Y$ until all required variables have been eliminated.

To eliminate a single variable $x \in Y$, the inequalities are first divided into sets, $Pos_S(x)$, $Neg_S(x)$ and $Zero_S(x)$ depending on the sign of x's coefficient. The method traditionally works on systems without equalities, so when doing this, each equality $e : \mathbf{a} \cdot \mathbf{x} = b$ is treated as two inequalities $\mathbf{a} \cdot \mathbf{x} \leq b$ and $-\mathbf{a} \cdot \mathbf{x} \leq -b$. Bounds are treated as any other inequalities, so if ub_x is an upper bound for x, then $x \leq ub_x$ is added to $Pos_S(x)$, and if lb_x is a lower bound for x, then $-x \leq -lb_x$ is added to $Neg_x(S)$.

A new system S' is then created, which consists of $Zero_S(x)$, together with one inequality $i_{p,n,x}$ for each pair $(p, n) \in Pos_S(x) \times Neg_S(x)$. For $p \in Pos_S(x)$ and $n \in Neg_S(x)$, $i_{p,n,x}$ equals p multiplied with the negated coefficient of x in n, added to n multiplied with the coefficient of x in p. That is, if p equals $\mathbf{a} \cdot \mathbf{x} \leq b$ and n equals $\mathbf{a}' \cdot \mathbf{x} \leq b'$, then $i_{p,n,x}$ is the inequality

$$i_{p,n,x} : -a'_x \cdot \mathbf{a} \cdot \mathbf{x} + a_x \cdot \mathbf{a}' \cdot \mathbf{x} \leq -a'_x \cdot b + a_x \cdot b'.$$

By construction, the coefficient for x in $i_{p,n,x}$ is zero, and the resulting system $S' = Zero_S(x) \cup \{\ i_{p,n,x} \mid (p, n) \in Pos_S(x) \times Neg_S(x)\ \}$ is the projection of S w.r.t. $\{x\}$, $proj_{\{x\}} S$ (see [AJ17]).

The order in which variables are eliminated naturally influences the size of the intermediary inequality systems. We have chosen to use the greedy heuristic that minimized the size of the immediately next system [Duf74], which is the most commonly used heuristic. It is easily calculated from the current system as the variable $x \in Y$ that minimized $|Pos_S(x)||Neg_S(x)| - |Pos_S(x)| - |Neg_S(x)|$.

It is apparent that in the worst case scenario where $|Pos_S(x)| = |Neg_S(x)| = \frac{|S|}{2}$ for all $x \in Y$, the number of inequalities in the created system S' is $\frac{1}{4}|S|^2$, which implies that (both time and space) complexity is double-exponential. For a large, dense system, the growth will be substantial, which prohibits it from use for practical purposes *if* the added inequalities are non-redundant or the non-redundant inequalities are not removed (See e.g. [LHM93] and [LS08]). It should, however, also be emphasized that not all inequalities in the succeeding system are necessarily non-redundant; in fact, the number of non-redundant inequalities will at most grow exponentially [**?**].

Gauss-elimination Equalities in an inequality system can be treated as two inequalities when doing FM-elimination as described above. However, an equality e can also be used to isolate a variable $x \in Y$ which can then be substituted in all other (in)equalities in S; in this paper, this is referred to as Gauss-elimination of the variable x using e. This also eliminates x from the system and does not cause the same combinatorial explosion of inequalities as FME may do.

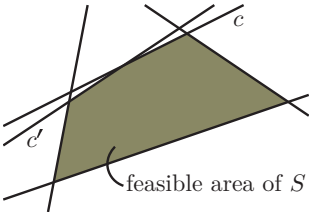

Figure 2: The inequality c is almost redundant compared to c', and vice versa.

Before performing FME we therefore do as many Gauss-elimination of variables in Y as possible. To avoid density, when the system S contains several equalities, we first choose the variable x (used in any equality) that is used the fewest times in total in S. We then choose the equation e among those using x that used the fewest variables, and do Gauss-elimination of x using e. This is then repeated until there are no more equalities using variables from Y.

Gauss-elimination prior to FME is also done e.g. in [SK05].

Redundancy removal

Using optimization software As mentioned, an inequality in S is redundant iff its left-hand-side can never exceed its right-hand-side when the values are restricted by all the other constraints of S. To detect redundancy we can therefore examine each inequality $c : \mathbf{a} \cdot \mathbf{x} \leq b$ in turn and remove it from the system if $\max \mathbf{a} \cdot \mathbf{x}$ subject to $S \setminus \{c\}$ is less than or equal to b; this property is checked using the optimization software `cplex`. Though equalities could be examined this way too, we only examines and remove inequalities.

Not all inequalities have to be examined, though. When removing redundancy from $S' = proj_{\{x\}}S$ we do not need to check inequalities in $Zero_S(x)$; if they were non-redundant before the elimination, they will be non-redundant after.

Parallel redundancy checking For large systems, checking all (in)equalities for redundancy is very time-consuming. We have therefore implemented a method for redundancy removal that uses several threads in parallel.

The method uses a manager to keep track of $k > 0$ individual workers who do the actual redundancy checking. The manager assigns a different inequality to each of the workers, who in parallel each check if their own inequality is redundant w.r.t. their own copy of S. When each worker is done, it reports the result back to the manager, and it gets a new inequality to check. If the reported inequality was redundant, the other workers are notified and remove it from their own copy of the system the next time they check a new inequality.

The manager collects a set containing all the reported redundant inequalities, and when all inequalities are checked, these inequalities can then be removed from the system.

For the method to work correctly, it is important that S contains no two (in)equalities defining the same halfspace, and such easily detectable redundancies are removed prior to the redundancy removal.

Coarsening boundary Several of the constants in the data used for our models are results of various approximations and hence the boundary of the feasible area is not exact. Coarsening the boundary is therefore permissible, and hence we will also remove inequalities that are *"almost redundant"*. An inequality $c : \mathbf{a} \cdot \mathbf{x} \leq b$ is almost redundant if $\max \mathbf{a} \cdot \mathbf{x}$ subject to $S \setminus \{c\}$ is less or equal to $b + \epsilon \cdot |b|$ for a small ϵ, see Figure 2; we have used $\epsilon = 0.01$. If $b = 0$, we instead require the maximum to be a smaller than a given ϵ'.

Thus, instead of only collecting a set of redundant inequalities, the manager also collects a set of almost redundant inequalities; the property is checked by the workers simultaneously with the ordinary redundancy check. After the parallel redundancy check, *one* worker then goes through all the almost redundant inequalities one by one and removes the ones that are still almost redundant. The almost redundant inequalities found this way is then removed from the system S.

Almost redundant inequalities can *not* be removed in parallel. Otherwise, in a situation as in Figure 2, both c and c' could simultaneously be found almost redundant by two different workers, causing them both to be eliminated. When removing almost redundant inequalities sequentially, both inequalities would be checked again, but only the one examined first would be removed.

A method for coarsening the boundary of the feasible area that relies on removing almost redundant inequalities are used in [LS08] and [SL12] too, though the approach is different.

Preprocessing

Prior to projecting the system we perform some simple preprocessing steps in order to have a smaller system as the starting point. The steps (which can be found in e.g. [BMW75], [AA95] and [Mar03]) are then applied repeatedly in a cycle, as long as "something happens" in a cycle, that is, an (in)equality or variable in Y is removed, a variable is substituted with a value, or a bound of a variable or an (in)equality's left-hand-side is updated.

The individual preprocessing steps are as follows; details can be found in [AJ17].

1. Remove all empty inequalities, i.e. inequalities c where $var(c) = \emptyset$, and remove all unused variables $x \in Y$.

For each variable x, we maintain an upper and lower bound, ub_x and lb_x, which initially is ± infinity.

2. If $ub_x = lb_x$ for a variable $x \in X$, then substitute x with lb_x in all (in)equalities in S.
3. If an inequality $a_x \cdot x \leq b$ belongs to S, then remove it from S. If $a_x > 0$ then update the upper bound for x to $\min\{ub_x, \frac{b}{a_x}\}$, otherwise update the lower bound to $\max\{lb_x, \frac{b}{a_x}\}$. For the equality $a \cdot x = b$, both bounds are updated.
4. If $Neg_S(x) = \emptyset$ for an $x \in Y$ (implying that x does not occur in an equality) then remove $Pos_S(x)$ from S. If $Neg_S(x)$ only consist of the inequality defining the lower bound of x then substitute x with lb_x in all (in)equalities in S. Similarly w.r.t. $Pos_S(x)$. We notice that this is *not* a normal preprocessing step, but corresponds to doing FME on x, which changes the feasible area of S.

For each inequality we also maintain an upper and lower bound (which potentially is ± infinity) for its left-hand-side. At a feasible point for S, the upper bound for the (in)equality c with left-hand-side $\mathbf{a} \cdot \mathbf{x}$ is thus $high^c = \sum_{x.a_x>0} a_x \cdot ub_x + \sum_{x.a_x<0} a_x \cdot lb_x$. Similarly, c's lower bound is given by $low^c = \sum_{x.a_x>0} a_x \cdot lb_x + \sum_{x.a_x<0} a_x \cdot ub_x$. These bounds might again imply tighter bounds for the variables.

5. If c is an inequality with right-hand-side b and $high^c \leq b$ then c is redundant and removed. If instead c is an equality (and still $high^c \leq b$) then necessarily $high^c = b$, and c is removed from S while all positive (negative) variables in c is replaced with their upper (lower) bound.
6. If c is an (in)equality with right-hand-side b, and $low^c \geq b$, then neccessarily $low^c = b^c$, and c is removed from S while all positive (negative) variables in c are replaced with their lower (upper) bound.
7. For any variable x used by c, if $a_x > 0$ and $low^c_x < +\infty$, where $low^c_x = \sum_{x'.a_{x'}>0, x' \neq x} a_{x'} \cdot lb_{x'} + \sum_{x'.a_{x'}<0} a_{x'} \cdot ub_{x'}$, then ub_x is updated to $\min\{ub_x, \frac{b - low^c_x}{a_x}\}$. Similarly, if $a_x < 0$ and $low^c_x < +\infty$, then lb_x is updated to $\max\{lb_x, \frac{b - low^c_x}{a_x}\}$. Updating bounds for (in)equalities with only one variable (step 3) is a special case of this.

Comparing two inequalities syntactically, we can in some cases detect redundancy as described below. All pairs of inequalites are then compared in an efficient manner. In the following, let c be an (in)equality in S with left-hand-side $\mathbf{a} \cdot \mathbf{x}$ and right-hand-side b, and let c' be another (in)equality with left-hand-side $\mathbf{a}' \cdot \mathbf{x}$ and right-hand-side b'.

8. If $\mathbf{a} = \sigma \cdot \mathbf{a}'$, then c is *linearly dependent* ([LHM93]). c is hence redundant if it is an inequality, and either $\sigma \geq 0$ and $\sigma \cdot b' \leq b$; or c' is an equality, $\sigma < 0$ and $\sigma \cdot b' \leq b$. c is also redundant if both c and c' are equalities and $\sigma \cdot b' \leq b$. It is enough to check if this holds for $\sigma = \frac{c_x}{c'_x}$ for the first variable x used by c'.
9. If all variables used in c and c' are non-negative, and there exists a $\sigma \geq 0$ such that $a_i \leq \sigma \cdot a'_i$ for all i, while $\sigma \cdot b' \leq b$, then c is *less strict* than c'; it is redundant and hence removed. If c is an equality, so must c' be, and hence c' is turned into an equality. This is only required to be tested for specific value of σ that depends on whether c' has a variable with a negative coefficient or not; see [AJ17].

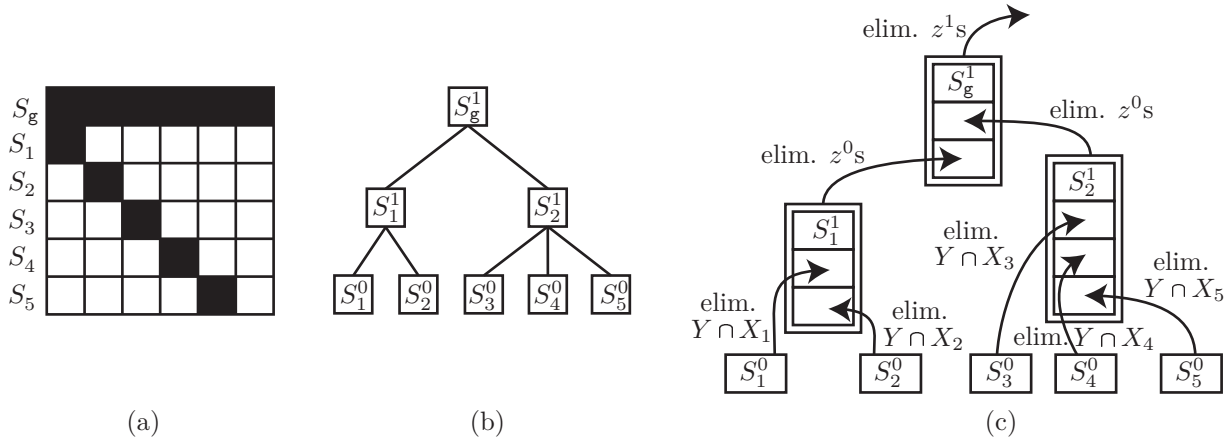

Figure 3: An illustration of (a) the coefficient matrix for a block-structured inequality system S (non-zero sections coloured black). (b) S decomposed into a tree stucture of subsystems. (c) Recursive projection of the tree-structure in (b).

3 Decomposing system

In the following we will describe how the structure of the problem can be exploited to make our projection framework perform better on large systems. Detailed pseudocode and proofs can be found in [AJ17].

The considered problem has a *(primal) block angular structure* [Wil78]. That is, the inequality system S can be divided into k local subsystems, $S_1, \ldots, S_k$, using smaller, disjoint sets of variables, $X_1, \ldots, X_k$, and a global system, S_g, whose constraints "connect" the otherwise independent subsystem by using variables from several local subsystems (see Figure 3(a)). If there were no global constraints, each subsystem could be projected separately and the resulting systems could then be combined to give the projection of the original system. Unfortunately, the global constraints cause the subsystems to get "mixed" when variables are eliminated and hence result in an increasing number of global and dense constraints, which again makes FME perform worse.

However, we can still exploit the structure of the problem. We will define and use auxiliary variables to ensure that we can project the subsystems separately without producing global constraints, before we combine the projected subsystems and eliminate the auxiliary variables.

Using auxiliary variables to separate blocks To remove local variables from the global inequalities, we do as follows for all subsystems S_i:

- For each global inequality c using variables in S_i, we define an auxiliary variable $z^0_{c,i}$ that equals the variables in S_i's contribution to c. We add the equality defining $z^0_{c,i}$ to S_i, and we substitute with $z^0_{c,i}$ in c. For example, if $c : a_1x_1 + a_2x_2 + a_3x_3 + \ldots + a_kx_k \leq b$ is an inequality in S_g and $X_i = \{x_1, x_2\}$, then we add $-z^0_{c,i} + a_1x_1 + a_2x_2 = 0$ to S_i and rephrase c as $z^0_{c,i} + a_2x_3 + \ldots + a_kx_k \leq b$.

  We name the thus expanded subsystem S^0_i.
- Then we project S^0_i w.r.t. all variables from Y that are present in S^0_i. We do keep the auxiliary z^0-variables. Because of these auxiliary variables this only produces inequalities with variables not present in other subsystems S_j.

After projecting each S^0_i we can then combine the results with the rephrased, global inequalities, S^0_g, to create the system $\mathcal{S}$. We can then eliminate all the auxiliary z^0-variables, Z^0, plus the variables in Y not occurring in any S_i from $\mathcal{S}$. This results in a system equivalent to the projection of S w.r.t. Y.

Example As an example, consider the block-structured problem S illustrated in Figure 3(a), where $X_i = \{u_i, w_i\}$ for $i \in \{1, 2, 3, 4, 5\}$, and $S_t = \{e, c\}$ consists of an equality defining the weighted sum, $e : -wSum + 1u_1 + 1w_1 + 2u_2 + 2w_2 + 3u_3 + 3w_3 + 4u_4 + 4w_4 + 5u_5 + 5w_5 = 0$, and another inequality limiting the ws, $c : w_1+w_2+w_3+w_4+w_5 \leq 20$. Assume we want to eliminate all variables but $wSum$, that is $Y = \bigcup_{i \in \{1,2,3,4,5\}} \{u_i, w_i\}$.

We then create the subsystems $S_1^0, \ldots, S_5^0$ and S_g^0, where $S_i^0 = S_i \cup \{-z_{e,i}^0 + iu_i + iw_i = 0\} \cup \{-z_{c,i}^0 + w_i = 0\}$, while the rephrased global inequalities are $S_g^0 = \{-wSum + z_{e,1}^0 + z_{e,2}^0 + z_{e,3}^0 + z_{e,4}^0 + z_{e,5}^0 = 0\} \cup \{z_{c,1}^0 + z_{c,2}^0 + z_{c,3}^0 + z_{c,4}^0 + z_{c,5}^0 \leq 20\}$.

To procure the projection of S w.r.t. Y we can instead project S_i^0 w.r.t. $Y \cap X_i$ for all $1 \leq i \leq 5$, join the projections together with S_g^0 in the system $\mathcal{S}$, and project $\mathcal{S}$ w.r.t. $\{ z_{c,i}^0, z_{e,i}^0 \mid 1 \leq i \leq 5 \}$.

Equivalent projections Comparing the union of the new (unprojected) subsystems, $\mathfrak{S} := S_1^0 \cup \ldots \cup S_k^0 \cup S_g^0$, with the original system S it is clear that all we have done is defining auxiliary variables and substituted them in the system. Intuitively, it is therefore clear, that eliminating the variables in Y from S is equivalent to eliminating Y and the defined z^0-variables from $\mathfrak{S}$.

This, in turn, is equivalent to projecting the subsystems S_i^0 separately w.r.t. $X_i \cap Y$, combining the resulting systems with S_g^0, and then eliminate the z^0-variables and the remaining Y-variables: When eliminating $Y \cup Z^0$ from $\mathfrak{S}$, we can choose to first eliminate $X_1 \cap Y$, then $X_2 \cap Y$ up to $X_k \cap Y$, and finally $Z^0 \cup Y \setminus (X_1 \cup \ldots \cup X_k)$. Any variable in $X_1 \cap Y$ has a zero-coefficient in all (in)equalities not in S_1^0, and the inequalities in $\mathfrak{S} \setminus S_1^0$ will therefore not be changed by the FME procedure when $X_1 \cap Y$ is eliminated. We can therefore set aside these inequalities until FME is done eliminating $X_1 \cap Y$. Likewise, when eliminating variables in $X_i \cap Y$, no (in)equality from either $S_{i+1}^0 \cup \ldots \cup S_k^0 \cup S_g^0$ or the already projected systems contains any variables from $X_i \cap Y$ and can hence be "put aside" and only included later when $Z^0 \cup Y \setminus (X_1 \cup \ldots \cup X_k)$ is eliminated.

Thus, the following holds.

Proposition 1. *The projection of S w.r.t. Y defines the same feasible area as the projection of $S_1^0 \cup \ldots \cup S_k^0 \cup S_g^0$ w.r.t. $Y \cup Z^0$.*

Further decomposition $\mathcal{S}$ is by construction block structured, and instead of eliminating $Z^0 \cup Y \setminus (X_1 \cup \ldots \cup X_k)$ immediately, if necessary we can use further auxiliary variables to postpone "mixing" blocks when eliminating the remaining variables. To do so, we collect all subsystems into k_1 small groups; we have mostly used groups of size 2, 3 or 1[1]. For each group i we do as follows:

- We join the systems in the group into a new system, S_i^1. For each (rephrased) global inequality c using variables occurring in S_i^1, we then define a variable, $z_{c,i}^1$, that equals S_i^1's contribution to c. We then add the defining equality to S_i^1 and rephrase c using $z_{c,i}^1$.
- Then we project the resultant system, S_i^1, w.r.t. the previous, auxiliary z^0-variables, while we do keep the newly created z^1-variables.

Subsequently we can then join the projected S^1-systems with the (rephrased) transverse inequalities, S_g^1, and finally project the last auxiliary variables, or we can repeat the step above step until the final projection can be done.

Tree structure Each time we further decompose a system, we use a partitioning of the subsystems $S_1^l, \ldots, S_{k_l}^l$ to create a new "level" of k_{l+1} subsystems, $S_1^{l+1}, \ldots, S_{k_{l+1}}^{l+1}$. This effectively creates a tree structure of smaller inequality systems, where each node is associated with a subsystem and a set of variables that should be eliminated from the system. Using this tree, an inequality system associated with a node is projected by recursively projecting the systems associated with the children of the node, and as expected, projecting the root of the tree constructed from S and Y as explained creates a system equivalent to the projection of S w.r.t. Y.

Proposition 2. *The projection of the system associated with the root of the tree constructed from S and Y w.r.t. the Y- and Z-variables as described corresponds to projecting S w.r.t. Y*

The intuition is as before; projecting all Y and Z-variables from the union of all constructed (unprojected) subsystems corresponds to projecting the Y variables from S, and because we can choose the elimination order of the variables, we only need to project the subsystems in the tree in the correct order; a rigorous proof can be found in [AJ17].

Using our projection framework we obtain the projection of S w.r.t. Y by calling PROJECTNODE(root of T), where T is the tree structure constructed from S and Y and PROJECTNODE is as below.

[1] "Combining" a single sybsystem corresponds to substituting a variable with a new variable, but can be done for convenience of notation

function PROJECTNODERECURSIVELY(Node n)
  $(S, Y) \leftarrow$ the system and variable set associated with n
  if n is a leaf **then**
    return FM-PROJECTIONFRAMEWORK(S, Y)
  else
    for all children m of n **do**
      $S \leftarrow S \cup$ PROJECTNODE(m)
    return FM-PROJECTIONFRAMEWORK(S, Y)

We note that due to the boundary coarsening this only approximates $proj_Y(S)$.

Example Consider the system from the previous example, which was decomposed into the subsystems $S_1^0, \ldots, S_5^0$ and S_g^0. Instead of projecting these subsystems as described in the example we insert an additional level in the decomposition, and we choose to group the five subsystems into two groups, namely $\{S_1^0, S_2^0\}$ and $\{S_3^0, S_4^0, S_5^0\}$. That is, we construct systems $S_1^1 = \{-z_{e,1}^1 + z_{e,1}^0 + z_{e,2}^0 = 0\} \cup \{-z_{c,1}^1 + z_{c,1}^0 + z_{c,2}^0 = 0\}$ and $S_2^1 = \{-z_{e,2}^1 + z_{e,3}^0 + z_{e,4}^0 + z_{e,5}^0 = 0\} \cup \{-z_{c,2}^1 + z_{c,3}^0 + z_{c,4}^0 + z_{c,5}^0 = 0\}$, and make the tree structure as in Figure 3(b). The rephrased global inequalities are now $S_t^1 = \{-wSum + z_{e,1}^1 + z_{e,2}^1 = 0\} \cup \{z_{c,1}^1 + z_{e,2}^1 \leq 20\}$. The projection of S w.r.t. Y is now made by projecting the root of the tree recursively as shown in Figure 3(c).

Other block structures Using this decomposition, it is of course also possible to project nested block structured problems, that is a system that on the top-level can be divided into a transverse part and a number of local parts that in themselves can be divided into further local parts and a transverse part, and so on. Other block structured problems such as staircase problems can also be decomposed into a tree structure and projected using this structure.

Parallelization When the system S is decomposed into subsystems in a tree structure as described above, the projection itself can be parallelized by maintaining a queue of not yet projected subsystems whose children have all been projected; this queue thus initially contains all leafs. The members of the queue are then solved independently by multiple solvers in parallel, who also add systems to the queue when the membership condition is met. The pseudocode for this is presented below. Besides a system S and a variable set Y, each node n is associated with a number, n_{count}, corresponding to the number of projected children (initially set to 0), and a system, n_{proj}, corresponding to the projection $proj_Y(S)$ when it is done (initially set to $\varnothing$).

function PARALLELTREEPROJECTION(S, Y)
  Construct tree structure T from S and Y
  $n_{count} \leftarrow 0$ and $n_{proj} \leftarrow \varnothing$ for all nodes n in T
  Create a set W of workers, initially idle
  Initialize a queue Q with all leaves in T
  while $root(T)_{count} = \varnothing$ **do**
    if Q is non-empty and a worker $w \in W$ is idle **then**
      Remove first node n from Q
      Call PROJECTSINGLENODE(n) on w
  return the projection of the root of T

function PROJECTSINGLENODE(Node n)
  $(S', Y') \leftarrow$ the system and the variable set associated with n
  $S' \leftarrow S' \cup_{m \in children(n)} m_{proj}$
  $n_{proj} \leftarrow$ FM-PROJECTIONFRAMEWORK(S',Y')
  Increase $parent(n)_{count}$ by one
  if $parent(n)_{count} = |children(parent(n))|$ **then**
    Add $parent(n)$ to Q
  return

All workers project a system using the FM-based projection framework, which again involves parallel redundancy removal, and thus it is important to only use as many resources (threads) in total as there are available. Then, when at some point there are less nodes left to project than there are workers in W, the superfluous workers can pass on their resources to the remaining workers.

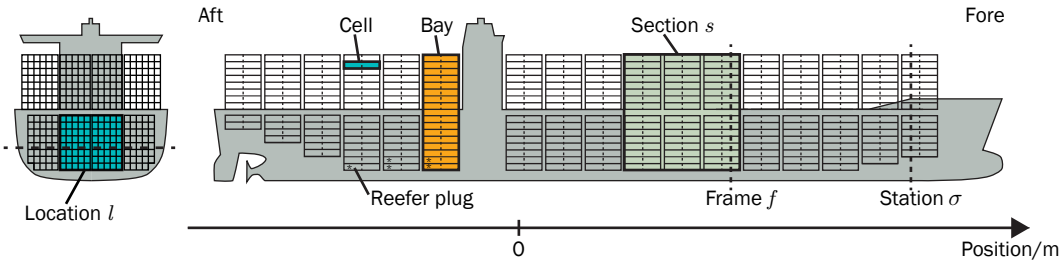

Figure 4: A vessel with structures [More reefer plugs] [-skal laves om]

4 Standard Vessel Model

In this section we will describe the data given for a stowage model, and we will explain how this information is translated into our proposed *Standard Vessel Model* (SVM), whose main purpose is to abstracts away much of the unnecessary complexity relating to the physical layout of the vessel.

The data comes from...

Vessel data

Vessel structure The cargo space of a container vessel is divided into parts called *bays* that each consists of a grid of *cells*. Each cell is divided into two *slots* and can accordingly hold one standard 40' container or two 20' containers. Some cells have power plugs allowing for *reefer* containers to be refrigerated as required.

The cargo itself is described by a container *type*, which is defined by a length (20' or 40'[2]), whether it is a reefer-container or not, and a weight class (in a discrete set of weights). Besides containers, vessels also carry *ballast tanks* in fixed positions along the vessel that can be filled with water to improve the stability of the vessel.

Vessel data

Capacities Though containers physically are placed in specific slots, the considered vessel data specifies certain capacities in subsections of each bay, called *locations*. These capacities are given in *TEU* (Twenty-foot Equivalent Units), i.e. a standard $20'$ container takes up one TEU, while a $40'$ container takes up two TEU. For each location, the vessel data includes upper bounds for: the total number of TEUs, $20'$ containers, $40'$ containers, reefer slots, reefer cells, plus separate total weight limit for $20'$ and $40'$ containers, respectively. The latter exist, since only $20'$ containers rest on the middle support posts of the stack it is in, while the end posts hold weight of both $20'$ and $40'$ containers. This may lead to different weight capacities of $20'$ and $40'$ containers.

Limits for each ballast tank are likewise given in the input data, as well as a limit for the total displacement, i.e. the weight of the vessel, including ballast water, cargo and the ship itself. The weight of the (empty) vessel (called *lightship*) is given in the data by a set of "blocks" placed along the vessel. Each block has a given weight, which is assumed equally distributed along the block.

Stress forces On a vessel, stress forces arise as a result of gravitation acting downwards and buoyancy acting upwards. This results in shear forces and bending moments along the longitudinal axis of the vessel, and limits on these are given for a set of reference points along the vessel called *frames*. The buoyancy force comes from the vessel's displacement of water and hence depends on the varying (and irregular) shape of the hull and the displacement of the vessel. The area submerged in water is given at another set of reference points called *stations* for a discrete set of displacement values.

Further requirements to ensure the stability of the vessel are imposed in real life and considered eg. in [Del13], but is not considered here.

Sections The goal of our Standard Vessel Model is to describe the various capacity- and hydrostatic constraints reasonably accurate, while abstracting away from unnecessary details relating to the irregularity of the vessel's shape and the non-coinciding reference points of tanks, bays, stations and frames. Instead, we will use bigger

[2] 45' long containers also exists but are more rare and is not considered in this paper.

sections of the vessel and the endpoints of these as the common reference for capacities and hydrostatic constraints. Each of these sections are defined to either span a number of succeeding bays, or a part of the ship containing no bays at all.

Input Data The vessel structure is described using the sets L (locations), BT (ballast tanks), B (blocks with a constant weight W_b for all $b \in B$), ST (stations) and F (frames). All elements in these sets have a fixed longitudinal position fore and aft ($P_e^{\mathtt{f}}$ and $P_e^{\mathtt{a}}$ for all $e \in L \cup BT \cup B \cup ST \cup F$), and for the latter two sets, the fore and aft position coincide. The division of the vessel into sections are described by the sets S (sections), which is divided into sections fore and aft ($S^{\mathtt{f}}$ and $S^{\mathtt{a}}$), as well as a set L_s for all $s \in S$ (locations in section s). A fore and aft position of each s ($P_s^{\mathtt{f}}$ and $P_s^{\mathtt{a}}$) is implicitly given by the locations in L_s.

The cargo is described by the set T (container types) with subsets T^{20} (20′ container types – $T \setminus T^{20}$ hence indicates the set of 40′ container types) and $T^{\mathtt{R}}$ (reefer container types). Further, each container type has a weight, W^τ, associated.

The capacity-limits are given by the constant $C_l^{\mathtt{TEU}}$, C_l^{20}, C_l^{40}, $C_l^{\mathtt{RS}}$, $C_l^{\mathtt{RC}}$, $C_l^{\mathtt{W20}}$, $C_l^{\mathtt{W40}}$ for each $l \in L$, and by $Max_b^{\mathtt{W}}$ for all $b \in BT$, plus $Max^{\mathtt{W}}$. The limits on the stress forces are given by $Max_f^{\mathtt{sf}}$, $Min_f^{\mathtt{sf}}$, $Max_f^{\mathtt{bm}}$, $Min_f^{\mathtt{bm}}$ for all $f \in F$. At each station $\sigma \in ST$, the submerged area of the ship at σ is given for a discrete set of displacements in the table A_σ.

Standard Vessel Model

The input data outlined above is used to construct the SVM described in the following.

Variables As decision variables we use $x_{s,\tau}$ for all $s \in S$ and $\tau \in T$, which denotes the number of containers of type τ stowed in sections s, and t_s for all $s \in S$, which denotes the amount of ballast water in ballast tanks in section s.

As important auxiliary variables, we define $x_\tau = \sum_{s \in S} x_{s,\tau}$ for all $\tau \in T$, the number of containers of type τ on the vessel in total, disregarding their placements. It is the relationship among these variables that we are actually interested in. Another important auxiliary variable is w_s, denoting the total weight of section s, everything included. Likewise the total weight, $w = \sum_{s \in S} w_s$ is used in hydrostatics constraints (see further below). Other auxiliary variables used to ease notation are mentioned below as necessary.

Capacity constraints For each section s we define the capacity of a certain type, e.g. C_s^{20}, by summing the corresponding capacities for all locations $l \in L_s$. We then require that the number/weight of containers of the given type in s is within this limit, e.g. $\sum_{\tau \in T^{20}} x_{s,\tau} \leq C_s^{20}$. This principle is applied to all the aforementioned location-based capacities.

The amount of (ballast) water in any section s should not exceed the combined amount of water in the portion of each tank that lies within that section; multiplying the latter amount of water with the density for ballast water, we get an upper bound for t_s, which is then imposed as a constraint. We also define constraints to ensure that all variables $x_{s,\tau}$ and t_s are positive and that $w \leq Max^{\mathtt{W}}$.

Hydrostatic constraints From the table A_σ we make an approximation of the area at σ for any positive d by linearizing it between the maximal d value given in the table and a value d_{min} at the point where the hull does not "curve" too much anymore; for the cross-section given in Figure 4 this would correspond to the displacement giving the marked water line. We note that since our SVM will mainly be used for some sort of maximization of loaded cargo it is a fair assumption to make that the displacement will be above the found d_{min}. The submerged area at a point p between two consecutive stations σ and σ' for a displacement d can then also be linearized as a function of the longitudinal position of p. From this we then calculate an approximation of the buoyancy of section s for a given d by averaging the areas between the two endpoints of s and multiplying with the distance between the points; if there are stations lying within s, several volumes are calculated and added. On the other hand, the downward-acting force at section s is given by the weight w_s. For s, the resulting force is thus the weight of section s minus the (positive) buoyancy stemming from s.

Given a displacement d, the shear forces and bending moment at each section s's aft endpoint is then calculated. For sections in $S^{\mathtt{f}}$ ($S^{\mathtt{a}}$) the shear force equals the sum of resulting forces fore (aft) of the aft endpoint $P_s^{\mathtt{a}}$, while the bending moment equals the sum of resulting forces of sections s' lying fore (aft) $P_s^{\mathtt{a}}$ times the distance from $P_s^{\mathtt{a}}$ to s''s (longitudinal) midpoint.

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.

To obtain upper (lower) bounds for the shear force at the sections' aft endpoints, we linearize the upper (lower) bound for the shear force between two consecutive frames as a function of their position. To obtain the bounds for the shear force at the aft endpoint of s we then use this linearization given by the point's two closest surrounding frames. Similarly for the upper and lower bounds for the bending moment.

Hydrostatic constraints are then added to our model to ensure that the shear force and bending moment at each section's aft endpoint is within the calculated upper and lower bounds.

5 Results

5.1 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"[3]. The revenue is dependent on the container type, so that "limiting" container types yields higher revenue.

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

5.2 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

[3] 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

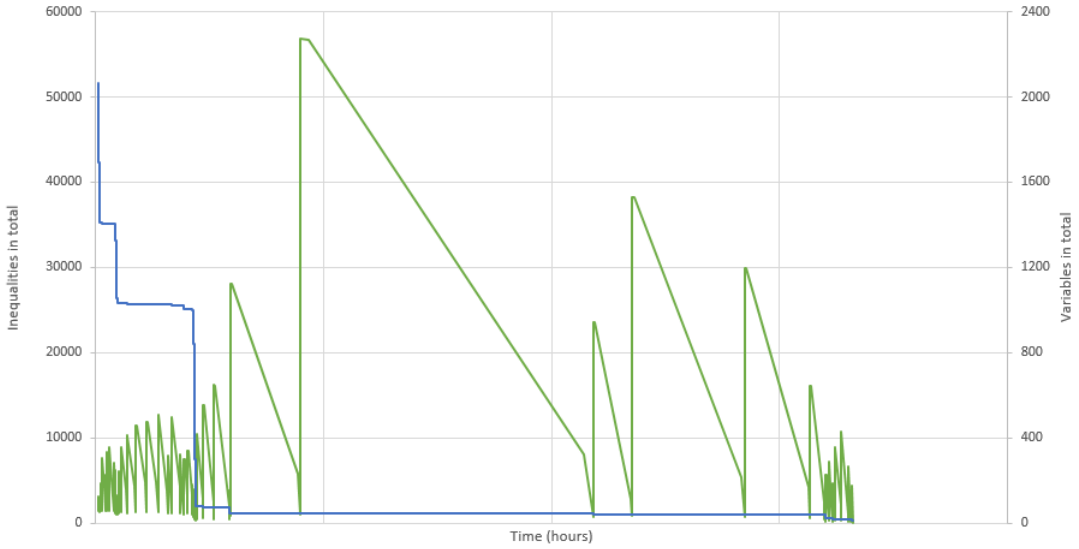

Figure 5: The inequalities growth and variable decrease for the 8 part-model with decomposition.

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.

5.3 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 5 shows the progression of the number of inequalities and variables, respectively, as the algorithm runs on the decomposed 8 part-model. Likewise, Figure 6 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 5 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

Figure 6: The inequalities growth for the 8 part-model with and without decomposition.

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 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.]

5.4 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

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.

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.

5.5 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 7 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.

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

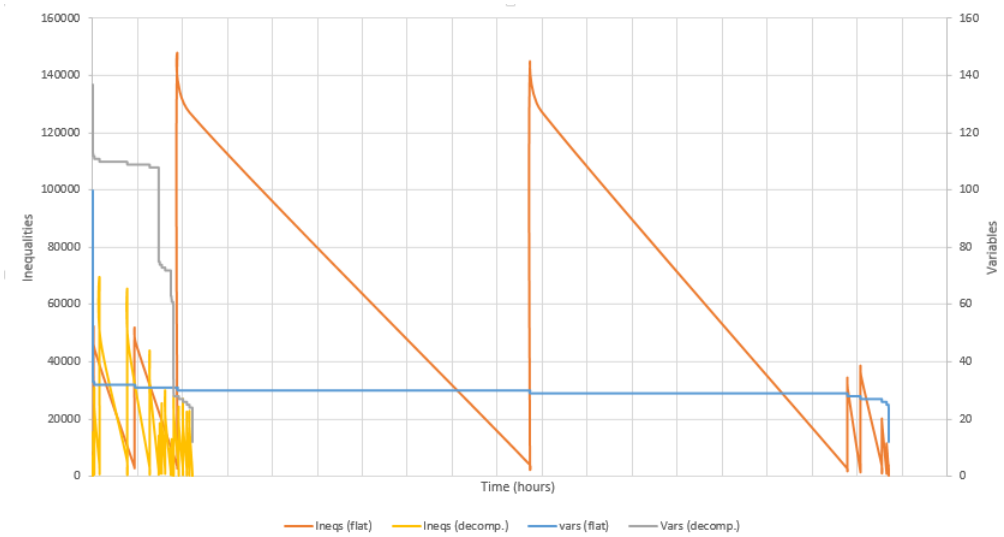

Figure 7: Number of inequalities and variables during flat and decomposed projection of a multi commodity flow problem

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.

References

[AA95] Erling D. Andersen and Knud D. Andersen. Presolving in linear programming. *Mathematical Programming*, 71(2):221–245, 1995.

[AJ17] Mai L. Ajspur and Rune M. Jensen. Using fourier-motzkin-elimination to derive capacity models of container vessels. Technical Report TR-2017-197, IT University of Copenhagen, 1 2017.

[BMW75] A. L. Brearley, G. Mitra, and H. P. Williams. Analysis of mathematical programming problems prior to applying the simplex algorithm. *Mathematical programming*, 8(1):54–83, 1975.

[Del13] Alberto Delgado. *Models and Algorithms for Container Vessel Stowage Optimization*. PhD thesis, IT University of Copenhagen, 2013.

[Duf74] Richard J. Duffin. *On Fourierï¿½s analysis of linear inequality systems*, pages 71–95. Springer, 1974.

[LHM93] Jean-Louis Lassez, Tien Huynh, and Ken McAloon. Simplification and elimination of redundant linear arithmetic constraints. In Frédéric Benhamou and Alain Colmerauer, editors, *Constraint Logic Programming*, pages 73–87. MIT Press, Cambridge, MA, USA, 1993.

[LS08] Alexander M. Lukatskii and Demetrius V. Shapot. A constructive algorithm for folding large-scale systems of linear inequalities. *Computational Mathematics and Mathematical Physics*, 48(7):1100–1112, 2008.

[Mar03] Istvan Maros. *Computational Techniques of the Simplex Method*, volume 61 of *International Series in Operations Research & Management Science*. Springer US, 2003.

[SK05] Axel Simon and Andy King. Exploiting sparsity in polyhedral analysis. In Chris Hankin and Igor Siveroni, editors, *Proceedings of the 12th International Symposium in Static Analysis (SAS)*, pages 336–351. Springer Berlin Heidelberg, 2005.

[SL12] Demetrius V. Shapot and Alexander M. Lukatskii. Solution building for arbitrary system of linear inequalities in an explicit form. *American Journal of Computational Mathematics*, 2(01):1, 2012.

[Wil78] H. P. Williams. *Model Building in Mathematical Programming*. John Wiley & Sons, LTD, 1978.

[Zie95] Günter M. Ziegler. *Lectures on polytopes*, volume 152 of *Graduate texts in mathematics*. Springer, New York, 1995.

Appendix

Vessel data

The vessel's data is given by the following sets and constant:

Sets used in vessel data

L	Locations.	T	Types of containers.
S	Sections.	$T^{20} \subseteq T$	Types that are $20'$ long.
$S^{\mathtt{f}}, S^{\mathtt{a}} \subseteq S$	Sections fore/aft.	$T^{\mathtt{R}} \subseteq T$	Types that are reefers.
$L_s \subseteq L$	Locations in section $s \in S$	BT	Ballast tanks.
F	Frames.	B	Blocks with constant weight.
ST	Stations.		

ST includes $\sigma^{\mathtt{a}}$ and $\sigma^{\mathtt{f}}$, which are two (artificial) stations at the aft most and fore most positions of the ship, respectively. Similarly, F contains the frames $f^{\mathtt{a}}$ and $f^{\mathtt{f}}$. The areas (bounds) for these stations (frames), are trivial.

Constants used in vessel data

C_l^{20}, C_l^{40}, $C_l^{\mathtt{TEU}}$, $C_l^{\mathtt{RS}}$, $C_l^{\mathtt{RC}}$, $C_l^{\mathtt{W20}}$, $C_l^{\mathtt{W40}} \in \mathbb{N}$	The capacity for each location $l \in L$, w.r.t. $20'$ containers, $40'$ container, TEUs, reefer slots, reefer cells, weight of $20'$ containers, and weight of $40'$ containers, respectively.
$W_\tau, W_b \in \mathbb{R}$	The weight of a type of container $\tau \in T$, and of a block $b \in B$, respectively.
$A_{\sigma,d} \in \mathbb{R}_0^+$	The area submerged in water at station $\sigma \in ST$ for some displacement d in a finite set D.
$P_l^{\mathtt{f}}, P_l^{\mathtt{a}}, P_b^{\mathtt{f}}, P_b^{\mathtt{a}}, P_t^{\mathtt{f}}, P_t^{\mathtt{a}} \in \mathbb{R}$	The longitudinal position of the fore and aft endpoint of location $l \in L$, block $b \in B$ and tank $t \in BT$, respectively.
$P_\sigma, P_f \in \mathbb{R}$	The lengthwise position of station $\sigma \in ST$, and of frame $f \in F$, respectively.
$Max^{\mathtt{wTotal}}$, $Max_b^{\mathtt{t}} \in \mathbb{R}^+$	Upper bound for the total displacement and for each ballast tank $b \in BT$, respectively.
$Max_f^{\mathtt{sf}}$, $Min_f^{\mathtt{sf}}$, $Max_f^{\mathtt{bm}}$, $Min_f^{\mathtt{bm}} \in \mathbb{R}$	The upper and lower bounds for the shear force and bending moment at each frame $f \in F$, respectively.

In order to simplify the model and make it more succinct we want to have weights and capacities given for each section, and bounds given at their endpoints.

Output data and standard model

Our resulting standard model uses the sets S, T, $T^{20}, T^{\mathtt{R}}$ and BT (given above), while the used variables are summarized in Table 5. Our standard model uses the constants W_τ and $Max^{\mathtt{wTotal}}$ (given above) as well as the constant given in Table 5. These are calculated from the input data as explained further below.

Regarding the variables, we note that even though $x_{s,\tau}$ is a number of containers and hence a natural number, we model it as a reel number to ensure that the resulting model is an LP.

Decision variables	
$x_{s,\tau} \in \mathbb{R}_0^+$	Number of containers of each type $\tau \in T$ to be stowed in each section $s \in S$.
$xt_s \in \mathbb{R}_0^+$	The weight of ballast tanks within section $s \in S$.
Auxilliary variables	
$x_\tau \in \mathbb{R}_0^+$	The total number of containers of each type $\tau \in T$.
$w_s \in \mathbb{R}_0^+$	The weight of section $s \in S$, everything included.
$wTotal \in \mathbb{R}_0^+$	The total displacement.
$b_{s,d} \in \mathbb{R}$	The buoyancy force for section $s \in S$ with a total displacement of $d \in \mathbb{R}$.
$sf_s \in \mathbb{R}$	The shear force at the aft endpoint of each section $s \in S$.
$bm_s \in \mathbb{R}$	The bending moment at the aft endpoint of each section $s \in S$.
Constants	
C_s^{20}, C_s^{40}, $C_s^{\mathtt{TEU}}$, $C_s^{\mathtt{RS}}$, $C_s^{\mathtt{RC}}$, $C_s^{\mathtt{W20}}$, $C_s^{\mathtt{W40}} \in \mathbb{N}$	The capacity for each section $s \in S$, w.r.t. $20'$ containers, $40'$ container, TEU, reefer slots, reefer cells, weight of $20'$ containers, and weight of $40'$ containers, respectively.
$P_s^{\mathtt{f}}, P_s^{\mathtt{a}} \in \mathbb{R}$	The fore and aft position of each section $s \in S$.
$Max_s^{\mathtt{sf}}$, $Min_s^{\mathtt{sf}}$, $Max_s^{\mathtt{bm}}$, $Min_s^{\mathtt{bm}} \in \mathbb{R}$	The upper and lower bounds for the shear force and bending moment at the aft endpoint of each section $s \in S$.
$Max_s^{\mathtt{t}} \in \mathbb{R}$	The upper bound for the weight of ballast tanks in section $s \in S$.

Table 5: Variables and constants used (to describe) our Standard Vessel Model.

Constraints

Location-based capacity constraints Firstly, we have constraints that ensure that for each location of the vessel, the stowed containers are within the allowed capacities w.r.t. the number of $20'$ containers, $40'$ containers, TEUs, and the weight of the $20'$ and $40'$ containers, respectively. These constraint are modeled in the inequalities (1)-(5) below. Likewise, we have a constraint, (6), that ensures that the weight of a location is within limits, taken the different distribution of the weight of $40'$ containers, respectively $20'$ containers, within a slot into consideration; the weight of a $40'$ container is distributed on the four outer corner places of a slot while the weight of $20'$ containers also rest on the inner corners of the slot. Lastly, (7) and (8) ensure that each reefer container can be refrigerated, and that the total number of cells taken up by the reefer containers are within capacity, respectively.

$$\forall s \in S: \quad \sum_{\tau \in T^{20}} x_{s,\tau} \leq C_s^{20} := \sum_{l \in L_s} C_l^{20} \tag{1}$$

$$\forall s \in S: \quad \sum_{\tau \in T \setminus T^{20}} 2 \cdot x_{s,\tau} \leq C_s^{40} := \sum_{l \in L_s} C_l^{40} \tag{2}$$

$$\forall s \in S: \quad \sum_{\tau \in T^{20}} x_{s,\tau} + 2 \cdot \sum_{\tau \in T \setminus T^{20}} x_{s,\tau} \leq C_s^{\mathtt{TEU}} := \sum_{l \in L_s} C_l^{\mathtt{TEU}} \tag{3}$$

$$\forall s \in S: \quad \sum_{\tau \in T^{20}} W_\tau \cdot x_{s,\tau} \leq C_s^{\mathtt{W20}} := \sum_{l \in L_s} C_l^{\mathtt{W20}} \tag{4}$$

$$\forall s \in S: \quad \sum_{\tau \in T \setminus T^{20}} W_\tau \cdot x_{s,\tau} \leq C_s^{\mathtt{W40}} := \sum_{l \in L_s} C_l^{\mathtt{W40}} \tag{5}$$

$$\forall s \in S: \quad 0.5 \cdot \sum_{\tau \in T^{20}} W_\tau \cdot x_{s,\tau} + \sum_{\tau T \setminus \in T^{20}} W_\tau \cdot x_{s,\tau} \leq C_s^{\mathtt{W40}} \tag{6}$$

$$\forall s \in S: \quad \sum_{\tau \in T^{\mathtt{R}}} x_{s,\tau} \leq C_s^{\mathtt{RS}} := \sum_{l \in L_s} C_l^{\mathtt{RS}} \tag{7}$$

$$\forall s \in S: \quad \sum_{\tau \in T^{20\mathtt{R}}} 0.5 \cdot x_{s,\tau} + \sum_{\tau \in T^{40\mathtt{R}}} x_{s,\tau} \leq C_s^{\mathtt{RC}} := \sum_{l \in L_s} C_l^{\mathtt{RC}} \tag{8}$$

The constraint in (9) defines the variables, x_τ, that specifies how many containers of a specific type τ is stowed on the vessel. These are the variables that we want the projected (in)equality system to be expressed in.

$$\forall \tau \in T: x_\tau = \sum_{s \in S} x_{s,\tau} \tag{9}$$

(10) below defines the total weight of section $s \in S$, including cargo, ballast tanks and the vessel itself. For this we need the fore and aft endpoints of s, which is given by the maximal fore-positions, respecitively the minimal aft-position, of the locations in s, i.e. $P_s^{\mathtt{f}} = \max\{\ P_l^{\mathtt{f}} \mid l \in L_s\ \}$ and $P_s^{\mathtt{a}} = \min\{\ P_l^{\mathtt{a}} \mid l \in L_s\ \}$.

$$\forall s \in S: \quad w_s = \sum_{\tau \in T} W_\tau \cdot x_{s,\tau} + xt_s + \sum_{b \in B} percent([P_b^{\mathtt{a}}; P_b^{\mathtt{f}}], [P_s^{\mathtt{a}}; P_s^{\mathtt{f}}]) \cdot W_b, \tag{10}$$

where $percent(J, I)$ defines how big a percentage of the interval $J = [J_1; J_2]$ lies within the interval $I = [I_1; I_2]$. That is

$$percent([J_1; J_2], [I_1; I_2]) = \frac{\max(\min(J_2, I_2) - \max(J_1, I_1), 0)}{I_2 - I_1}.$$

We also ensure that the number of containers of each type as well as the amount of ballast in the ballast tanks (placed) in each section is positive. For the ballast tanks, we also require that each tank's content is within limits, and likewise for the the total weight (displacement). These constraints are given in (11), (12), and (13) below.

$$\forall s \in S, \tau \in T: \quad 0 \le x_{s,\tau}. \tag{11}$$

$$\forall s \in S: \quad 0 \le xt_s \le Max(xt_s) \coloneqq \sum_{b \in BT} percent([P_b^{\mathtt{a}}; P_b^{\mathtt{f}}], [P_s^{\mathtt{a}}; P_s^{\mathtt{f}}]) \cdot Max_b^{\mathtt{t}} \tag{12}$$

$$\sum_{s \in S} w_s \le Max^{\mathtt{Wtotal}} \tag{13}$$

Hydrostatic constraints At each station σ the submerged area of the cross-section for a given displacement d in a discrete set of values, D, is given by $A_{\sigma,d}$. We make the assumption that above a given treshold $d_{min} \in D$ (and until the maximal displacement d_{max} given in D), the submerged area is close to a linear function of the displacement. The submerged area $A'_{\sigma,d}$ for any $d \in [d_{min}; d_{max}]$ can therefore be approximated as

$$A'_{\sigma,d} = \frac{A_{\sigma,d_{max}} - A_{\sigma,d_{min}}}{d_{max} - d_{min}} \cdot (wTotal - d_{max}) + A_{\sigma,d_{max}},$$

and we extend this to hold for every $d \in \mathbb{R}$.

The submerged area between two consecutive stations σ and σ' are linearized, so that the submerged area for displacement d at any point $p \in [P_\sigma; P_{\sigma'}]$, where $\sigma \neq \sigma^{\mathtt{f}}$ and $\sigma' = argmin_{\{\ s \mid P_s > P_\sigma\ \}} P_s$, is approximated by

$$A''_{p,d} = \frac{A'_{\sigma',d} - A'_{\sigma,d}}{P_{\sigma'} - P_\sigma} \cdot (p - P_\sigma) + A'_{\sigma,d}.$$

From this we can calculate the buoyancy of section $s \in S$ at a displacement $d \in \mathbb{R}$. First we let $P \coloneqq \{P_s^{\mathtt{a}}, P_s^{\mathtt{f}}\} \cup \{\ P_\sigma \mid \sigma \in ST,\ P_s^{\mathtt{a}} < P_\sigma < P_s^{\mathtt{f}}\ \}$ be the points of the stations between (and including) the section's two endpoints. Then the buoyancy of the section s is given by

$$b_{s,d} = \sum_{p \in P \setminus \{P_s^{\mathtt{f}}\}} \frac{1}{2} \cdot (A''_{p,d} + A''_{next_p,d}) \cdot (next_p - P), \text{where } next_p = \min\{\ p' \in P \mid p' > p\ \}.$$

In a similar manner, we linearize the upper bound for the shear force between two consecutive frames, so that the upper bounds for the shear force at the aft endpoints of section s is

$$Max_s^{\mathtt{sf}} = \frac{Max_{f_2}^{\mathtt{sf}} - Max_{f_1}^{\mathtt{sf}}}{P_{f_2} - P_{f_1}} \cdot (P_s^{\mathtt{a}} - P_{f_1}) + Max_{f_1}^{\mathtt{sf}},$$

where $f_1 = argmax_{\{\ f \in F \mid P_f \le P_s^{\mathtt{a}}\ \}} P_f$ and $f_2 = argmax_{\{\ f \in F \mid P_f > P_s^{\mathtt{a}}\ \}} P_f$. Of course, if there is an $f \in F$ such that $P_s^{\mathtt{a}} = P_f$ then we let $Max_s^{\mathtt{sf}} = Max_f^{\mathtt{sf}}$. Similarly we find $Min_s^{\mathtt{sf}}$, as well as for $Max_s^{\mathtt{bm}}$ and $Min_s^{\mathtt{bm}}$.

The shear forces at each section s's aft endpoint can then be calculated. If $s \in S^{\mathtt{f}}$ ($s \in S^{\mathtt{a}}$), then the shear force at the aft endpoint equals the total weight w_s of the sections fore (aft) $P_s^{\mathtt{a}}$ minus the (positive) buoyancy of the sections fore (aft) $P_s^{\mathtt{a}}$, this is given in (14) below. For each of the sections, we then require these shear forces to be within limits (15).

Similarly, the bending moment at section s's aft endpoint is calculated as the sum of resulting forces of sections s' ($w_{s'} - b_{s',wTotal}$) lying fore (aft) $P_s^{\mathtt{a}}$ times the distance from $P_s^{\mathtt{a}}$ to s''s (longitudinal) midpoint (16). Naturally, these values are then required to lie within the calculated limits (17).

$$\forall s \in S: \; sf_s = \sum_{s' \epsilon S'_s} (w_{s'} - b_{s',wTotal}), \tag{14}$$

$$\text{where } S'_s = \begin{cases} \{ s' \in S^{\mathrm{f}} \mid P^{\mathrm{f}}_{s'} \geq P^{\mathrm{f}}_{s} \} & \text{if } s \in S^{\mathrm{f}} \\ \{ s' \in S^{\mathrm{a}} \mid P^{\mathrm{a}}_{s'} < P^{\mathrm{a}}_{s} \} & \text{if } s \in S^{\mathrm{a}} \end{cases}.$$

$$\forall s \in S: \; Min^{\mathrm{sf}}_s \leq sf_s \leq Max^{\mathrm{sf}}_s \tag{15}$$

$$\forall s \in S: \; bm_s = \sum_{s' \epsilon S'_s} (w_{s'} + b_{s',wTotal}) \cdot |P^{\mathrm{a}}_s - \frac{P^{\mathrm{f}}_{s'} - P^{\mathrm{a}}_{s'}}{2}| \tag{16}$$

$$Min^{\mathrm{bm}}_s \leq bm_s \leq Max^{\mathrm{bm}}_s. \tag{17}$$