The footprint form of a matrix: Definition, properties, and an application

The footprint form of a matrix: definition, properties, and an application

Elvio G. Amparore

Università di Torino, Dipartimento di Informatica, Torino, Italy

Gianfranco Ciardo*, Andrew S. Miner

Iowa State University, Department of Computer Science, Ames, Iowa, USA

Abstract

We propose an alternative to the well-known row echelon form of a matrix, focused on its "footprint", that is, on the position of both the leading and trailing nonzero entries of each row. When a matrix is in footprint form, its leading entries are all in different positions (as in the echelon form) but, in addition, its trailing entries are also in different positions. This new form has several interesting properties. In particular, a matrix in footprint form has the smallest footprint among the matrices obtainable from it through elementary row operations, and supports an efficient way to compute the rank of any of its submatrices with a particular shape. This is critical for our application: scoring potential variable orders for the decision diagram encoding all solutions to a set of linear integer constraints, with the goal of finding an order that minimizes the decision diagram size. We then experimentally evaluate the effectiveness of this scoring, on several problem instances small enough to compare all possible orders.

Keywords: canonical matrix forms, integer linear constraints, multivalued decision diagrams.
2020 MSC: 15A03, 15A21, 68R07, 90C10,

1. Background

It is well known that, given an $r_{orig} \times c$ real matrix $\mathbf{A}_{orig}$ (assumed to have no zero columns), we can apply a sequence of the three *elementary row operations*, (1) swapping two rows, (2) multiplying a row by a nonzero value, and (3) adding a multiple of a row to another row, to transform it into a *left row echelon form* (for brevity, simply *echelon form*) matrix $\mathbf{A}_e$. In such form,

*Corresponding author
Email addresses: amparore@di.unito.it (Elvio G. Amparore), ciardo@iastate.edu (Gianfranco Ciardo), asminer@iastate.edu (Andrew S. Miner)

$$\mathbf{A}_e = \begin{array}{|cccccccccccc|}
\hline
l_1 & - & - & - & - & - & - & - & - & - & - & - \\
\hline
0 & 0 & 0 & l_2 & - & - & - & - & - & - & - & - \\
\hline
0 & 0 & 0 & 0 & 0 & l_3 & - & - & - & - & - & - \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & l_4 & - & - & - \\
\hline
\end{array}$$

$$\begin{array}{cccccccccccc}
\lambda_1 & & & \lambda_2 & & \lambda_3 & & & \lambda_4 & & & \\
\end{array}$$

$$\begin{array}{cccccccccccc}
 & & & & \tau_1 & & & & \tau_3 & \tau_4 & & \tau_2 \\
\end{array}$$

$$\mathbf{A}_f = \begin{array}{|cccccccccccc|}
\hline
l_1 & - & - & - & t_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & l_2 & - & - & - & - & - & - & - & t_2 \\
\hline
0 & 0 & 0 & 0 & 0 & l_3 & - & - & t_3 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & l_4 & t_4 & 0 & 0 \\
\hline
\end{array}$$

$$\begin{array}{cccccccccccc}
\lambda_1 & & & \lambda_2 & & \lambda_3 & & & \lambda_4 & & & \\
\end{array}$$

Figure 1: Echelon form (top) and footprint form (bottom).

the first $r \leq \min\{c, r_{orig}\}$ rows are nonzero, the remaining $r_{orig} - r$ rows are zero and the *leading* (i.e., first nonzero) entry of row i, for $1 < i \leq r$, is in position $\lambda_i > \lambda_{i-1}$, implying that the leading positions form a "ladder" $1 = \lambda_1 < \lambda_2 < \cdots < \lambda_r \leq c$. Of course, r is the rank of $\mathbf{A}_{orig}$, as well as that of $\mathbf{A}_e$; thus, from now on, we assume that the $r_{orig} - r$ zero rows are removed from $\mathbf{A}_e$, and we let $\mathcal{A}$ be the set of matrices obtainable from $\mathbf{A}_e$ through elementary row operations, all of them having rank r and size $r \times c$.

Assuming no zero columns excludes the degenerate case $\mathbf{A}_{orig} = \mathbf{0}$, thus set $\mathcal{A}$ contains an (uncountably) infinite number of matrices in echelon form, since given one such matrix, we can multiply one of its rows by a nonzero value or add a multiple of row j to row i, subject to $i < j$, and the result matrix is still in echelon form. However, remarkably, all these echelon form matrices have the same leading positions $\lambda_1, ..., \lambda_r$ [**?**]. Figure 1 on the top shows a matrix $\mathbf{A}_e$ in echelon form, where "$-$" indicates an arbitrary value (i.e., zero or nonzero, but still subject to the constraint that no column can contain only zero entries), and $l_1, ..., l_r$ are the (nonzero) leading entries of the r rows.

2. The footprint form of a matrix

Instead of just requiring an echelon (ladder) shape formed by the leading entries on the bottom left, it is possible to define a different matrix form by focusing on the positions of both the leading and trailing nonzero entries of each row.

First, we need to define the *footprint* of a matrix. Recall that, unlike for a set, the elements in a *multiset* are still not ordered, but can appear multiple times, i.e., $\{\!\{a, b, a\}\!\} = \{\!\{a, a, b\}\!\}$ and $|\{\!\{a, a, b\}\!\}| = 3$.

Definition 1. Given an $r \times c$ matrix $\mathbf{A}' \in \mathcal{A}$, its *footprint* is the multiset

$$\Phi(\mathbf{A}') = \{\!\{[\lambda'_1, \tau'_1], ..., [\lambda'_r, \tau'_r]\}\!\},$$

where $[\lambda'_i, \tau'_i]$ is the (integer) interval of column positions between the leading nonzero position λ'_i and trailing nonzero position τ'_i of row i of $\mathbf{A}'$, for $1 \leq i \leq r$. Let $\mathcal{F} = \{\Phi(\mathbf{A}') : \mathbf{A}' \in \mathcal{A}\}$ be the set of footprints of matrices in $\mathcal{A}$. □

Then, the new matrix form we are interested to explore in this paper is defined as follows.

Definition 2. An $r \times c$ matrix is in *row footprint form* (for brevity, simply *footprint form*) if all its leading entries are in different positions and all its trailing entries are in different positions. □

Definition 3. An $r \times c$ *row footprint form* matrix is an *ordered row footprint form* matrix (for brevity, simply *ordered footprint form*) iff it is also in row-echelon form, i.e. the rows are ordered such that all leading entries are ordered $(\lambda_1 < \cdots < \lambda_r)$. □

Algorithm 2, similar to Gauss-Jordan elimination, suffices to show that an ordered footprint form can always be achieved. The algorithm first transforms $\mathbf{A}_{orig}$ into an echelon matrix $\mathbf{A}_e$, with all leading entries in different positions $(1 \leq \lambda_1 < \cdots < \lambda_r \leq c)$, using Gaussian elimination (shown as Algorithm 1, for completeness). Then, the algorithm considers each column j of $\mathbf{A}_e$ in reverse order (from c downward), and ensures that column j contains at most one trailing entry: if the set $\mathcal{T}_j$ of rows with a trailing position in column j is not empty, then the bottom-most row i is used to eliminate any non-zero trailing entries in position j for all rows i' above row i (by multiplying row i by $-\mathbf{A}_e[i', j]/\mathbf{A}_e[i, j]$ and adding the result to row i'). Note this will never change the leading position of row i', since $\lambda_{i'} < \lambda_i$, and will reduce the trailing position of row i' by at least one. Then, τ_i will remain equal to j, and no other row will have trailing position j. Moreover, since j decreases, any successive iteration of the loop will not include row i in set $\mathcal{T}_j$, which means that no other row will end up having a non-zero trailing entry in position τ_i. The matrix $\mathbf{A}_f$ obtained from $\mathbf{A}_e$ at the end of the algorithm must be in ordered footprint form: it remains in echelon form, with the same leading entries and values as $\mathbf{A}_e$, but now each trailing position $\tau_1, ..., \tau_r$ is unique. In general, any row permutation of ordered footprint form will still be in footprint form, but will destroy the echelon form. Moreover, preserving the left ladder for the leading entries, i.e., preserving the echelon form, means that we cannot control the order of the trailing positions; for example, $\tau_1 < \tau_3 < \tau_4 < \tau_2$ in the footprint form matrix $\mathbf{A}_f$ shown in the bottom of Figure 1. If we used Algorithm 2 to build $\mathbf{A}_f$, it is easy to see that the fourth rows of $\mathbf{A}_e$ and of $\mathbf{A}_f$ coincide (we will actually see that all footprint-form matrices in $\mathcal{A}$ share the same footprint, so the zero/nonzero pattern of the rows with a leading entry in position λ_4 in $\mathbf{A}_e$ and in $\mathbf{A}_f$ must coincide). Then, the last two entries of the fourth row of $\mathbf{A}_e$ must be zero, as well as the last two entries of the third row of $\mathbf{A}_e$, while the entry in position τ_4 of the third row of $\mathbf{A}_e$ may happen to be zero, or it may be nonzero (in the latter case, Algorithm 2 added an appropriate multiple of the fourth row of $\mathbf{A}_e$ to the third row of $\mathbf{A}_e$ to ensure that this entry is zero in $\mathbf{A}_f$), but none of these observations is of

Algorithm 1 Put $\mathbf{A} \in \mathbb{R}^{r \times c}$ with no 0 columns into row echelon form

procedure REF(**inout** real matrix $\mathbf{A}$, **inout** integer r, **in** integer c)
  $j \leftarrow 1$; $l \leftarrow 0$; ▷ l is the number of known leading positions
  while $(l < r) \wedge (j \leq c)$ **do**
    if $\exists i \in \{l+1, ..., r\}$ such that $\mathbf{A}[i, j] \neq 0$ **then**
      $l \leftarrow l + 1$; $\lambda_l \leftarrow j$;
      swap rows $\mathbf{A}[i, \bullet] \leftrightarrow \mathbf{A}[l, \bullet]$;
      for all $i' \in \{l+1, ..., r\}$ **do** ▷ Eliminate entries below $\mathbf{A}[l, j]$
        $\mathbf{A}[i', \bullet] \leftarrow \mathbf{A}[i', \bullet] - (\mathbf{A}[i', j]/\mathbf{A}[l, j]) \cdot \mathbf{A}[l, \bullet]$;
    $j \leftarrow j + 1$;
  $r \leftarrow l$; ▷ Rows below l, if any, are all zeroes and are removed

Algorithm 2 Put $\mathbf{A} \in \mathbb{R}^{r \times c}$ with no 0 columns into ordered row footprint form

procedure ORFF(**inout** real matrix $\mathbf{A}$, **inout** integer r, **in** integer c)
  $\mathbf{A} \leftarrow \text{REF}(\mathbf{A}, r, c)$; ▷ Put $\mathbf{A}$ into row-echelon form, may reduce r
  for all $i \in \{1, ..., r\}$ **do** $\tau_i \leftarrow \max\ \{j : \mathbf{A}[i, j] \neq 0\}$;
  $j \leftarrow c$; $t \leftarrow 0$; ▷ t is the number of surely-unique trailing positions
  while $t < r - 1$ **do**
    if $(\mathcal{T}_j \leftarrow \{i' : \tau_{i'} = j\}) \neq \emptyset$ **then**
      $i \leftarrow \max\ \mathcal{T}_j$;
      for all $i' \in \mathcal{T}_j \setminus \{i\}$ **do** ▷ Eliminate trailing above $\mathbf{A}[i, j]$
        $\mathbf{A}[i', \bullet] \leftarrow \mathbf{A}[i', \bullet] - (\mathbf{A}[i', j]/\mathbf{A}[i, j]) \cdot \mathbf{A}[i, \bullet]$;
        $\tau_{i'} \leftarrow \max\ \{j : \mathbf{A}[i', j] \neq 0\}$; ▷ Reduce $\tau_{i'}$ by at least one
      $t \leftarrow t + 1$; ▷ τ_i is now fixed
    $j \leftarrow j - 1$;

importance when discussing $\mathbf{A}_e$, as the echelon form only focuses on the leading entries.

Again, the set of ordered footprint form matrices in $\mathcal{A}$ is uncountably infinite since, given a matrix in ordered footprint form, it remains in ordered footprint form if we multiply one of its rows by a nonzero value. Of course, the set of footprint form matrices in $\mathcal{A}$ is also countably infinite, since this is a superset of the ordered footprint form matrices in $\mathcal{A}$.

Lemma 1. Since matrix $\mathbf{A}_{orig}$ has no zero columns, a footprint form matrix $\mathbf{A}_f \in \mathcal{A}$ has footprint $\Phi(\mathbf{A}_f) = \{\!\{[\lambda_1, \tau_1], ..., [\lambda_r, \tau_r]\}\!\}$ with $\lambda_i = 1$ and $\tau_j = c$ for exactly one i and j.

Proof. Since $\mathbf{A}_{orig}$ has no zero columns, so do all matrices in $\mathcal{A}$, including $\mathbf{A}_e$ and $\mathbf{A}_f$, as no sequence of elementary row operations can transform a nonzero column into a zero column. Then, every matrix in $\mathcal{A}$ contains at least one nonzero element in column 1 but, in particular, $\mathbf{A}_f$ must contain *exactly* one nonzero element in column 1. Otherwise, $\mathbf{A}_f$ would have two rows with leading

entries in position 1, i.e., $\lambda_i = 1$ for some row i. A similar argument shows that $\tau_j = c$ for some row j. □

3. Properties of the footprint form

We now define the concept of a "minimal" footprint of a matrix. To explain this, we first need to define how to compare footprints.

Definition 4. Define binary relation $\preceq$ over $\mathcal{F}$ such that, given matrices $\mathbf{A}'$ and $\mathbf{A}''$ in $\mathcal{A}$, their footprints $\Phi(\mathbf{A}') = \{\!\{a_1, ..., a_r\}\!\}$ and $\Phi(\mathbf{A}'') = \{\!\{b_1, ..., b_r\}\!\}$ satisfy

$$\Phi(\mathbf{A}') \preceq \Phi(\mathbf{A}'') \quad \Leftrightarrow \quad \exists\, \pi \in \mathcal{P}_r \ \text{ s.t. } \ \forall i \in \{1, ..., r\},\ a_i \subseteq b_{\pi(i)}, \tag{1}$$

where $\mathcal{P}_r$ is the set of permutations of $(1, ..., r)$. □

Theorem 1. Relation $\preceq$ is a partial order over $\mathcal{F}$.

Proof. Given $\phi = \Phi(\mathbf{A}) = \{\!\{a_1, ..., a_r\}\!\}$, $\phi' = \Phi(\mathbf{A}') = \{\!\{a'_1, ..., a'_r\}\!\}$, and $\phi'' = \Phi(\mathbf{A}'') = \{\!\{a''_1, ..., a''_r\}\!\} \in \mathcal{F}$, we have:

Reflexivity: $\phi \preceq \phi$ follows from the reflexivity of $\subseteq$, and choosing the identity permutation.

Antisymmetry: Suppose $\phi \preceq \phi'$ using permutation π and $\phi' \preceq \phi$ using permutation σ, i.e., for $i \in \{1, ..., r\}$, $a_i \subseteq a'_{\pi(i)}$ and $a'_i \subseteq a_{\sigma(i)}$. Since ϕ is finite, it contains at least one maximum element a_k satisfying $\{a_i \in \phi : a_k \subseteq a_i\} = \{a_k\}$, which, combined with $a_k \subseteq a'_{\pi(k)}$, $a'_{\pi(k)} \subseteq a_{(\sigma\cdot\pi)(k)}$, and the transitivity and antisymmetry of $\subseteq$, implies $a_{(\sigma\cdot\pi)(k)} = a_k$, thus $a_k = a'_{\pi(k)}$.
If we remove a_k from both ϕ and ϕ', we can repeat the argument and conclude by induction that, for $i \in \{1, ..., r\}$, $a_i = a'_{\pi(i)}$, thus $\phi = \phi'$.

Transitivity: Suppose $\phi \preceq \phi'$ using permutation π and $\phi' \preceq \phi''$ using permutation σ. Letting $\rho = \pi \cdot \sigma$, it follows that, for $i \in \{1, ..., r\}$, $a_i \subseteq a''_{\rho(i)}$, thus $\phi \preceq \phi''$ using permutation ρ. □

Now that we have established how to compare footprints, we can prove the main property of matrices in footprint form, given in Theorem 2. First, we need the following lemma.

Lemma 2. Given a full-rank $n \times n$ matrix $\mathbf{C}$, there is a permutation $\pi \in \mathcal{P}_n$ such that, for $i \in \{1, ..., n\}$, $\mathbf{C}[\pi(i), i] \neq 0$.

Proof. This is immediate: since $\mathbf{C}$ has full rank, $\det(\mathbf{C}) \neq 0$. Then, Leibniz formula $\det(\mathbf{C}) = \sum_{\pi\in\mathcal{P}_n} (sign(\pi) \prod_{i=1}^{n} \mathbf{C}[\pi(i), i])$ can be nonzero only if at least one $\pi \in \mathcal{P}_n$ is such that, for $i \in \{1, ..., n\}$, $\mathbf{C}[\pi(i), i] \neq 0$. □

Theorem 2. Given matrix $\mathbf{A}_{orig}$ and a footprint form matrix $\mathbf{A}_f$ obtained from it, the footprint $\Phi(\mathbf{A}_f) = \{\!\{[\lambda_1, \tau_1], ..., [\lambda_r, \tau_r]\}\!\}$ is the least element of $(\mathcal{F}, \preceq)$, i.e., for any $\mathbf{A}' \in \mathcal{A}$, we have $\Phi(\mathbf{A}_f) \preceq \Phi(\mathbf{A}')$. Furthermore, $\mathcal{F}$ contains only multisets whose elements are of the form $[\lambda, \tau]$ where $\lambda \in \{\lambda_1, ..., \lambda_r\}$ and $\tau \in \{\tau_1, ..., \tau_r\}$, subject to the constraint that, if $\lambda = \lambda_i$ then $\tau \geq \tau_i$ and, if $\tau = \tau_i$, then $\lambda \leq \lambda_i$.

PROOF. Since any matrix $\mathbf{A}' \in \mathcal{A}$ can be obtained from $\mathbf{A}_f$ using a sequence of elementary row operations, there exists a *rank decomposition* $\mathbf{A}' = \mathbf{C}' \cdot \mathbf{A}_f$ where $\mathbf{C}'$ is a full-rank $r \times r$ matrix, and Lemma 2 guarantees the existence of a permutation $\pi \in \mathcal{P}_r$ such that, for $i \in \{1, ..., r\}$, $\mathbf{C}'[\pi(i), i] \neq 0$. However, observe that, since row j of $\mathbf{A}'$ is the linear combination $\mathbf{A}'[j, \bullet] = \sum_{i=1}^{n} \mathbf{C}'[j, i] \cdot \mathbf{A}_f[i, \bullet]$ of the rows of $\mathbf{A}_f$, all with different leading and different trailing positions, we have

$$\lambda'_j = \min\{\lambda_i : \mathbf{C}'[j, i] \neq 0\} \quad \text{and} \quad \tau'_j = \max\{\tau_i : \mathbf{C}'[j, i] \neq 0\}.$$

Then, we conclude that $\Phi(\mathbf{A}_f) \preceq \Phi(\mathbf{A}')$, since permutation π can be used to show that, for $i \in \{1, ..., r\}$, $[\lambda_i, \tau_i] \subseteq [\lambda'_{\pi(i)}, \tau'_{\pi(i)}]$. This argument also implies that all leading (or trailing) positions of any matrix $\mathbf{A}'$ obtainable from $\mathbf{A}_f$ must equal some leading (resp. trailing) position of $\mathbf{A}_f$ and, furthermore, that, if row i of $\mathbf{A}_f$ is part of the linear combination for row j of $\mathbf{A}'$, then both $\lambda'_j \leq \tau_i$ and $\tau'_j \geq \tau_i$ hold, implying that, if $\lambda = \lambda_i$, then $\tau \geq \tau_i$ and, if $\tau = \tau_i$, then $\lambda \leq \lambda_i$. □

Corollary 1. All footprint form matrices in $\mathcal{A}$ have the same (least) footprint $\Phi(\mathbf{A}_f)$.

PROOF. It immediately follows from Theorem 2 that any footprint form matrix has a footprint that is the (unique) least element of $(\mathcal{F}, \preceq)$. □

Lemma 3. $(\mathcal{F}, \preceq)$ admits a greatest element.

PROOF. Given a footprint form matrix $\mathbf{A}_f$, from Lemma 1, there is a row i in $\mathbf{A}_f$ with $\lambda_i = 1$, and a row j in $\mathbf{A}_f$ with $\lambda_j = c$. We can simply add row i to each other row, and then add row j to each other row. The resulting matrix has footprint $\Phi = \{\!\{[1, c], ..., [1, c]\}\!\}$ and, obviously, no other matrix obtainable from $\mathbf{A}_f$ can have a larger footprint. □

We now show how the footprint $\Phi(\mathbf{A}_f) = \{\!\{[\lambda_1, \tau_1], ..., [\lambda_r, \tau_r]\}\!\}$ of a footprint form matrix $\mathbf{A}_f$ can be used to efficiently determine the rank of some of its submatrices, as well as the rank of certain submatrices of any matrix in $\mathcal{A}$.

Theorem 3. Let $\mathbf{A}[\mathcal{R}, \mathcal{C}]$ be the submatrix of $\mathbf{A}$ corresponding to the sets of rows $\mathcal{R}$ and columns $\mathcal{C}$. Then, given any footprint form matrix $\mathbf{A}_f \in \mathcal{A}$, it holds that

(1) $rank(\mathbf{A}_f[\mathcal{R}, \{1, ..., n\}]) = |\{i \in \mathcal{R} \ : \ \lambda_i \leq n\}|$;

(2) $rank(\mathbf{A}_f[\mathcal{R}, \{n, ..., c\}]) = |\{i \in \mathcal{R} \ : \ \tau_i \geq n\}|$.

In addition, for any matrix $\mathbf{A} \in \mathcal{A}$, it holds that

(3) $rank(\mathbf{A}[\{1, ..., r\}, \{1, ..., n\}]) = |\{i \ : \ \lambda_i \leq n\}|$;

(4) $rank(\mathbf{A}[\{1, ..., r\}, \{n, ..., c\}]) = |\{i \ : \ \tau_i \geq n\}|$.

PROOF. For (1), if $\lambda_i > n$, row $i \in \mathcal{R}$ of $\mathbf{A}_f[\mathcal{R}, \{1, ..., n\}]$ contains only zeros, thus does not contribute to its rank. If instead $\lambda_i \leq n$, row $i \in \mathcal{R}$ of $\mathbf{A}_f[\mathcal{R}, \{1, ..., n\}]$ cannot be obtained through a sequence of elementary row operations on the other nonzero rows of $\mathbf{A}_f[\mathcal{R}, \{1, ..., n\}]$, since they all have leading entries in different positions, and none in position λ_i. Thus, all nonzero rows in $\mathbf{A}_f[\mathcal{R}, \{1, ..., n\}]$ are linearly independent, and the rank of $\mathbf{A}_f[\mathcal{R}, \{1, ..., n\}]$ is $|\{i \in \mathcal{R} \ : \ \lambda_i \leq n\}|$. An analogous argument applies for (2), if we consider the positions of its trailing entries. For (3) and (4), consider cases (1) and (2) when $\mathcal{R} = \{1, ..., r\}$ and observe that, since all rows of $\mathbf{A}$ can be obtained as linear combinations of rows of $\mathbf{A}_f$ and vice versa, the same holds for their restriction to the first n columns or the last $c - n + 1$ columns. □

Note that an analogous theorem restricted to cases (1) and (3) could be stated for the echelon form, since the leading entries have similar properties in the two forms. However, the echelon form has no concept of trailing entries, thus offers no immediate way to evaluate the rank for cases (2) and (4).

4. Reduced footprint form

It is known that any echelon matrix can be transformed into a *reduced echelon form* using a sequence of elementary row operations, so that not only is it in echelon form, but also its leading entries $l_1, ..., l_r$ all are equal to 1 and all the entries above them are equal to 0. Remarkably, this reduced echelon form is *canonical*, i.e., given $\mathbf{A}_{orig}$ only one reduced echelon matrix $\mathbf{A}_{re}$ can be obtained from it. An example of reduced echelon matrix $\mathbf{A}_{re}$ is shown at the top of Figure 2, where the grayed entries above each leading entry must be 0.

We now derive an analogous form for footprint form matrices, called *reduced footprint form*, and show that it, too, is a canonical form.

Definition 5. An $r \times c$ matrix is in *reduced row footprint form* (for brevity, simply *reduced footprint form*) iff it is in ordered row footprint form, all the leading entries $l_1, ..., l_r$ equal 1, and all entries above each trailing entry are 0. □

In other words, the only difference between the reduced echelon form and the reduced footprint form is that the former requires leading entries to have zeros above them, while the latter requires the trailing entries to have zeros above them. The bottom of Figure 2 shows an example, where all the gray entries above each trailing entry must be 0.

A reduced footprint form matrix can always be obtained, using Algorithm 3 (again, similar to Gauss-Jordan elimination). The algorithm first obtains an

$$\mathbf{A}_{re} = \begin{array}{|cccccccccccc|}\hline 1 & - & - & 0 & - & 0 & - & - & 0 & - & - & - \\ 0 & 0 & 0 & 1 & - & 0 & - & - & 0 & - & - & - \\ 0 & 0 & 0 & 0 & 0 & 1 & - & - & 0 & - & - & - \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & - & - & - \\ \hline \end{array}$$
$$\begin{array}{cccccccccccc} \lambda_1 & & & \lambda_2 & & \lambda_3 & & & \lambda_4 & & & \end{array}$$

$$\mathbf{A}_{rf} = \begin{array}{|cccccccccccc|}\hline & & & & \tau_1 & & & & \tau_3 & \tau_4 & & \tau_2 \\ 1 & - & - & - & r_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & - & - & - & - & 0 & 0 & - & r_2 \\ 0 & 0 & 0 & 0 & 0 & 1 & - & - & r_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & r_4 & 0 & 0 \\ \hline \end{array}$$
$$\begin{array}{cccccccccccc} \lambda_1 & & & \lambda_2 & & \lambda_3 & & & \lambda_4 & & & \end{array}$$

Figure 2: Reduced echelon form (top) and reduced footprint form (bottom).

Algorithm 3 Put $\mathbf{A} \in \mathbb{R}^{r \times c}$ with no 0 columns into reduced row footprint form

```
1: procedure RRFF(inout real matrix A, inout integer r, in integer c)
2:     A ← ORFF(A, r, c);        ▷ Put A into ordered row footprint form, may reduce r
3:     i ← r;
4:     while i > 0 do
5:         for all i' < i such that A[i', τ_i] ≠ 0 do        ▷ Eliminate entries above A[i, τ_i]
6:             A[i', •] ← A[i', •] − (A[i', τ_j]/A[i, τ_i]) · A[i, •];
7:         A[i, •] ← (1/A[i, λ_i]) · A[i, •];        ▷ Force leading entry to 1
8:         i ← i − 1;
```

ordered row footprint form matrix, utilizing Algorithm 2. It then considers each row i of $\mathbf{A}_f$ in reverse order (from r to 1), and for each row i' above i with $\mathbf{A}_f[i', \tau_i] \neq 0$, it eliminates $\mathbf{A}_f[i', \tau_i]$ by multiplying row i by $-\mathbf{A}_f[i', \tau_i]/\mathbf{A}_f[i, \tau_i]$ and adding the result to row i'. This elementary row operation does not change the leading position of row i', since $\lambda_{i'} < \lambda_i$. It also does not change the trailing position of row i': because the matrix is in (ordered) row footprint form, the trailing positions are unique, and since $\mathbf{A}_f[i', \tau_i]$ is not zero, it follows that $\tau_{i'}$ must be greater than τ_i. After this step, the leading entry of row i is forced to become 1, by multiplying row i by $1/\mathbf{A}_f[i, \lambda_i]$. The matrix $\mathbf{A}_{rf}$ resulting from this process is obviously in reduced footprint form, since it remains in ordered footprint form, and now all leading entries are 1, and each trailing entry has only 0 entries above it. A computer program implementing this algorithm to obtain the reduced row footprint form from an arbitrary matrix can be found at `https://github.com/amparore/reduced_row_footprint_form`.

Theorem 4. The reduced footprint form $\mathbf{A}_{rf}$ of a matrix $\mathbf{A}$ is unique.

Proof. The proof of uniqueness for the reduced footprint form is analogous to

the one for the reduced echelon form. If, by contradiction, two reduced footprint matrices $\mathbf{A}_{rf}$ and $\mathbf{A}'_{rf}$ could be obtained from $\mathbf{A}_{orig}$, then we would be able to transform $\mathbf{A}_{rf}$ into $\mathbf{A}'_{rf}$ using a sequence of elementary row operations, i.e., any row of $\mathbf{A}'_{rf}$ would equal some linear combination of rows of $\mathbf{A}_{rf}$. The only possible linear combination of rows of $\mathbf{A}_{rf}$ resulting in row r of $\mathbf{A}'_{rf}$ is just row r of $\mathbf{A}_{rf}$ itself, since both $\mathbf{A}_{rf}$ and $\mathbf{A}'_{rf}$ are in echelon form, thus they have the same leading positions $\lambda_1, ..., \lambda_r$, and any linear combination involving more than row r would have a leading position to the left of λ_r. Thus row r of $\mathbf{A}_{rf}$ equals row r of $\mathbf{A}'_{rf}$, because of the normalization imposed on the leading entries. Considering now row $r-1$ of $\mathbf{A}'_{rf}$, a similar argument implies that rows above $r-1$ of $\mathbf{A}_{rf}$ cannot be involved in its linear combination; however, entry in column τ_r of row $r-1$ of both $\mathbf{A}_{rf}$ and $\mathbf{A}'_{rf}$ must be 0 (since τ_r is the trailing position for row r in both matrices), thus row r cannot be part of the linear combination either and, again, row $r-1$ of $\mathbf{A}_{rf}$ must equal row $r-1$ of $\mathbf{A}'_{rf}$. Continuing the argument by induction, we obtain that $\mathbf{A}_{rf} = \mathbf{A}'_{rf}$. □

We observe that the choice of achieving a canonical footprint form by keeping the ladder shape on the lower left and normalizing the leading entries to 1 is completely arbitrary. For example, we could instead choose to have the ladder shape in the upper left corner (the trailing entries would still not necessarily form a ladder shape, but they would be required to have zero entries below, not above, them), or a ladder shape in the lower right corner (the trailing entries would then form a ladder, while the leading entries would not necessarily do so but would have zero entries above them), or a ladder shape in the upper right corner (the trailing entries would then form a ladder, while the leading entries would not necessarily do so but would have zero entries below them). Independently of this choice, we could then choose to normalize the trailing entries to 1 instead of the leading entries. Remarkably, all of these canonical forms still have the same footprint, since they are in footprint form so they share the same (least) footprint $\Phi(\mathbf{A}_f)$; this is also easy to see since they are all obtainable from each other through just row normalizations and row permutations, and neither of these two operations change the footprint of a matrix.

Table 1 summarizes the commonalities and differences between row echelon form (REF), row footprint form (RFF), ordered row footprint form (ORFF), reduced row echelon form (RREF), and reduced row footprint form (RRFF), of a $r_{orig} \times c$ real matrix $\mathbf{A}_{orig}$ with no zero columns. Note that a matrix in REF, with all leading entries equal to 1, and having zeroes above the leading *and* the trailing entries, is in *both* RREF and RRFF simultaneously. However, because RREF and RRFF are unique representations, if the RREF and RRFF matrices do not coincide for a given input matrix $\mathbf{A}_{orig}$, then they cannot be made to coincide through elementary row operations. There is a similar relationship between RREF and RFF matrices: if the RREF matrix has distinct trailing entry positions, then it is in ordered RFF; if not, then it cannot be put into RREF and RFF simultaneously.

REF	**RFF**
r rows, c columns, rank is $r = rank(\mathbf{A}_{orig})$	
$\lambda_1, ..., \lambda_r$ are all different	
$\lambda_1 < \cdots < \lambda_r$	$\lambda_1 < \cdots < \lambda_r$ for *ordered* RFF
Trailing positions are not considered	$\tau_1, ..., \tau_r$ are all different
RREF	**RRFF**
Unique matrix for a given input	
All leading entries are equal to 1	
It is also a REF	
May also be a ORFF	It is also a ORFF
May also be a RRFF	May also be a RREF
Entries above leading entries are 0	Entries above trailing entries are 0

Table 1: Comparison between row echelon and row footprint forms.

5. An application

We now describe the application that motivated our investigation into the definition and properties of the footprint form.

5.1. Constraint solving with multivalued decision diagrams

The constraint solving problem.. A *bounded integer constraint problem* is specified by a finite set of variables $\mathcal{V} = \{x_1, ..., x_L\}$, where each x_k takes value over $\mathcal{X}_k = \{0, ..., n_k\}$ for some bound $n_k \in \mathbb{N}$, and a finite set $\mathcal{D} = \{d_1, ..., d_N\}$ of *constraints*, each of the form $d_j : \mathcal{X} \to \{0, 1\}$, where $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_L$. We call any vector $\mathbf{x} \in \mathcal{X}$ an *assignment* of the variables and say that $\mathbf{x}$ *satisfies* d_j if $d_j(\mathbf{x}) = 1$. One key question is then to find a *solution*, i.e., *one* assignment $\mathbf{x}$ that satisfies all the N constraints. Another question is to find the set of *all* solutions, $\mathcal{S}_\mathcal{D} = \{\mathbf{x} \in \mathcal{X} : \forall d_j \in \mathcal{D}, d_j(\mathbf{x}) = 1\}$. This second problem is particularly important if we are interested in finding an optimal solution (according to some criterion) among all solutions. While several types of constraints may arise in practical problems, we limit our discussion to the case of *linear* constraints of the form $d_j(\mathbf{x}) = 1$ iff $\sum_{x_k \in \mathcal{V}} a_{j,k} \cdot x_k = b_j$, where $a_{j,k} \in \mathbb{Z}$ is the weight associated to variable x_k for constraint d_j, and $b_j \in \mathbb{N}$ is a constant. Thus, we are interested in the set of solutions $\mathbf{x} \in \mathcal{X}$ subject to $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$, where $\mathbf{A}$ is an $N \times L$ matrix of integer weights and $\mathbf{b}$ is an $N \times 1$ vector of non-negative integer constants. Let $Supp(d_j) = \{x_k \in \mathcal{V} : a_{j,k} \neq 0\}$ be the *support* of constraint d_j.

In the following, we assume that no constraint d_j has only one variable x_k in its support. This is because such d_j would imply $x_k = b_j/a_{j,k}$. If this value is not a natural number, then the problem has no solution; otherwise, we can eliminate both constraint d_j and variable x_k, by substituting the value $b_j/a_{j,k}$ anywhere variable x_k appears in other constraints. We also assume that each variable appears in at least one constraint, since variables not appearing in any constraint are irrelevant to the solution; this implies that matrix $\mathbf{A}$ has no zero columns.

***Multivalued decision diagrams.*.** A *multivalued decision diagram* (MDD) [**?**] over the set $\mathcal{X}$ defined above is a directed acyclic graph whose nodes are assigned to *levels*. A *nonterminal* node p is at some level $k \in \{1, ..., L\}$, we write $p.lvl = k$, and has $|\mathcal{X}_k|$ outgoing edges labeled with distinct elements of $\mathcal{X}_k$, we write $p[i_k] = q$ if the edge labeled with i_k points to node q, and require that $p.lvl > q.lvl$. The only *terminal* nodes are $\bot$ and $\top$, at level 0. Given node p at level $k \leq l$ and a sequence of variable assignments $\sigma = (x_h \leftarrow i_h, ..., x_{l-1} \leftarrow i_{l-1}, x_l \leftarrow i_l)$, with $h \leq k$, let $p(\sigma)$ be the node reached by following the corresponding edges:

$$
\begin{aligned}
p() &= p \\
p(x_h \leftarrow i_h, ..., x_{l-1} \leftarrow i_{l-1}, x_l \leftarrow i_l) &= \begin{cases} p(x_h \leftarrow i_h, ..., x_k \leftarrow i_k) & \text{if } k < l \\ p[i_k](x_h \leftarrow i_h, ..., x_{k-1} \leftarrow i_{k-1}) & \text{if } k = l. \end{cases}
\end{aligned}
$$

Node q is *reachable* from node p if there exists a sequence σ such that $p(\sigma) = q$. Node p encodes a function f_p defined by $f_p(i_1, ..., i_L) = p(x_1 \leftarrow i_1, ..., x_L \leftarrow i_L)$, thus a set $\mathcal{X}_p = \{(i_1, ..., i_L) \in \mathcal{X} : f_p(i_1, ..., i_L) = \top\}$. In practice, we enforce *canonicity* on MDDs by merging duplicate nodes (nodes p and q at level k are duplicates if $p[i_k] = q[i_k]$ for all $i_k \in \mathcal{X}_k$) and either eliminating all redundant nodes (node p at level k is redundant if $p[0] = \cdots = p[n_k]$), this is the *fully-reduced form* [**?**], or keeping all redundant nodes, except for those where all edges point to $\bot$, and never skipping levels, except when pointing to $\bot$, i.e., if $p[i] = q$, then $q.lvl = p.lvl - 1$ or $q = \bot$, this is the *quasi-reduced form* [**?**].

We can use MDDs to generate all solutions to an integer constraint problem by mapping the L variables to distinct MDD levels, then building the MDD encoding the set $\{\mathbf{x} \in \mathcal{X} : d_j(\mathbf{x}) = 1\}$ for each constraint d_j, and finally computing the intersections of these sets (in some order, since computing the intersection of the sets encoded by two MDDs is a standard MDD operation). For this application, the MDD encoding the set of solutions $\mathcal{S}_\mathcal{D}$ contains redundant nodes only in association with variables not in the support of any constraint; if such variables are eliminated from consideration, the fully-reduced and quasi-reduced forms coincide, thus we assume the conceptually simpler quasi-reduced form, where any nonempty subset of $\mathcal{X}$ is encoded by a "root" node at level L.

***Variable order.*.** While MDDs can often compactly encode large sets, there are sets whose MDD encoding can vary exponentially in size depending on the order π mapping each variable x_k to a (different) level $\pi(k)$, i.e., π is a permutation of $\{1, ..., L\}$. It is known that finding an optimal order (achieving the minimum MDD size, measured in number of nodes) is NP-hard [**?**], and that some sets have an exponential MDD encoding for any order [**?**]. Various polynomial-time heuristics have been proposed to derive reasonably good variable orders, usually based on the intuition that it is best to map "related" variables (e.g., belonging to the support of a constraint) to MDD levels "close" to each other. However, different constraints with non-disjoint supports may suggest incompatible orders. A simple but somewhat effective heuristic is *sum-of-spans* (SOS) [**?**], which seeks a variable order minimizing $\sum_{d_j \in \mathcal{D}} Span(d_j)$,

where, for a given order π, the span of constraint d_j is defined as

$$Span(d_j) = \max\{\pi(x_k) : x_k \in Supp(d_j)\} - \min\{\pi(x_k) : x_k \in Supp(d_j)\} + 1.$$

Minimizing SOS is also NP-hard, but good orders can be computed *statically* (i.e., before beginning the construction of any MDD) using heuristic search algorithms such as simulated annealing [**?**]. However, SOS is not a robust heuristic. For example, if constraint d_3 is the sum of two linear constraints d_1 and d_2, it is redundant, in the sense that, if d_1 and d_2 are satisfied, then so is d_3. Even more obviously, multiple instances of the same constraint are redundant. Such redundant constraints contribute to the value of SOS, and can skew it toward arbitrarily suboptimal choices. Intuitively, we should focus on the "essential" information captured by the constraints when attempting to find a good order.

5.2. Size of the MDD encoding $\mathcal{S}_\mathcal{D}$

Definition 6. For a set of linear constraints $\mathcal{D} = \{d_1, ..., d_N\}$ and a sequence of variable assignments $\sigma = (x_h \leftarrow i_h, ..., x_l \leftarrow i_l)$, with $h \leq l$, define $\mathbf{d}(\sigma) \in \mathbb{Z}^N$ to be the vector of partial sums of the constraints:

$$\mathbf{d}(\sigma)[j] = \sum_{h \leq k \leq l} i_k \cdot a_{j,k}.$$

In matrix form, this is written as $\mathbf{d}(\sigma) = \mathbf{A}[\bullet, \{h, ..., l\}] \cdot \mathbf{i}$, where $\mathbf{i} = [i_h, ..., i_l]$ and $\bullet$ indicates $\{1, ..., N\}$. □

Theorem 5 (Adapted from [?]). Given a set of linear constraints $\mathcal{D}$, an MDD node $p^\star$ encoding $\mathcal{X}_{p^\star} = \mathcal{S}_\mathcal{D}$, an MDD level $k \in [1, ..., L]$, and any two sequences $\sigma = (x_k \leftarrow s_k, ..., x_L \leftarrow s_L)$ and $\sigma' = (x_k \leftarrow s'_k, ..., x_L \leftarrow s'_L)$ with $p^\star(\sigma) \neq \bot$ and $p^\star(\sigma') \neq \bot$, we have $p^\star(\sigma) = p^\star(\sigma')$ if and only if $\mathbf{d}(\sigma) = \mathbf{d}(\sigma')$.

Proof. Let $p = p^\star(\sigma) = p^\star(\sigma')$, at level $k - 1$. Since $p \neq \bot$, there exists a sequence of variable assignments $\gamma = (x_1 \leftarrow s_1, ..., x_{k-1} \leftarrow s_{k-1})$ such that $p(\gamma) = \top$. Thus, $p^\star(\sigma\gamma) = \top$ and $p^\star(\sigma'\gamma) = \top$, where both sequences $\sigma\gamma$ and $\sigma'\gamma$ include all variables, and therefore satisfy all constraints; i.e., $\mathbf{d}(\sigma\gamma) = \mathbf{d}(\sigma'\gamma) = \mathbf{b}$. But we also have that $\mathbf{d}(\sigma) + \mathbf{d}(\gamma) = \mathbf{d}(\sigma\gamma)$ and $\mathbf{d}(\sigma') + \mathbf{d}(\gamma) = \mathbf{d}(\sigma'\gamma)$. Therefore $\mathbf{d}(\sigma) + \mathbf{d}(\gamma) = \mathbf{d}(\sigma') + \mathbf{d}(\gamma)$, and we must have $\mathbf{d}(\sigma) = \mathbf{d}(\sigma')$.

Now, let $\mathbf{d}(\sigma) = \mathbf{d}(\sigma')$. We show by contradiction that $p^\star(\sigma) = p^\star(\sigma')$. Suppose instead $p^\star(\sigma) \neq p^\star(\sigma')$. Let $p = p^\star(\sigma)$ and $q = p^\star(\sigma')$. Since $p \neq q$ and because canonicity implies there are no duplicate nodes, there must exist a sequence of assignments to variables x_{k-1} down to x_1 that causes p and q to reach different terminal nodes. Without loss of generality, let γ be such a sequence and suppose $p(\gamma) = \top$ and $q(\gamma) = \bot$. Then, $\sigma\gamma$ is a complete sequence describing a solution to the set of constraints, i.e., $\mathbf{d}(\sigma) + \mathbf{d}(\gamma) = \mathbf{d}(\sigma\gamma) = \mathbf{b}$. But, since $\mathbf{d}(\sigma) = \mathbf{d}(\sigma')$, we have $\mathbf{d}(\sigma') + \mathbf{d}(\gamma) = \mathbf{b}$, implying that $\sigma'\gamma$ is also a solution to the set of constraints, thus $p^\star(\sigma'\gamma)$ should be $\top$, not $\bot$, a contradiction. □

Theorem 5 implies that all sequences σ leading from the root to node $p \neq \perp$ have the same vector $\mathbf{d}(\sigma)$, thus we can also denote this vector as $\mathbf{d}_p$, independent of the specific sequence σ leading to p.

Definition 7. Constraint $d_j \in \mathcal{D}$ is *active at level* k if $Supp(d_j) \cap \{x_1, ..., x_k\} \neq \emptyset$ and $Supp(d_j) \cap \{x_{k+1}, ..., x_L\} \neq \emptyset$. □

Note that a constraint "starting" at level k, i.e., such that x_k is the highest variable in its support, is not (yet) considered active at level k.

Lemma 4. For any two nodes p and q at level $k > 0$, if constraint d_j is not active at level k, then $\mathbf{d}_p[j] = \mathbf{d}_q[j]$.

PROOF. If constraint d_j is not active at level k, either $Supp(d_j) \cap \{x_1, ..., x_k\} = \emptyset$ (the constraint is not active anymore) or $Supp(d_j) \cap \{x_{k+1}, ..., x_L\} = \emptyset$ (the constraint is not yet active). Then, for any σ leading to node p or q, we have either $\mathbf{d}(\sigma)[j] = b_j$ (in the former case) or $\mathbf{d}(\sigma)[j] = 0$ (in the latter case). □

From Theorem 5, if $\mathbf{d}_p = \mathbf{d}_q$ then $p = q$. Therefore, the number of nodes in the MDD encoding the set of solutions $\mathcal{S}_\mathcal{D}$ is bounded by the number of possible distinct $\mathbf{d}$ vectors that may exist at any level. Each vector $\mathbf{d}_p$ corresponding to a node p at level k is obtained as the product $\mathbf{A}[\bullet, \{k+1, ..., L\}] \cdot \mathbf{x}[\{k+1, ..., L\}]$, where $\mathbf{x}$ is a solution to $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$, thus the dimension of the space of the $\mathbf{d}$ vectors for level k is at most $rank(\mathbf{A}[\bullet, \{k+1, ..., L\}])$. From Lemma 4, the entries of these vectors at level k can differ only for constraints active at level k, i.e., with at least one, but not all, variables in the support of the constraint having assigned values. We now show how these ideas are related to our notion of footprint.

First of all, the $N \times L$ matrix $\mathbf{A}$ is analogous to our matrix $\mathbf{A}_{orig}$, in the sense that it may contain linearly dependent rows (constraints). We can then rearrange $\mathbf{A}$ and $\mathbf{b}$ through elementary row operations (using Algorithm 2) to obtain $\mathbf{A}_f$ and $\mathbf{b}_f$, where the obtained (ordered) footprint form matrix $\mathbf{A}_f$ has size $r \times L$, after discarding the zero rows ($r \leq N$ is the rank of both $\mathbf{A}_f$ and $\mathbf{A}$), and regard $\mathbf{A}_f$ and $\mathbf{b}_f$ as specifying a set of r new constraints $\mathcal{D}' = \{d'_1, ..., d'_r\}$, which are satisfied if and only if the original N constraints $\{d_1, ..., d_N\}$ are satisfied, and define the corresponding vector of partial sums $\mathbf{d}'$, analogous to $\mathbf{d}$. From Theorem 2, $\mathbf{A}_f$ has minimum footprint $\Phi(\mathbf{A}_f)$, and we can use it to characterize, for each level, which elements of $\mathbf{d}'$ may differ. If constraint d'_j has interval $[\lambda_j, \tau_j]$, then it is active if and only if $\lambda_j \leq k < \tau_j$ (when $k = \tau_j$, the top-most variable x_k has not yet been assigned).

Let the number of constraints active at level k be $\alpha(k) = |\{i : \lambda_i \leq k < \tau_i\}|$. Then, because $\mathbf{A}_f$ contains no zero columns and no two leading or two trailing entries can be in the same position, it is clear that $\alpha(L)$ is always 0, $\alpha(1)$ is always 1, and $\alpha(k-1)$, for $1 < k \leq L$, can only be equal to:

- $\alpha(k)$, if there is no $\tau_i = k$ and no $\lambda_i = k$, or there is a $\tau_i = k$ and a $\lambda_j = k$.
- $\alpha(k) + 1$, if there is a $\tau_i = k$ and there is no $\lambda_i = k$.
- $\alpha(k) - 1$, if there is no $\tau_i = k$ and there is a $\lambda_i = k$.

5.3. An example

As an example, consider the following set of $N = 4$ linear constraints:

$$\begin{array}{rrrrrrrrrl} x_1 & & & + x_4 & & & + x_7 & = & B & (d_1) \\ x_1 & & & & + x_5 & + x_6 & + x_7 & = & B & (d_2) \\ & x_2 & + x_3 & + x_4 & & & + x_7 & = & B & (d_3) \\ & x_2 & + x_3 & & + x_5 & + x_6 & + x_7 & = & B & (d_4) \end{array} \tag{2}$$

where the seven non-negative integer variables $x_1, ..., x_7$ are bounded by B. Collecting these four constraints into matrix $\mathbf{A}$, and running Algorithm 2 to transform this matrix into ORFF, we obtain

$$\mathbf{A}_f = \begin{array}{|c|ccc|cccc|} \hline x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\ \hline 1 & -1 & -1 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & -1 & -1 & 0 \\ \hline \end{array} \tag{3}$$

corresponding to the $r = 3$ constraints

$$\begin{array}{rrrrrrrrrll} x_1 & - x_2 & - x_3 & & & & & = & 0 & (d'_1) & \text{from } (d_1) - (d_3) \\ & x_2 & + x_3 & & + x_5 & + x_6 & + x_7 & = & B & (d'_2) & \text{same as } (d_4) \\ & & & x_4 & - x_5 & - x_6 & & = & 0 & (d'_3) & \text{from } (d_3) - (d_4) \end{array}$$

(only three constraints remain because $d_4 = d_2 + d_3 - d_1$). Figure 3 shows the MDD encoding the set of all 15 integer solutions to these constraints, assuming $B = 2$ and using the variable order where variable x_k is assigned to level k. Nonterminal nodes are drawn as arrays of edges and the terminal node $\perp$ and edges to it are omitted for clarity. For each nonterminal node p, the vectors $\mathbf{d}_p$ and $\mathbf{d}'_p$ of partial sums for the original constraints d_1, d_2, d_3, d_4 and the footprint form constraints d'_1, d'_2, d'_3, respectively, are shown to the right of p ($\mathbf{d}_p$ above and $\mathbf{d}'_p$ below). In these vectors, "$\Downarrow$" indicates that the constraint is not yet active and "$\Uparrow$" indicates that the constraint is no longer active.

As is clear from $\mathbf{A}_f$, only constraint d'_1 is active at level 1, so nodes at level 1 correspond to the possible values of the partial sum $-x_2 - x_3$ that can occur in a solution to the constraints. Since there are $B + 1$ such possible values, there can be at most (exactly, in this case) $B + 1 = 3$ nodes at level 1. Similarly, at levels 6 and 3, only d'_2 is active (at level 6, d'_1 and d'_3 are not yet active; at level 3, d'_3 has just become inactive and d'_1 is not yet active).

At levels 5, 4, and 2, there are instead two active constraints. Looking at level 5, the partial sums are $x_6 + x_7$ for d'_2, and $-x_6$ for d'_3. Since $x_6 + x_7$ can take on $B + 1$ values and $-x_6$ can take on $B + 1$ values (in a solution), there can be at most $(B+1)^2$ vectors $\mathbf{d}'$ at level 5. However, not all combinations are possible, because $x_6 + x_7$ cannot be smaller than x_6. Note that all combinations with $\mathbf{d}'[3] \in \{-2, -1, 0\}$ and $\mathbf{d}'[2] \in \{0, 1, 2\}$, subject to $-\mathbf{d}'[3] \leq \mathbf{d}'[2]$, are present at level 5. The same is true for level 4. The case of level 2 is similar but with $\mathbf{d}'[1] \in \{-2, -1, 0\}$ and $\mathbf{d}'[2] \in \{0, 1, 2\}$, subject to $-\mathbf{d}'[1] \leq \mathbf{d}'[2]$.

Recall that we defined the number of constraints active at level k as $\alpha(k) = |\{i : \lambda_i \leq k < \tau_i\}|$, thus we can easily compute its value by inspecting $\mathbf{A}_f$. The

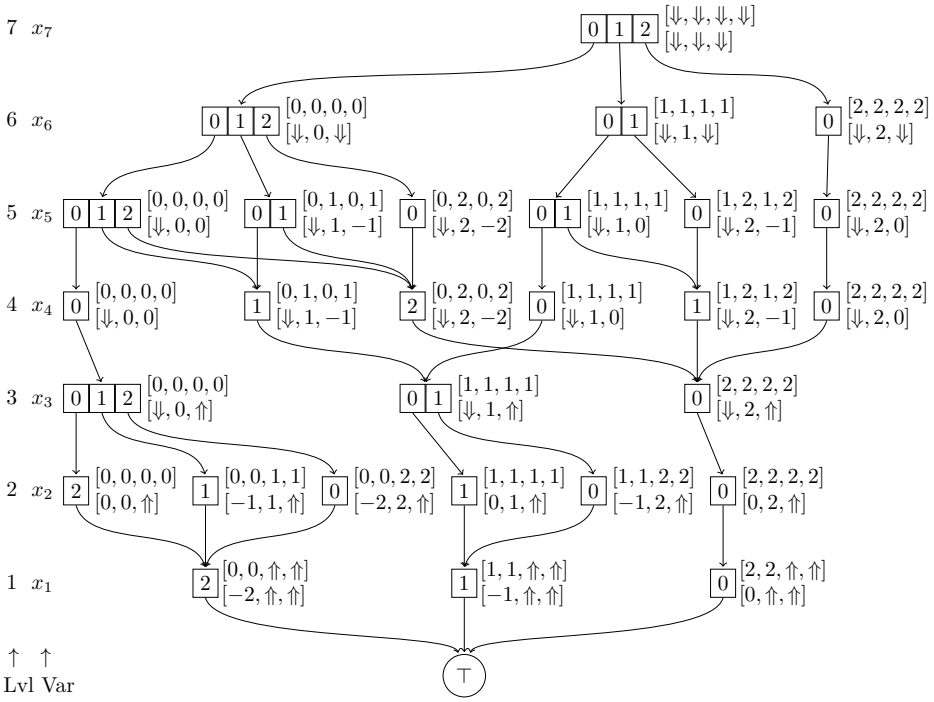


Figure 3: MDD encoding solutions to linear constraints, annotated with **d** vectors.

following table provides, for each level k, the MDD node count and the number of active constraints in it for our example:

Level	1	2	3	4	5	6	7
$\alpha(k)$	1	2	1	2	2	1	0
MDD node count	3*	6*	3	6*	6	3	1

MDD node counts marked with an asterisk in the table correspond to "singleton" levels k, i.e., all of their nodes have a single outgoing edge. This is because x_k is the lowest variable in the support of some constraint d_j, thus d_j is satisfied only when x_k assumes a specific value, given that all the other variables in $Supp(d_j)$ have already been assigned. These singleton levels correspond to the leading positions λ_j in the footprint $\Phi(\mathbf{A}_f)$, thus the number of singleton levels (but not their position) in the MDD is the same, $rank(\mathbf{A})$, for any variable order. These observations hold because $\mathbf{A}_f$ is in row footprint form, even if this form is not unique. Any other footprint form matrix would lead to an MDD that is isomorphic to that of Fig. 3, i.e., with the same nodes and the same shape of the **d** vectors. For instance, had we transformed (3) into the *reduced* footprint form by adding row 3 to row 2, the integer entries of the **d** vectors would be different, but the size of these vectors and the position of their "⇓" and "⇑" entries would be the same, and so would the $\alpha(k)$ values.

5.4. A heuristic based on the active constraints in the footprint

Based on the observations in the previous section, we can investigate heuristic scores based on estimating the number of combinations of partial sums at each level, as a predictor of the number of MDD nodes required at that level. The recently proposed i_{rank} score [**?**] focuses on the number of *linearly-independent active constraints* at each level, and can be expressed as

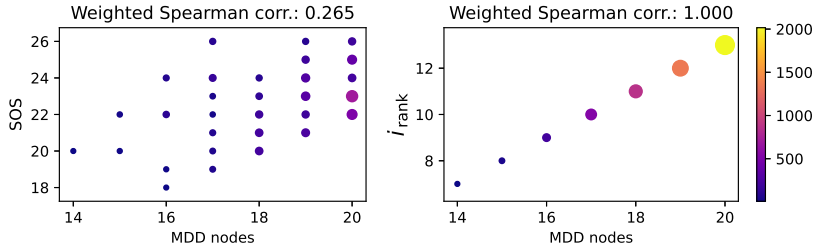
$$i_{\text{rank}} \;=\; \sum_{k=1}^{L} \alpha(k) \;=\; \sum_{j=1}^{r} (\tau_j - \lambda_j),$$

i.e., i_{rank} can be viewed as the area delimited by the leading positions (included) and trailing positions (excluded) in the $\mathbf{A}_f$ matrix. This score has the potential to be more accurate than the SOS score, since it takes into account the linear dependence between the constraints, while SOS does not.

Intuitively, a larger value of i_{rank} likely corresponds to more possible combinations of partial sums at each level, thus potentially results in more MDD nodes. We can thus use a heuristic search (again, simulated annealing [**?**]) to find a variable order that minimizes, or at least reduces, the i_{rank} score. Of course, a more detailed analysis would also consider the actual range of variables and the effect of entries with magnitude greater than 1 in $\mathbf{A}_f$, which may further restrict the possible combinations of partial sums since all variables must have integer values, but we have not yet found a way to incorporate such additional considerations efficiently. Despite these simplifications, the experimental results below show that, our heuristic based on just the dimensionality of the active constraints at each level tends to work very well.

Figure 4 compares the SOS and i_{rank} scores for all $7! = 5040$ possible orders of the seven variables for the set of linear constraints in (2). Each dot corresponds to the subset of the 7! orders resulting in a given number of MDD nodes (x-axis) and a given score (y-axis), while the size and color of the dot indicates the size of that subset. Correlation is computed using the weighted Spearman coefficient [**?**], where each sample is the unique pair (*score, nodes*), with a weight equal to the number of times that pair occurs in the 7! orders. Results are shown for $B = 1$ in Fig. 4 (A) and $B = 8$ in Fig. 4 (B). When $B = 1$, i_{rank} exhibits a perfect correlation between score and number of MDD nodes, while SOS suffers from some confusion, particularly for orders resulting in larger MDDs. For both $B = 1$ and $B = 8$, if a heuristic search succeeds in finding an order that achieves the minimum i_{rank} score of 7, then that order does in fact minimize the number of MDD nodes; furthermore, since orders with lower i_{rank} score never correspond in this example to MDDs with greater number of nodes, even if the heuristic search might be interrupted before finding an optimal order (as it would likely happen in large problem instances), investing more effort in the search never results in a worse order. This is instead not the case for SOS, even when $B = 1$, since the smallest MDD size, 14 nodes, is obtained for *some*, but not all, of the orders corresponding to a SOS score of 20, while the optimal SOS score of 18 results in a larger MDD size, 16 nodes (in other words, investing

(A) Correlation between score and MDD node count, when $B = 1$.



(B) Correlation between score and MDD node count, when $B = 8$.

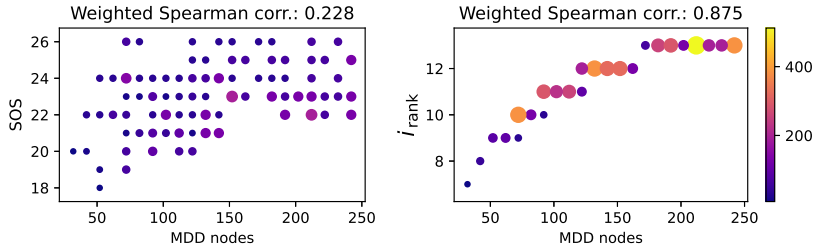


Figure 4: Comparison between SOS and i_{rank} on example (2).

a larger effort to further reduce the SOS score may actually result in a worst order, in this example).

Next, consider a generalization of the set of linear constraints (2), for $B = 8$:

$$\begin{array}{cccccccccc} a_1{\cdot}x_1 & & & + a_4{\cdot}x_4 & & & + a_7{\cdot}x_7 & = & 8 & (d_1'') \\ a_1{\cdot}x_1 & & & & + a_5{\cdot}x_5 & + a_6{\cdot}x_6 & + a_7{\cdot}x_7 & = & 8 & (d_2'') \\ & a_2{\cdot}x_2 & + a_3{\cdot}x_3 & + a_4{\cdot}x_4 & & & + a_7{\cdot}x_7 & = & 8 & (d_3'') \\ & a_2{\cdot}x_2 & + a_3{\cdot}x_3 & & + a_5{\cdot}x_5 & + a_6{\cdot}x_6 & + a_7{\cdot}x_7 & = & 8 & (d_4'') \end{array} \tag{4}$$

where $a_1, ..., a_7 \in \{1, 2\}$, resulting in $2^7 = 128$ possible variants, or $a_1, ..., a_7 \in \{1, 8\}$, also resulting in 2^7 (different) variants. We repeated the experiments of Figure 4, considering all 7! orders for each variant of system (4).

Figure 5 shows the correlation between SOS or i_{rank} score and number of MDD nodes for each of the 128 variants of each of the two cases, computed using each of the 7! orders. Figure 5(A), corresponding to the case when the coefficients can be either 1 or 2, shows that *all* orders with minimum i_{rank} score are optimal (in terms of MDD nodes) in 92 of the 128 variants, while *some* orders with minimum i_{rank} score are optimal in the remaining 36 variants. In particular, i_{rank} exhibits the highest correlation, 0.99, in 49 of the variants. The SOS score has instead much worse performance, as it not only exhibits quite low correlation, but *no* order with minimum SOS score is optimal in any of the variants. Figure 5(B), corresponding to the somewhat more challenging case when the coefficients can be either 1 or 8, still shows much better performance for i_{rank} over SOS. Among the 128 variants, *all* orders with minimum i_{rank} score

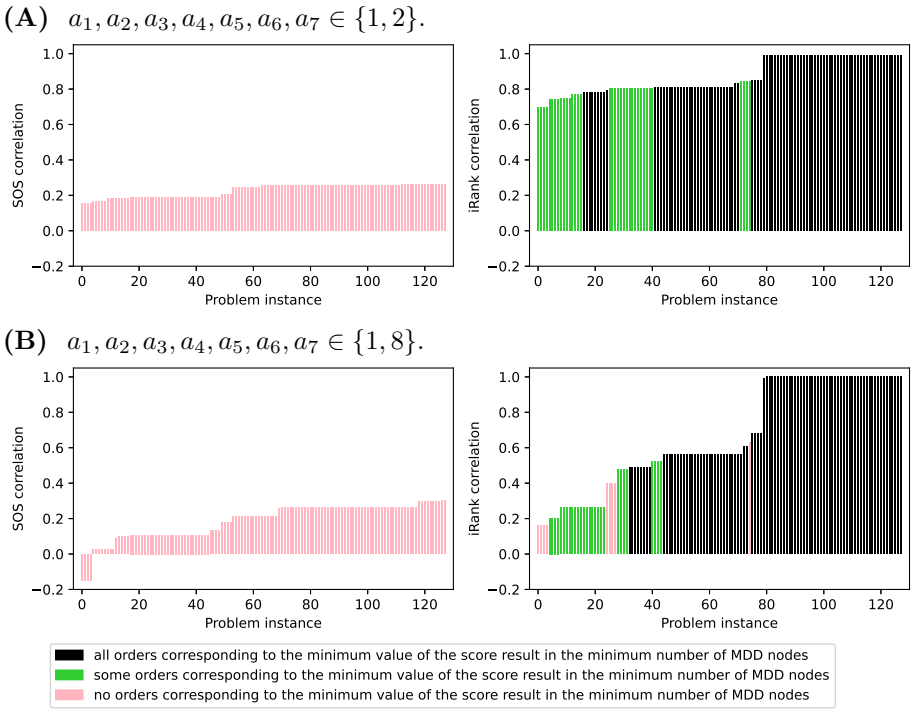


Figure 5: Correlations between score and MDD size for SOS and i_{rank}.

are optimal in 91 variants, *some* orders with minimum i_{rank} score are optimal in 28 variants, and *no* orders with minimum i_{rank} score are optimal in only 9 variants. SOS has again quite low correlation (even negative, -0.149, in 4 of the variants), and *no* order with minimum SOS score is optimal in any of the variants.

Figure 6 shows detailed results for some of the variants used to compute the plots of Figure 5.

- Figure 6(A) reports the variant $a_1 = a_2 = a_3 = a_4 = a_5 = 1$, and $a_6 = a_7 = 2$, one with the worst SOS correlation, 0.155 (and the second worst i_{rank} correlation, 0.741) when the coefficients are either 1 or 2. While only some of the orders with minimum i_{rank} score of 7 are optimal, all of them are better than any of the other orders.

- Figure 6(B) reports the variant $a_1 = a_2 = a_4 = a_5 = a_7 = 1$, and $a_3 = a_6 = 2$, one with the worst i_{rank} correlation, 0.7 (and the fourth worst SOS correlation, 0.184) when the coefficients are either 1 or 2. Here, also, all orders with minimum i_{rank} score of 7 are better than any of the other orders.

- Figure 6(C) reports the variant $a_1 = \cdots = a_6 = 1$ and $a_7 = 8$, where i_{rank}

still achieves a relatively high correlation of 0.629, while the correlation for SOS is barely positive. In this variant, all orders corresponding to the minimum i_{rank} score of 7 are very good (resulting in 56 nodes), but not optimal (the minimum number of nodes, 51, is achieved only by *some*, but not all, of the orders with a non-minimum i_{rank} score of 9).

- Figure 6(D) reports one of the most challenging variants, $a_1 = a_3 = a_4 = a_6 = 1$ and $a_2 = a_5 = a_7 = 8$, where i_{rank} has a correlation of only 0.16, while SOS has *negative* correlation. Now, orders with the minimum i_{rank} score of 7 may result in 42, 49, or 56 nodes, where 42 nodes could be considered "good", but still not as good as the 41 nodes achieved by *some* of the orders with the *worst* i_{rank} score of 13 (although *all* of the worst orders, resulting in 62 nodes, also have this i_{rank} score of 13). The best results, 37 nodes, are achieved by *some* of the orders with an i_{rank} score of 9, although some other orders, also with an i_{rank} score of 9, result in 58 nodes, which is more than the number of nodes achieved by *any* of the orders with the minimum i_{rank} score of 7.

We stress that the exhaustive results we report could be obtained in practice only because the constraint solving problem instance we chose has few variables and few constraints. In real applications with hundreds or even thousands of variables and constraints, one cannot (and would not desire to) explore the effect of all orders on the size of the MDD encoding all the solutions, but would instead build the MDD for only one specific variable order, arguably the one suggested by our i_{rank} score. Furthermore, for large problems, even just finding an order minimizing the i_{rank} score might not be feasible, thus an order with a "small" value for the i_{rank} score will often have to do. For examples of large applications of the i_{rank} score, see [**?**]

6. Conclusion

We defined the *footprint form* of a matrix, which, unlike the well-known echelon form, considers the positions of both the leading and trailing entries in each row. We proved that the footprint form provides an immediate way to compute the rank of certain sub-matrices, and that it can be made canonical by imposing additional restrictions.

Our work was motivated by a practical application: finding a proxy for the number of nodes in the multivalued decision diagram (MDD) encoding the set of solutions to a system of integer linear constraints, with the goal of finding a variable order that minimizes the size of this MDD. We then exhaustively evaluated the resulting i_{rank} score on small problem instances, confirming that it exhibits a strong correlation to the MDD size, thus it can be effectively employed in practice.

In the future, we plan to investigate efficient ways to compute an upper bound (rather than just a proxy) for the number of MDD nodes corresponding to a given variable order, taking into account not just the positions, but the

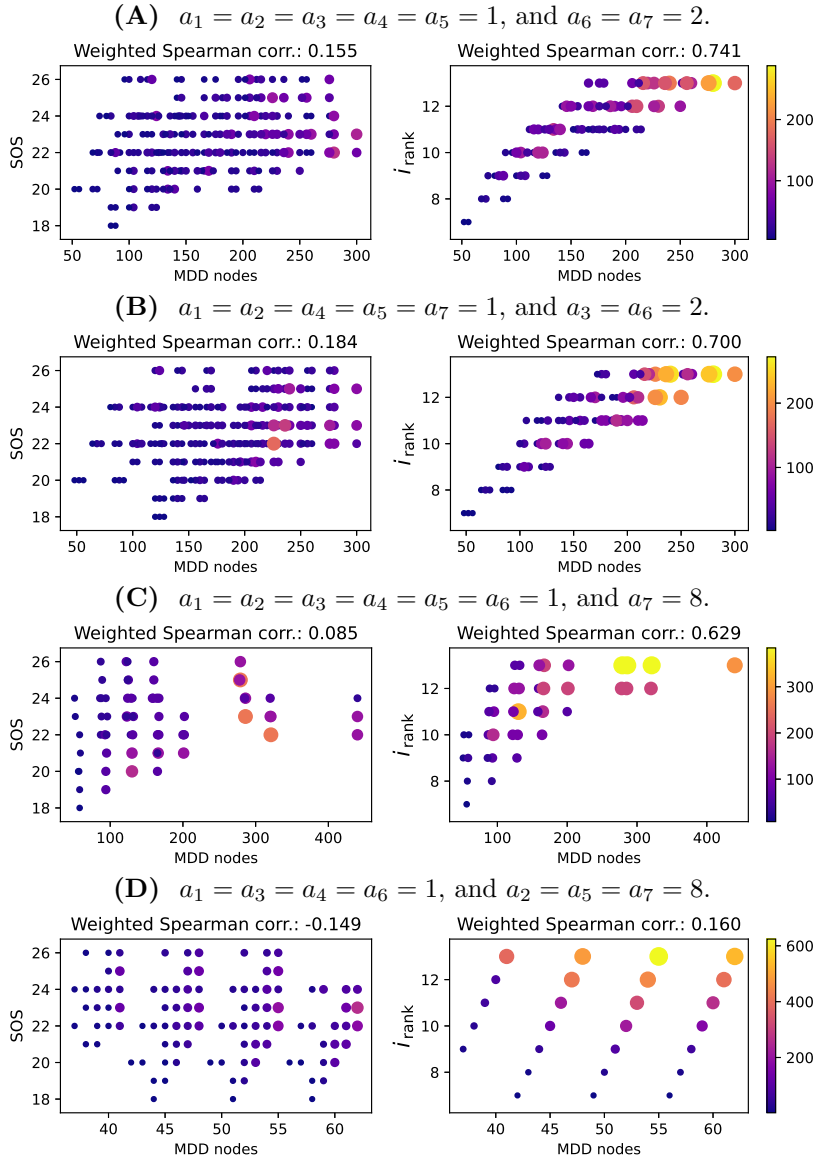


Figure 6: Comparison between SOS and i_{rank} on example (4), non-unitary weights.

actual values of the nonzero coefficients in the linear system, and possibly even derive special conditions under which we can efficiently obtain an exact count of the number of MDD nodes, of course without having to build the actual MDD.

Acknowledgments

We are grateful to Lorenzo Ciardo and Leslie Hogben for their feedback and help on a preliminary version of this manuscript.