1. Problem Setup

Let

- $\mathcal{V}$ be the set of variable indices, and
- $\mathcal{C}$ be the set of constraint (row) indices

of a MILP defined by

$$\begin{aligned} \min \quad & c^T x, \\ \text{s.t.} \quad & Ax \leq b, \\ & \ell \leq x \leq u, \\ & x_j \in \mathbb{Z} \ (\text{or } \{0,1\}), \end{aligned}$$

where the structure of A, the objective coefficients, and the bounds provide both numerical and structural information about the problem.

2. Hierarchical Partitioning Using Block Rules

The ordering is determined by recursively partitioning the full index sets into smaller blocks. Two sets of rules are employed at each recursion level:

2.1 Primary (Parent) Block Rules

At each stage, a list of *primary* rules is applied in sequence. Each rule r in this list maps every index $v \in \mathcal{V}$ (or $i \in \mathcal{C}$) to a label $r(v)$ (or $r(i)$). In effect, the rule defines a partition of the index set:

$$\mathcal{V} = \bigcup_{l \in \mathcal{L}_r} \mathcal{V}_l, \quad \text{with} \quad \mathcal{V}_l = \{v \in \mathcal{V} : r(v) = l\},$$

and similarly for constraints. A rule is deemed *effective* on the current block if it partitions the set into more than one block (i.e., if $|\mathcal{L}_r| > 1$).

2.2 Secondary (Child) Block Rules

If none of the primary rules yield an effective partition at the current level (i.e., all primary rules return a single block), then a second set of *child* rules is applied. These rules again attempt to partition the index set based on alternative criteria. When a child rule is effective (i.e., splits the indices into at least two blocks), the procedure uses the resulting partition.

2.3 Recursion

Suppose that at a recursion level k the current block is given by subsets of variable indices $V_k \subseteq \mathcal{V}$ and constraint indices $C_k \subseteq \mathcal{C}$. The procedure is as follows:

1. **Apply Primary Rules:**
   For each rule r in the primary list, compute the partition

   $$V_k = \bigcup_{l \in \mathcal{L}_r} V_{k,l}.$$

   If any r yields more than one block, then each block $V_{k,l}$ (and similarly for C_k) is processed recursively by reapplying the full set of primary rules at the next level.
2. **Fallback to Child Rules:**
   If no primary rule partitions the current block, then the child rules are applied in order. If one of these rules partitions the block into multiple subsets, each subset is processed recursively. Furthermore, the order of child rules is rotated between recursive calls to avoid a potential bias from repeatedly applying the same rule.
3. **Termination:**
   The recursion stops when either the maximum allowed recursion depth is reached or when neither the primary nor the child rules partition the current block further (i.e., every rule returns a single block for the indices in the block).

3. Intra-Block Ordering

When recursion terminates for a block (either by reaching the maximum depth or by failing to partition further), the indices within that block are ordered using intra-block rules. Each index v (or i) is assigned a tuple of scores based on a set of intra rules:

$$\mathbf{s}(v) = \left(s_1(v), s_2(v), \ldots, s_k(v)\right).$$

Then the final order within the block is obtained by performing a lexicographical sort over these score tuples. Mathematically, if $v_1, v_2, \ldots, v_n$ are the indices in the block, they are ordered so that

$$\mathbf{s}(v_{(1)}) \leq_{\text{lex}} \mathbf{s}(v_{(2)}) \leq_{\text{lex}} \cdots \leq_{\text{lex}} \mathbf{s}(v_{(n)}).$$

4. Final Ordering

The final ordering for the entire problem is constructed by concatenating the ordered indices from each block, preserving the partitioning hierarchy. Denote by

$$\mathrm{Sort}_{\mathrm{Recursive}}(V)$$

the ordering function defined recursively by

$$\mathrm{Sort}_{\mathrm{Recursive}}(V) = \begin{cases} \bigcup_{l \in \mathcal{L}} \mathrm{Sort}_{\mathrm{Recursive}}(V_l), & \text{if } V \text{ is partitioned into blocks } \{V_l\}_{l \in \mathcal{L}} \text{ by an effective rule,} \\ \mathrm{LexSort}\big(V, \mathbf{s}(v)\big), & \text{otherwise,} \end{cases}$$

with a corresponding definition for constraints.

Thus, the overall ordering is given by the concatenation of the orderings produced at each recursive step, ensuring that the final arrangement respects both the coarse partitioning (by structural rules) and the refined ordering (by intra rules).

5. Summary

The recursive hierarchical rule composition procedure for MILP ordering can be summarized mathematically as follows:

- **Initialization:**
  Begin with the full sets $\mathcal{V}$ and $\mathcal{C}$.
- **Recursive Partitioning:**
  At each level, attempt to partition the current index block using a prescribed sequence of primary rules. If these fail to partition, try a secondary (child) rule set. For any effective partition (i.e., more than one block), process each sub-block recursively.
- **Termination and Intra-Block Ordering:**
  When no rule yields further partitioning (or a maximum recursion depth is reached), assign each index a score tuple via intra rules and determine the order by lexicographical sorting.
- **Final Output:**
  The complete ordering is the concatenation of the recursively determined orderings of all blocks.

This approach guarantees a stable and deterministic ordering that captures both the high-level structure (via block rules) and fine details (via intra rules) of the MILP formulation.