Scoring Rules (Multi-Block Version)

Instead of relying on a single large multiplier (e.g., 10^6) for "type" or "sense" alone, the system now uses **two or more block rules** that each produce a partial **score component** (or label). All of these partial labels form the **first part** of a tuple, which defines the **major block** into which each variable or constraint belongs. Then, all **intra-block** rule scores are combined into a **single numeric value** appended as the **last element** of the tuple. The final **lexicographic sort** of these tuples enforces a strict hierarchy:

1. **Compare block components** (in order)
2. If two items have the **same** block components, compare their **intra-block sum** (the last tuple element)

This yields a clean, multi-dimensional grouping while preserving the advantage of a lexicographic approach—no gigantic multipliers are needed.

Column (Variable) Ordering

Block Rules (for Variables)

Here are two **example** block rules (though you might have more):

1. **Type-Based Rule**
   - Distinguishes each variable as binary (3), integer (2), continuous (1), and possibly detects other categories (e.g., semicontinuous, effectively binary if $[a, a+1]$, etc.).
   - Produces a numeric label, e.g. 3, 2, or 1, depending on the variable's integrality status.
2. **Bound-Based Rule**
   - Looks at the domain of each variable:
     - Both bounds finite, nonnegative => 4
     - Both bounds finite, straddling zero => 3
     - One infinite bound => 2
     - Both bounds infinite => 1
   - Produces another numeric label (e.g., 4, 3, 2, or 1).

An individual variable might thus receive a **two-component** block label (e.g., $(3, 4)$ for a **binary** variable with **nonnegative finite** bounds).

Intra-Block Rules (for Variables)

After you place variables into these block categories, you still want to refine ordering **within** each block. For instance:

1. **Column Coefficients**: $\sum_j \log\bigl(1 + |\delta_{ij}|\bigr)$ measures how "large" or "frequent" variable i is across constraints.
2. **Objective Coefficient**: $\log\bigl(1 + |\mathrm{Obj}_i|\bigr)$ (often scaled by 100 to emphasize variables with high objective impact).
3. **Occurrences**: count of non-zero entries in the constraint matrix for variable i.

These individual scores can be **summed** into a **single** "intra-block" metric. For example:

$$\mathrm{IntraBlockScore}_i = \Bigl[\sum_j \log(1 + |\delta_{ij}|)\Bigr] \;+\; 100\,\log\bigl(1 + |\mathrm{Obj}_i|\bigr) \;+\; [\#\mathrm{occurrences}].$$

Final Tuple for Each Variable

Each variable i ends up with a tuple:

$$\underbrace{(\mathrm{BlockLabel1},\ \mathrm{BlockLabel2},\ \ldots)}_{\text{Multiple block-rule outputs}},\ \underbrace{\mathrm{IntraBlockSum}}_{\text{last element}}.$$

- **BlockLabel1** might be from the type-based rule (e.g. 3 for binary).
- **BlockLabel2** might be from the bound-based rule (e.g. 4 for nonnegative).
- **IntraBlockSum** is a single numeric total from the "coefficient sum," "objective," and "occurrences."

A **descending lexicographic** sort on these tuples means:

1. All variables with a higher **type-based** label come first.
2. If there's a tie, we compare the **bound-based** label.
3. If still tied, we compare the **intra-block** total—highest sum ranks first.

Row (Constraint) Ordering

Block Rules (for Constraints)

Similarly, constraints can be partitioned along multiple dimensions. Two examples:

1. **Sense-Based Rule**
   - Classifies each constraint by sense: "$\geq$" => 3, "$=$" => 2, "$\leq$" => 1.
2. **Composition-Based Rule**
   - Checks if the constraint contains **only** integral/binary variables => 3,
   - **Only** continuous => 2,
   - A **mix** => 1.

Hence, a constraint might have a **two-part** block label $(3, 2)$ if it is a "$\geq$" constraint that contains **only continuous** variables, for example.

Intra-Block Rules (for Constraints)

Within each constraint block, you can order them by:

1. **Row Coefficients**: $\sum_i \log\bigl(1 + |\gamma_{ji}|\bigr)$.
2. **RHS**: $\log(1 + |\mathrm{RHS}_j|)$ (scaled by 100).
3. **Range**: $\log\bigl(1 + \mathrm{Range}_j\bigr)$ for ranged constraints.

Sum these into a single number:

$$\mathrm{IntraBlockScore}_j = \sum_i \log(1 + |\gamma_{ji}|) \;+\; 100\,\log(1 + |\mathrm{RHS}_j|) \;+\; \log\bigl(1 + \mathrm{Range}_j\bigr).$$

Final Tuple for Each Constraint

$$(\mathrm{SenseLabel},\ \mathrm{CompositionLabel},\ \ldots,\ \mathrm{IntraBlockSum}).$$

Sorted **descending lexicographically**:

1. Compare sense label first.
2. If there's a tie, compare composition label.
3. If still tied, compare the intra-block sum.

Why Tuples with the Last Element as Intra-Score?

1. **Absolute Hierarchy**: Each block rule label is a separate **position** in the tuple. If any item differs in an earlier position, it outranks or is outranked, regardless of the subsequent positions.
2. **Simple Summation**: By combining the multiple "intra-block" factors (e.g., objective, coefficient sums) into **one** numeric total, you get a **single** final comparison for tie-breaking.

3. **No Need for Large Multipliers**: You don't have to multiply each block label by a huge constant. The **lexicographic** ordering ensures earlier positions in the tuple always dominate over later ones.

Example Ordering Flow

1. **Variables**:
   - Variable x_3 has block label $(3, 4) \rightarrow$ (binary, nonnegative).
   - Variable x_7 has block label $(2, 4) \rightarrow$ (integer, nonnegative).
   - Their **intra-block** scores might be 150 vs. 300.

   In lexicographic order, $(3, 4, *)$ (i.e., binary + nonnegative) ranks above $(2, 4, *)$, so x_3 is placed before x_7 **regardless** of their 150 vs. 300. If two variables share the same block label, then the one with the **higher** intra-score (e.g., 300) comes first.
2. **Constraints**:
   - Constraint c_5 might have sense label = 3 ("$\geq$") and composition label = 3 (only integer variables), plus an intra score of 200.
   - Constraint c_2 might have sense label = 3 ("$\geq$") and composition label = 2 (only continuous variables), plus an intra score of 500.

   Even though c_2 has a bigger **intra** score (500 vs. 200), it has composition label 2 vs. 3 $\rightarrow$ $(3, 3, 200)$ outranks $(3, 2, 500)$ in descending lexicographic order.

Conclusion

By creating **tuples** whose **first components** come from **block classifications** (type vs. bound, sense vs. composition, etc.) and whose **last component** is a **single intra-block sum**, you get a **strict multi-level** ordering:

1. **Major** separation by block rule results (type, sense, etc.).
2. **Minor** sorting within each block by the combined "intra-score."

This ensures an intuitive, transparent hierarchy without juggling large numeric multipliers. The final ordering is simply a **descending lexicographic** comparison of tuples—**blocking** first, **intra** tie-breaks last.