Step-by-Step Process to Achieve a Unique Canonical Form for MILO Problems

1. Normalization for Scaling Invariance

- **Objective**: Remove scaling effects (multiplying rows/columns by scalars).
- **Steps**:
  - **Column Normalization**: For each variable (column), divide coefficients by the column's L2 norm. Adjust the objective coefficient $c_i \leftarrow c_i \cdot \|a_i\|_2$.
  - **Row Normalization**: For each constraint (row), divide coefficients by the row's L2 norm. Adjust the RHS $b_j \leftarrow b_j / \|a_j\|_2$.

2. Variable Scoring

- **Objective**: Prioritize variables based on type, sparsity, and coefficient patterns.
- **Criteria**:
  i. **Type**: Integer variables receive a fixed bonus (e.g., +100) over continuous variables.
  ii. **Sparsity**: Variables appearing in more constraints receive +1 per constraint.
  iii. **Coefficient Hash**:
     - Normalize and sort column coefficients.
     - Generate a hash (e.g., fixed-precision string or polynomial hash) of the sorted coefficients.
- **Example**:
  - x_i (integer, appears in 3 constraints, hash `H1`) → Score = $100 + 3 + \mathrm{H1}$.

3. Constraint Scoring

- **Objective**: Prioritize constraints based on RHS, integer variables, and coefficient patterns.
- **Criteria**:
  i. **RHS Type**: Constraints with $b_j \neq 0$ receive a fixed bonus (e.g., +50).
  ii. **Integer Variables**: +1 per integer variable in the constraint.
  iii. **Coefficient Hash**:
     - Normalize and sort row coefficients **by the sorted variable order**.
     - Generate a hash of the sorted coefficients.
- **Example**:
  - C_j ($b_j \neq 0$, 2 integer variables, hash `H2`) → Score = $50 + 2 + \mathrm{H2}$.

4. Sorting Variables and Constraints

- **Objective**: Fix the order of variables/constraints deterministically.
- **Steps**:
  i. **Sort Variables**: Descending by score → hash → type (integer > continuous).
  ii. **Sort Constraints**: Descending by score → hash → RHS type ($b_j \neq 0$ first).

5. Recursive Block Decomposition

- **Objective**: Exploit sparsity by grouping interacting variables/constraints into blocks.
- **Steps**:
  i. **Build a Bipartite Graph**: Nodes = variables/constraints; edges = non-zero coefficients.
  ii. **Detect Communities**: Use clustering (e.g., Louvain algorithm) to identify dense subgraphs.
  iii. **Recursively Split**:
    - Process each block independently.
    - Repeat scoring/sorting within blocks until no further subdivision is possible.

6. Hash Function Implementation

- **Objective**: Ensure identical coefficient patterns produce identical hashes.
- **Steps**:
  i. **Normalize and Sort**:
    - For variables: Sort normalized column coefficients (ignore constraint order).
    - For constraints: Sort normalized row coefficients **by the sorted variable order**.
  ii. **Generate Hash**:
    - Convert sorted coefficients into a fixed-precision string (e.g., `"0.500,0.707"`).
    - Use a polynomial rolling hash or cryptographic hash (truncated for efficiency).

7. Handling Ties

- **Objective**: Ensure deterministic outcomes when scores/hashes collide.
- **Tiebreakers**:
  - For variables: Compare objective coefficients c_i.
  - For constraints: Compare RHS values b_j.

8. Validation

- **Objective**: Confirm canonical form invariance.
- **Steps**:
  i. **Generate Permuted/Scaling Variants**: Create multiple instances of the same problem with shuffled rows/columns or scaled coefficients.

ii. **Apply Canonicalization**: Ensure all variants produce the same variable/constraint order and block structure.
iii. **Solver Testing**: Measure solver runtime variability across canonicalized instances.

Example Workflow

1. **Original Problem**:

$$\begin{aligned} \min \quad & 4x_1 + 5x_2 + 2x_3 + 3x_4 \\ \text{s.t.} \quad & 2x_1 + 3x_3 \leq 6 \\ & x_2 - x_4 \leq 0 \\ & x_1 + x_2 + x_3 + x_4 \leq 10 \\ & x_1, x_2 \in \mathbb{Z}, \quad x_3, x_4 \in \mathbb{R}. \end{aligned}$$

2. **Permuted Problem**:
   - Variables: $[x_4, x_2, x_1, x_3]$, Constraints: Reordered as $[C3, C1, C2]$.
3. **Canonicalization Steps**:
   - **Normalization**: Scale columns/rows to unit L2 norm.
   - **Scoring**: Assign scores to variables/constraints.
   - **Sorting**: Order variables as x_1, x_2, x_3, x_4; constraints as $C3, C1, C2$.
   - **Block Decomposition**: Split into subblocks (e.g., integer vs. continuous variables).
   - **Output**: Unique canonical form identical to the original problem's sorted version.

Step-by-Step Application to the Example Problem

Original Problem:

$$\begin{aligned} \min \quad & 4x_1 + 5x_2 + 2x_3 + 3x_4 \\ \text{s.t.} \quad & 2x_1 + 3x_3 \leq 6 \quad \text{(C1)} \\ & x_2 - x_4 \leq 0 \quad \text{(C2)} \\ & x_1 + x_2 + x_3 + x_4 \leq 10 \quad \text{(C3)} \\ & x_1, x_2 \in \mathbb{Z}, \quad x_3, x_4 \in \mathbb{R}. \end{aligned}$$

Permuted Problem:

- **Variables**: $[x_4, x_2, x_1, x_3]$.
- **Constraints**: Reordered as $[C3, C1, C2]$.

- **Objective**: $\min \quad 3x_4 + 5x_2 + 4x_1 + 2x_3.$

Step 1: Normalization for Scaling Invariance

Column Normalization:

Variable	Original Column	L2 Norm	Normalized Column	Adjusted Objective
x_1	$[2, 0, 1]$	$\sqrt{2^2 + 0^2 + 1^2} = \sqrt{5}$	$[2/\sqrt{5}, 0/\sqrt{5}, 1/\sqrt{5}]$	$4 \cdot \sqrt{5} \approx 8.944$
x_2	$[0, 1, 1]$	$\sqrt{0^2 + 1^2 + 1^2} = \sqrt{2}$	$[0/\sqrt{2}, 1/\sqrt{2}, 1/\sqrt{2}]$	$5 \cdot \sqrt{2} \approx 7.071$
x_3	$[3, 0, 1]$	$\sqrt{3^2 + 0^2 + 1^2} = \sqrt{10}$	$[3/\sqrt{10}, 0/\sqrt{10}, 1/\sqrt{10}]$	$2 \cdot \sqrt{10} \approx 6.325$
x_4	$[0, -1, 1]$	$\sqrt{0^2 + (-1)^2 + 1^2} = \sqrt{2}$	$[0/\sqrt{2}, -1/\sqrt{2}, 1/\sqrt{2}]$	$3 \cdot \sqrt{2} \approx 4.242$

Row Normalization:

Constraint	Original Row	L2 Norm	Normalized Row	Adjusted RHS
C1	$[2, 0, 3, 0]$	$\sqrt{2^2 + 3^2} = \sqrt{13}$	$[2/\sqrt{13}, 0, 3/\sqrt{13}, 0]$	$6/\sqrt{13} \approx 1.664$
C2	$[0, 1, 0, -1]$	$\sqrt{1^2 + (-1)^2} = \sqrt{2}$	$[0, 1/\sqrt{2}, 0, -1/\sqrt{2}]$	$0/\sqrt{2} = 0$
C3	$[1, 1, 1, 1]$	$\sqrt{1^2 + 1^2 + 1^2 + 1^2} = 2$	$[0.5, 0.5, 0.5, 0.5]$	$10/2 = 5$

Step 2: Variable Scoring

Variable	Type	# Constraints	Sorted Normalized Coefficients	Hash	Score
x_1	Integer	2	$[0, 0.447, 0.894]$	H1	$100 + 2 + \mathrm{H1} = 102.8$
x_2	Integer	2	$[0, 0.707, 0.707]$	H2	$100 + 2 + \mathrm{H2} = 102.6$
x_3	Continuous	2	$[0, 0.316, 0.949]$	H3	$0 + 2 + \mathrm{H3} = 2.3$
x_4	Continuous	2	$[-0.707, 0, 0.707]$	H4	$0 + 2 + \mathrm{H4} = 2.1$

Sorted Variables: x_1, x_2, x_3, x_4.

Step 3: Constraint Scoring

Constraint	RHS ≠ 0	# Integer Vars	Sorted Coefficients (by Variable Order)	Hash	Score
C3	Yes	2	$[0.5, 0.5, 0.5, 0.5]$	H7	$50 + 2 + \mathrm{H7} = 52.9$
C1	Yes	1	$[0.555, 0, 0.832, 0]$	H5	$50 + 1 + \mathrm{H5} = 51.5$
C2	No	1	$[0, 0.707, 0, -0.707]$	H6	$0 + 1 + \mathrm{H6} = 1.6$

Sorted Constraints: C3, C1, C2.

Step 4: Sorting Variables and Constraints

After Sorting:

- **Variables**: x_1, x_2, x_3, x_4.
- **Constraints**: C3, C1, C2.

Canonical Form:

$$
\begin{aligned}
\min \quad & 8.944x_1 + 7.071x_2 + 6.325x_3 + 4.242x_4 \\
\text{s.t.} \quad & 0.224x_1 + 0.354x_2 + 0.158x_3 + 0.354x_4 \leq 5 \quad (\text{C3}) \\
& 0.277x_1 + 0.277x_3 \leq 1.664 \quad (\text{C1}) \\
& 0.5x_2 - 0.5x_4 \leq 0 \quad (\text{C2}) \\
& x_1, x_2 \in \mathbb{Z}, \quad x_3, x_4 \in \mathbb{R}.
\end{aligned}
$$

Limitations of L_2 Normalization

Using L_2 normalization to achieve a canonical form for MILO problems presents two significant issues. First, applying L_2 normalization involves scaling each column by its Euclidean norm, which typically yields non-integer factors. This poses a problem for integer or binary variables since multiplying them by non-integer scaling factors disrupts their discrete nature—integer variables must remain integer, and binary variables must remain in $\{0, 1\}$. Second, if the same underlying problem is subjected to different arbitrary scalings, the computed L_2 norms will differ, and consequently, the normalization procedure will produce numerically distinct canonical forms. In other words, two mathematically equivalent formulations may normalize to different forms, defeating the purpose of achieving a unique canonical representation.