Experiment Outline

Table of Contents

1. Problem Initialization

The experiment begins with loading an optimization problem and preparing it for analysis.

Steps:

1. **Load the Problem Instance**
   - Retrieve the problem from a specified source.
   - Validate the structure and log relevant details, such as the number of variables, constraints, and the objective function type.
2. **Solve the Original Problem**
   - Optimize the problem in its original form.
   - Record solve time, solution status, and objective value.
3. **Generate the Canonical Form of the Original Problem**
   - Transform the problem into a standardized representation.
   - Solve the canonical form and store its corresponding metrics.

2. Iterative Analysis with Permutations

The core experiment consists of multiple iterations, where variations of the original problem are analyzed.

Steps for Each Iteration:

1. **Create a Permuted Version of the Problem**
   - Apply a reordering of variables and constraints.
   - Log the transformation details.
2. **Measure Structural Difference Before Canonicalization**
   - Compute a metric quantifying the difference between the original and permuted formulations.
   - Record this **permutation distance**.
3. **Solve the Permuted Problem**
   - Optimize the problem after permutation.
   - Store solve time and objective value.
4. **Generate and Solve the Canonical Form of the Permuted Problem**
   - Apply the canonical transformation to the permuted problem.
   - Solve the transformed problem and record the results.
5. **Analyze the Relationship Between Permutation and Canonicalization**
   - Compare the canonical representation of the permuted problem to that of the original problem.
   - Compute the **adjusted permutation distance** after canonicalization.
6. **Log Iteration Results**
   - Validate equivalence between formulations.
   - Compare variable and constraint counts.
   - Compare objective values.
   - Assess solve time differences.
   - Evaluate structural differences before and after canonicalization.

These steps are repeated for multiple iterations to capture variability in behavior.

3. Variability Analysis

After all iterations are complete, statistical analysis is performed to quantify the impact of permutations and canonicalization.

Steps:

1. **Compute Solve-Time Variability**
   - Measure the **dispersion of solve times** across permutations.
   - Compare the variability of **raw permutations** vs. **canonical forms**.
   - Determine whether canonicalization improves solve-time consistency.
2. **Compute Permutation Distance Variability**
   - Measure the **spread of permutation distances** across iterations.
   - Compare **pre-canonicalization** vs. **post-canonicalization** variability.
   - Assess whether canonicalization consistently reduces structural differences.

4. Conclusion

- Summarize the impact of permutations on problem structure and solver performance.
- Evaluate the effectiveness of canonicalization in reducing variability.
- Identify potential improvements for ensuring stability and efficiency in future analyses.

5. Mathematical Summary of Permutation and Distance Computation**

1. Problem Representation and Permutation

A **Mixed-Integer Programming (MIP)** problem is defined as:

$$Ax \leq b, \quad x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}$$

where:

- $A \in \mathbb{R}^{m \times n}$ is the constraint matrix,
- $x \in \mathbb{R}^n$ is the decision variable vector,
- $b \in \mathbb{R}^m$ is the right-hand side vector.

We apply **two independent permutations**:

- A **row permutation** P_{row}, which reorders the constraints.
- A **column permutation** P_{col}, which reorders the variables.

The **permuted problem** becomes:

$$P_{\text{row}} A P_{\text{col}} x \leq P_{\text{row}} b$$

where:

- $P_{\text{row}} \in \mathbb{R}^{m \times m}$ is a **permutation matrix** representing the reordering of constraints.
- $P_{\text{col}} \in \mathbb{R}^{n \times n}$ is a **permutation matrix** representing the reordering of variables.

Each permutation matrix is **orthogonal**:

$$PP^\top = P^\top P = I$$

and contains exactly one "1" in each row and each column.

2. Distance Between Permutations

To quantify the difference between two permutations, we use **separate distance measures** for rows and columns.

2.1 Row and Column Permutations

Given two permutations:

- **Row permutation** π^1_{row} vs. π^2_{row}, each a bijection of $\{1, \ldots, m\}$.
- **Column permutation** π^1_{col} vs. π^2_{col}, each a bijection of $\{1, \ldots, n\}$.

We define distances d_{rows} and d_{cols} separately.

2.2 Hamming Distance

The **Hamming distance** measures how many positions differ:

$$d_{\text{Hamming}}(\pi^1, \pi^2) = \sum_{i=1}^{k} \mathbf{1}(\pi^1(i) \neq \pi^2(i))$$

where $\mathbf{1}(\cdot)$ is the indicator function.

2.3 Kendall Tau Distance

The **Kendall Tau distance** counts the number of pairwise **inversions**:

$$d_{\text{Kendall}}(\pi^1, \pi^2) = \sum_{1 \leq i < j \leq k} \mathbf{1}\Big((\pi^1(i) < \pi^1(j) \text{ and } \pi^2(i) > \pi^2(j)) \text{ or } (\pi^1(i) > \pi^1(j) \text{ and } \pi^2(i) < \pi^2(j))\Big)$$

where an **inversion** occurs if the relative ordering of two elements differs between the two permutations.

This distance is useful when assessing how much a permutation **disrupts order**.

3. Combined Distance Measure

Given **row** and **column** distances, we define an aggregated distance:

$$d_{\text{total}} = \alpha \cdot d_{\text{rows}} + \beta \cdot d_{\text{cols}}$$

where:

- d_{rows} is computed using one of the above distances on $\pi^1_{\text{row}}, \pi^2_{\text{row}}$.
- d_{cols} is computed using one of the above distances on $\pi^1_{\text{col}}, \pi^2_{\text{col}}$.
- α, β are weighting parameters (default: $\alpha = \beta = 1$).

This scalar d_{total} summarizes how much **both rows and columns** are permuted.

6. Variability Metrics in Optimization Experiments

This document explains the mathematical formulation used to measure variability in:

1. **Solve-time variability**: Measures the spread of solve times across permutations.
2. **Permutation distance variability**: Measures the spread of permutation distances before and after canonicalization.

Both metrics use **standard deviation (σ)** to quantify the dispersion of values in their respective sets.

Solve-Time Variability

Solve-time variability captures how much solve times fluctuate across different **permutations of the same problem** and how much **canonicalization** stabilizes this variability.

Mathematical Definition

Let:

- t_{orig} be the solve time of the **original** problem.
- $t_{\text{perm},i}$ be the solve time of the i-th **permuted** problem.
- $t_{\text{canon-orig}}$ be the solve time of the **canonicalized form of the original** problem.
- $t_{\text{canon-perm},i}$ be the solve time of the **canonicalized form of the i-th permuted problem**.
- N be the number of permutations.

The standard deviation of **solve times across permutations (including the original)** is given by:

$$\sigma_{\text{perm}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(t_{\text{perm},i} - \bar{t}_{\text{perm}})^2}$$

where:

$$\bar{t}_{\text{perm}} = \frac{1}{N+1}\left(t_{\text{orig}} + \sum_{i=1}^{N} t_{\text{perm},i}\right)$$

Similarly, the standard deviation of **canonicalized solve times** is:

$$\sigma_{\text{canon}} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (t_{\text{canon-perm},i} - \bar{t}_{\text{canon}})^2}$$

where:

$$\bar{t}_{\text{canon}} = \frac{1}{N+1} \left(t_{\text{canon-orig}} + \sum_{i=1}^{N} t_{\text{canon-perm},i} \right)$$

Interpretation

- **Higher** σ_{perm} → High variability in solve times across permutations.
- **Higher** σ_{canon} → High variability even after canonicalization.
- **If** $\sigma_{\text{canon}} < \sigma_{\text{perm}}$ → Canonicalization improves stability.

Permutation Distance Variability

Permutation distance variability measures how much the **structural differences** (measured by a **permutation distance**) change across permutations.

Mathematical Definition

Let:

- $d_{\text{perm},i}$ be the **permutation distance** between the **original and permuted problem**.
- $d_{\text{canon},i}$ be the **permutation distance** between the **canonical form of the original and the canonical form of the permuted problem**.
- N be the number of permutations.

The standard deviation of **permutation distances before canonicalization** is:

$$\sigma_{\text{perm-dist}} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (d_{\text{perm},i} - \bar{d}_{\text{perm}})^2}$$

where:

$$\bar{d}_{\text{perm}} = \frac{1}{N} \sum_{i=1}^{N} d_{\text{perm},i}$$

Similarly, the standard deviation of **permutation distances after canonicalization** is:

$$\sigma_{\text{canon-dist}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(d_{\text{canon},i} - \bar{d}_{\text{canon}})^2}$$

where:

$$\bar{d}_{\text{canon}} = \frac{1}{N}\sum_{i=1}^{N} d_{\text{canon},i}$$

Interpretation

- **Higher** $\sigma_{\text{perm-dist}}$ $\rightarrow$ Large variability in permutation distances across different problem permutations.
- **Higher** $\sigma_{\text{canon-dist}}$ $\rightarrow$ Canonicalization does not effectively eliminate permutation differences.
- **If** $\sigma_{\text{canon-dist}} < \sigma_{\text{perm-dist}}$ $\rightarrow$ Canonicalization reduces structural variation.

Summary of Metrics

Metric	Formula	Interpretation
Solve-Time Variability (Original + Permuted)	σ_{perm}	Measures fluctuation in solve times across problem permutations
Solve-Time Variability (Canonical Forms)	σ_{canon}	Measures fluctuation in solve times of canonicalized problems
Permutation Distance Variability (Before Canonicalization)	$\sigma_{\text{perm-dist}}$	Measures how much the problem's structure changes across permutations
Permutation Distance Variability (After Canonicalization)	$\sigma_{\text{canon-dist}}$	Measures if canonicalization is making structures more consistent