Row-Partition Branching for Set Partitioning Problem

NAMSUK CHO, JEFF LINDEPROM

University of Wisconsin-Madison

Set Partitioning Problem

\[ \min c^T x \]
\[ s.t. Ax = e \]
\[ x \in \{0, 1\}^n, \]

where \( A \in \{0, 1\}^{m \times n}, e \) is an m-dimensional vector of ones.

• Denote row index \( I = \{1, 2, \ldots, m\} \) and column index set \( J = \{1, 2, \ldots, n\} \).

• SOS-1 (GUB): \( \sum_{j \in J} x_j = 1 \) where for each row \( i \in I, J_i = \{ j \in J | a_{ij} = 1 \} \).

Ryan-Foster Branching

• Based on the logic that in any feasible solution to (SPP), each pair of rows \((p, q) \in (I \times I)\) is either covered by the same column or by two different columns.

\[ x_j x_k \cdots x_{k+1} x_j \]
\[ (p \ 1 \ 0 \ 0 \ 1 \ 1) \]
\[ q \ 0 \ 1 \ 1 \ 0 \ 1 \ 1 \]

• Subproblems:

\[ \sum_{j \in J} x_j = 1 \quad \text{or} \quad \sum_{j \in J} x_j = 2, \]

Row-Partition Branching

• Consider any subset of rows \( S \subseteq I \) with \( |S| = s \), suppose \( s = 5 \).

\[ C_k(S) = \begin{cases} 1 \quad & (j \in J | \sum_{j \in C(S)} a_{ij} = k) \\ 0 \quad & \text{otherwise} \end{cases} \]

• e.g., 5 rows partition

\[ \begin{align*}
1.5 - 5 & = \text{cover 1 column in } C_1(S), \text{ or} \\
2.5 - 4 + 1 & = \text{cover 1 column in } C_2(S) \text{ and 1 column in } C_3(S), \text{ or} \\
3.5 - 3 + 2 & = \text{cover 1 column in } C_1(S) \text{ and 1 column in } C_2(S), \text{ or} \\
4.5 - 3 + 1 & = \text{cover 1 column in } C_2(S) \text{ and 2 columns in } C_3(S), \text{ or} \\
5.5 - 2 & = \text{cover 2 columns in } C_2(S) \text{ and 1 column in } C_1(S), \text{ or} \\
6.5 - 1 & = \text{cover 1 columns in } C_3(S) \text{ and 3 columns in } C_1(S), \text{ or} \\
7.5 - 1 & = \text{cover 5 columns in } C_1(S).
\end{align*} \]

Row-Partition Branching : 3 rows branching

\[ S = \{r, p, q\} \]

\[ \begin{align*}
A & = \begin{pmatrix}
x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\
1 & 1 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 1 & 0
\end{pmatrix} \\
& = \begin{pmatrix}
r & 1 & 1 & 0 & 1 & 1 & 0 \\
q & 1 & 0 & 1 & 1 & 0 & 1
\end{pmatrix}
\end{align*} \]

Node 1:

\[ \sum_{j \in C_1(S)} x_j = 1 \quad (x_1 = 1) \]

Node 2:

\[ \sum_{j \in C_2(S)} x_j = 1 \quad (x_2 + x_3 + x_5 = 1), \quad \sum_{j \in C_3(S)} x_j = 1 \quad (x_3 + x_5 + x_7 = 1) \]

Node 3:

\[ \sum_{j \in C_3(S)} x_j = 3 \quad (x_3 + x_5 + x_7 = 3) \]

We prefer not to branch on constraints (if we can avoid it)

Fixing Variable Bounds

Node 2:

\[ \sum_{j \in C_2(S)} x_j = 1 \quad (x_1 = 1), \quad \sum_{j \in C_3(S)} x_j = 1 \quad (x_2 + x_3 + x_5 = 1) \]

Node 3:

\[ \sum_{j \in C_3(S)} x_j = 3 \quad (x_3 + x_5 + x_7 = 3) \]

• Node 2 : \( x_1 = 0 \quad \forall j \in C_1(S) \Rightarrow \sum_{j \in C_1(S)} x_j = 1 \quad \forall j \in C_1(S) \)

• Sub-divide node 2 by enumerating 3 cases for the row that will be covered by the column in \( C_1(S) \).

• (2a) (Row p): \( x_1 = 0 \quad \forall j \in C_1(S), \forall j \in C_2(p, q), \forall j \in C_2(p, r), \forall j \in C_3(q, r) \)

• (2b) (Row q): \( x_1 = 0 \quad \forall j \in C_1(S), \forall j \in C_2(p, q), \forall j \in C_2(p, r), \forall j \in C_3(q, r) \), \( x_1 \in C_1(p, q) \)

• (2c) (Row r): \( x_1 = 0 \quad \forall j \in C_1(S), \forall j \in C_2(p, q), \forall j \in C_2(p, r), \forall j \in C_3(q, r) \), \( x_1 \notin C_1(p, q) \)

\[ S = \{r, p, q\} \]

Table: Results for very large instances

Possible alternatives of Row-Partition Branching

• 2R : “pure” 2-row branching (Ryan-Foster): \( S \in \arg \max_{S \subseteq \rho} T_2(S) \).

• 3R : “pure” 3-row branching: \( S \in \arg \max_{S \subseteq \rho} \frac{T_3}{\rho} \).

• DM : Dynamic Method (Heuristic): choose to do 3-row partition branching only if the resulting child nodes are estimated to be “strong” compared to the best two-row set. \( T_3(S) > \frac{3}{2} T_2(S) \).

Experimental Settings

• Test set

• 390 instances from MIPLIB and CORRIL (create 30 scrambled clones from 13 instances)

• 499 random correlated-coefficient (CC): PAR, PAC, sparse, dense

• Run all the experiments via HTCondor

Computations Results

Figure: Performance comparison - number of nodes explored - CC PAR instance

Figure: Performance comparison - number of nodes explored - CC PAC instance

Figure: Performance comparison - number of nodes explored - benchmark instance

Conclusions

• We described a novel branching strategy based on partitioning a subset of the rows for (SPP)

• 2R, 3R and DM are all of comparable quality, and all appear to perform better than all other methods in terms of nodes explored

• For larger instances, doing 3R and DM can lead to significant computational improvement.