Detecting Implied Integers using Totally Unimodular Submatrices

Rolf van der Hulst* - r.p.vanderhulst@utwente.nl, University of Twente

* The author acknowledges funding support from the Dutch Research Council (NWO) on grant number OCENW.M20.151.

Implied Integers

Setting: mixed-integer linear program (MILP) with variable set N and integers $I \subseteq N$.

Definition: $J \subseteq N$ are *implied integers*, if for each feasible solution x to the relaxed problem, there exists a solution $x' = (x_{N\setminus J}, x'_J)$ that is feasible for the original problem such that $c^\intercal x' \leq c^\intercal x$ and x'_J is integral.

Original problem	Relaxed problem	Tightened problem
$\min c^\intercal x$	$\min c^\intercal x$	$\min c^\intercal x$
s.t. $Ax \leq b$	s.t. $Ax \leq b$	s.t. $Ax \leq b$
$x_j \in \mathbb{Z}, \forall j \in \mathbf{I}$	$x_j \in \mathbb{Z}, \forall j \in \mathbf{I}\setminus\mathbf{J}$	$x_j \in \mathbb{Z}, \forall j \in \mathbf{I}\cup\mathbf{J}$

Goal: detect implied integers in mixed-integer linear programs.

Why are implied integers useful?

Theorem 1. If $J \subseteq N$ are implied integers, then the original problem, the relaxed problem and the tightened problem are equivalent.

Best of both worlds:

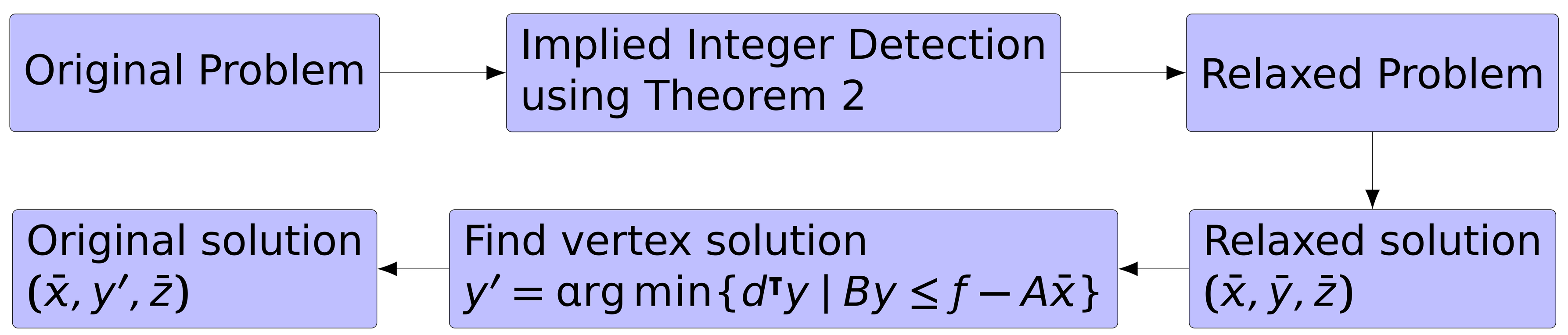
- Relaxed problem: no need to branch on variables in J
- Tightened problem: use integrality of x_J in cutting planes, presolving, domain propagation, etc.

Implied Integers and Total Unimodularity

Theorem 2. Consider a MILP of the form

$$\begin{aligned} \min\ & c^\intercal x + d^\intercal y + e^\intercal z \\ \text{s.t. } & Ax + By \leq f \\ & Dx + Ez \leq g \\ & x \in \mathbb{Z}^{n_1} \\ & y \in \mathbb{Z}^{n_2} \times \mathbb{R}^{n_3} \\ & z \in \mathbb{R}^{n_4} \end{aligned}$$

If B is totally unimodular and A and f are integral, then y are implied integers.



References

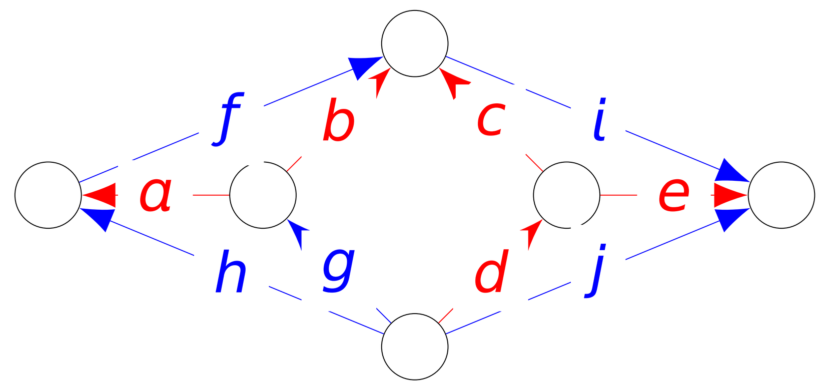
[1] R. E. Bixby and D. K Wagner. An almost linear-time algorithm for graph realization. *Mathematics of Operation research*, 13(1), 1988.

[2] A. Gleixner et al. MIPLIB 2017: data-driven compilation of the 6th mixed-integer programming library. *Mathematical Programming Computation*, 13(3):443–490, sep 2021.

Detecting Network Matrices

Detecting totally unimodular (TU) matrices is too slow. We detect **network matrices**, a large subclass of TU matrices, instead.

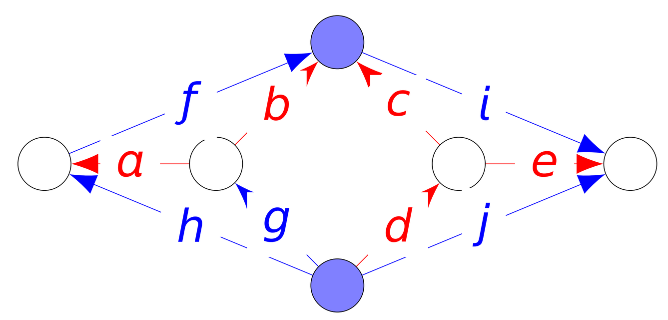
$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} f & g & h & i & j \end{array} \\ \begin{bmatrix} -1 & 0 & 1 & 0 & 0 \\ 1 & -1 & -1 & 0 & 0 \\ 0 & 1 & 1 & -1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array}$$



A matrix is a network matrix if there exists a directed spanning tree such that the ± 1 entries of each column correspond to a oriented path in the tree.

Repeatedly solve the **Column Augmentation Problem:** Given network matrix M and column b, is $M' = \begin{bmatrix} M & b \end{bmatrix}$ a network matrix?

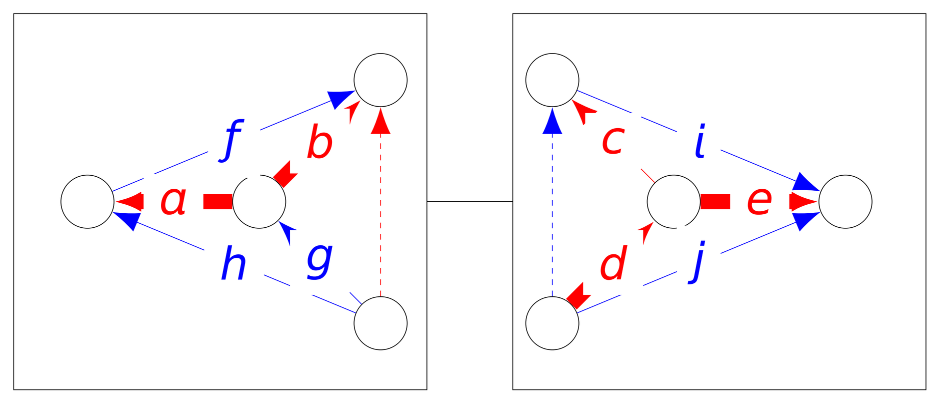
Main Challenge: ambiguity. We can *flip* the graph around any 2-separation, to find another graph with the same matrix.

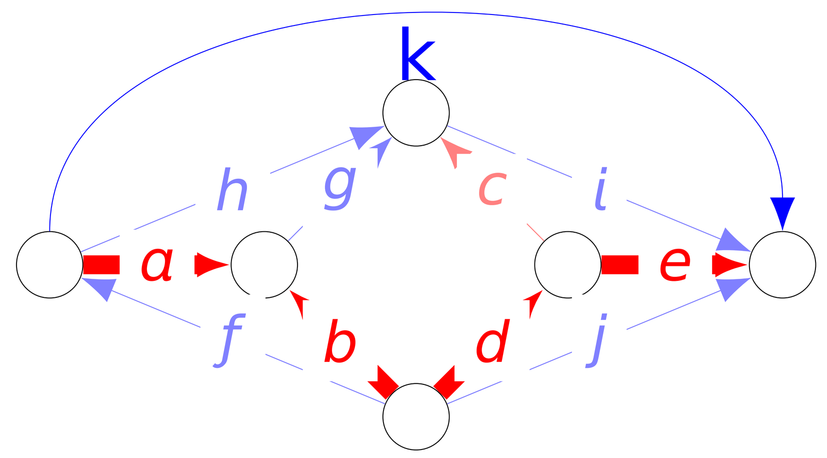


Solution: use *SPQR trees* (3-connected components) to represent all 2-separations of G and all graphs of M.

Solving the column augmentation problem: determine if the nonzeros of b form an oriented path [1]. The path problem *decomposes* over the SPQR tree!

$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{cccccc} f & g & h & i & j & k \end{array} \\ \left[\begin{array}{ccccc|c} -1 & 0 & 1 & 0 & 0 & 1 \\ 1 & -1 & -1 & 0 & 0 & -1 \\ 0 & 1 & 1 & -1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{array}\right] \end{array}$$





Our second contribution: we developed an algorithm for the **Row Augmentation Problem** that uses similar ideas.

Algorithm

We detect the structure of Theorem 2 by growing B, which we do by solving the row/column augmentation problem.

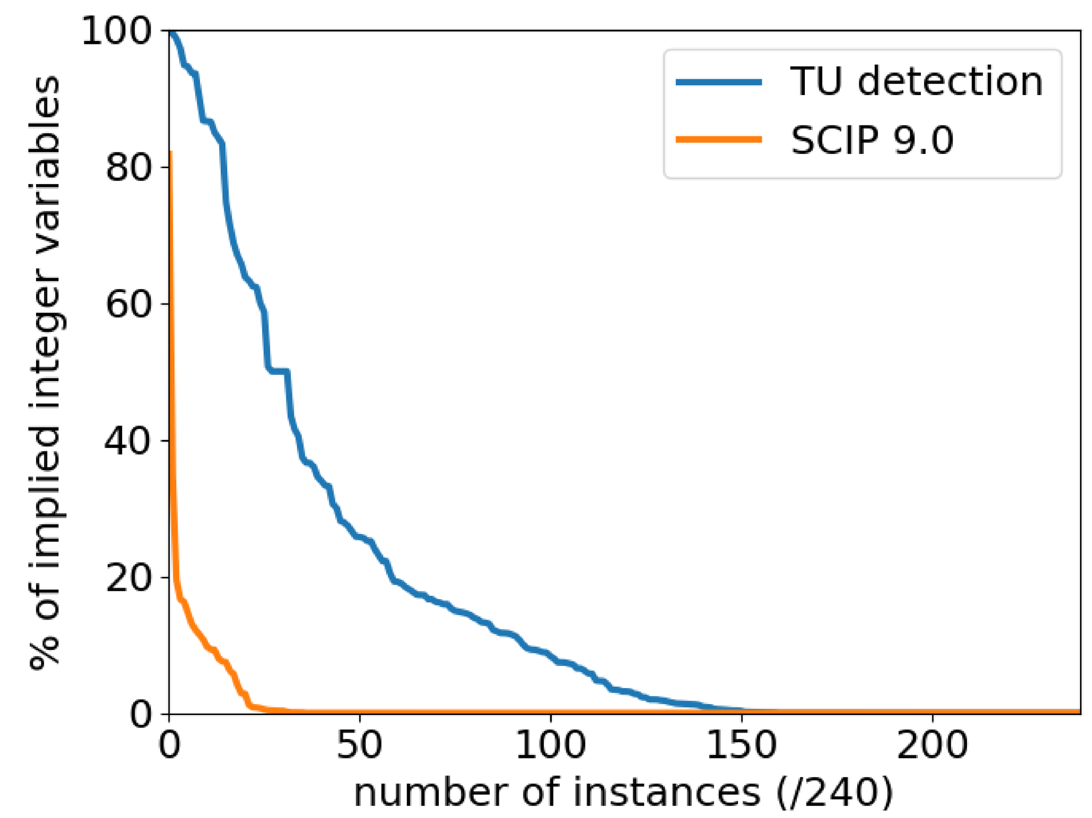
Phases:

1. For each block of the submatrix formed by the continuous variables, determine if it can augment B.
2. Greedily augment B with the columns of integer variables.
3. Run steps 1-2 for network matrices (column augmentation) and transposed network matrices (row augmentation). Use the one that has the most columns.

Results

Comparison on MIPLIB 2017 benchmark set [2] with SCIP 9.0 as baseline.

Method	SCIP 9.0	TU detection
Mean % of implied integer variables	1.3%	**16.4%**
# affected instances (out of 240)	42	**162**



- Mean detection time: **0.2 seconds**
- Strange: 3% increase in solving time, 2% increase in nodes

Conclusion & Discussion

- We can quickly find an **order of magnitude** more implied integers in MILP's from practice than the current methods.
- Detected implied integers do not enhance performance of SCIP; further research into exploiting implied integers in SCIP is necessary.