Finding Embedded Multi-Commodity Flow Submatrices in MIPs and Separation of Cutset Inequalities

Christian Raack Tobias Achterberg

Cooperation of the Zuse-Institute Berlin and ILOG

Aussois 2009

Changing the rules of business™

Outline

Introduction
Network Detection
Separation

Introduction

$$\begin{aligned}&\min\ cx\\ &\text{s.t. } Ax \le b,\quad x \in \mathbb{Z}^I \times \mathbb{R}^C\end{aligned} \qquad \textbf{(MIP)}$$

Cutting Planes in Cplex

clique, cover, disjunctive, flow cover, flow path, gomory, gub, implied bounds, mir, zero-half

- Rather general – work for most MIPs
- Not "consequently" exploit structure of constraint matrix A
- No "real" knowledge about the underlying problem

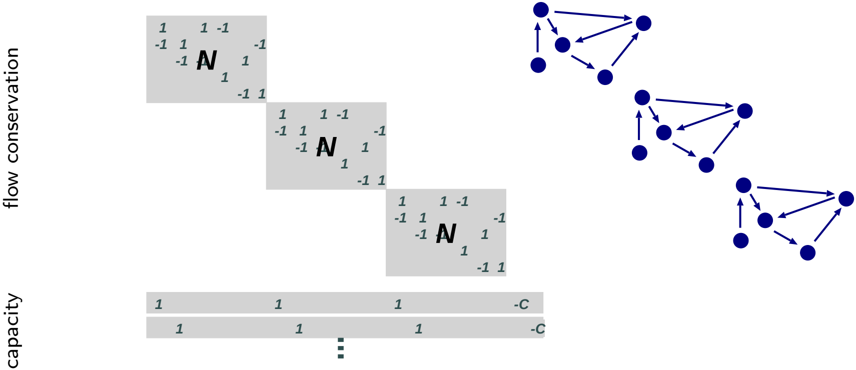
Introduction

$$\min cx \quad \text{s.t. } Ax \leq b, \ \ x \in \mathbb{Z}^I \times \mathbb{R}^C \qquad \textbf{(MIP)}$$

Idea

- Tons of polyhedral studies for special problems
  $\longrightarrow$ network design, facility location, scheduling, steiner tree ...
- Results (facets) not used in general MIP solvers except for "simple" relaxations such as knapsack sets, single node flow sets, stable set relaxations
- Why not investing more time for problem identification ?
- And generate (more) problem specific cutting planes !

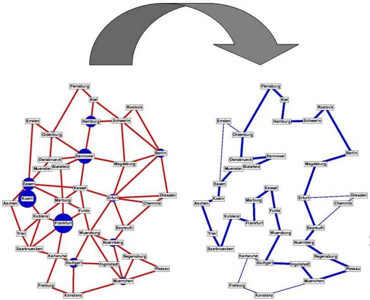
Coupled Multi-Commodity Flow (MCF)



block structure: flow for every commodity, network matrix N

coupling: capacity constraints for arcs, $\text{Flow}(a) \leq \text{Capacity}(a)$

Network Design



given potential network topology, user demands, link capacities

find dimensioning of the links + MCF flow

such that demands are satisfied and (some) cost is minimal

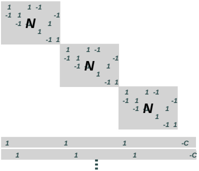
Applications: telecommunication, public transport, ...

Modeling: link-flow formulation

MCF flow

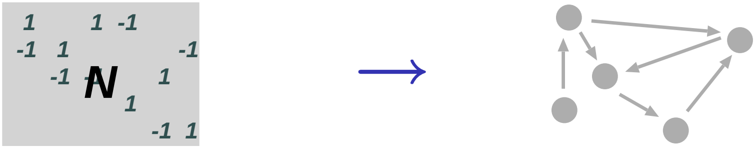
Capacity

$\longrightarrow$



Introduction
Network Detection
Separation

Network Detection – Single-Commodity



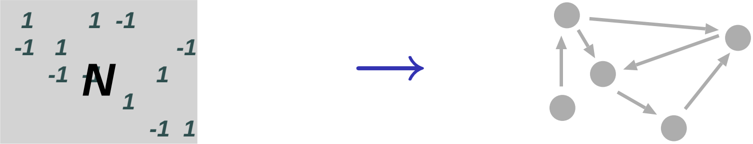
Network detection (in the context of the network simplex):

Literature: Brown & Wright [84], Bixby & Fourer [88], Gülpinar et al. [98, 04], Gutin & Zverovitsch [04], Figueiredo & Labbe & Souza [07]

Approaches: Row/column-scanning addition/deletion, Signed graphs, IP formulation

We use **Row scanning addition**, it is simple, fast, and successful

Network detection – Single-Commodity

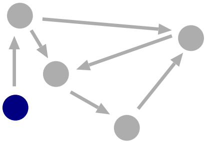


Row Scanning Addition [BixbyFourer '88]

- Start with empty set of rows
- Add adjacent flow row so that the subset remains a network
  $\longrightarrow$ Valid network submatrix after every step
- If necessary scale and/or reflect rows

Network detection – Single-Commodity

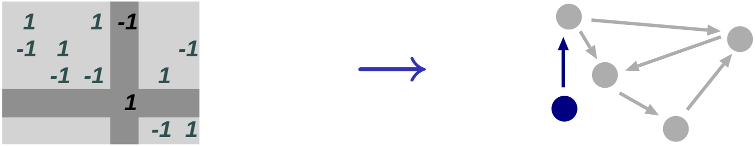
$$\begin{matrix} 1 & & 1 & -1 & & \\ -1 & 1 & & & & -1 \\ & -1 & -1 & & 1 & \\ & & & 1 & & \\ & & & & -1 & 1 \end{matrix} \longrightarrow$$



Row Scanning Addition [BixbyFourer '88]

- Start with empty set of rows
- Add adjacent flow row so that the subset remains a network
  $\longrightarrow$ Valid network submatrix after every step
- If necessary scale and/or reflect rows

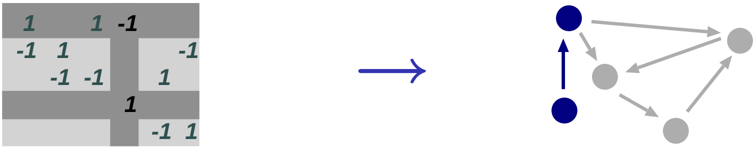
Network detection – Single-Commodity



Row Scanning Addition [BixbyFourer '88]

- Start with empty set of rows
- Add adjacent flow row so that the subset remains a network
  $\longrightarrow$ Valid network submatrix after every step
- If necessary scale and/or reflect rows

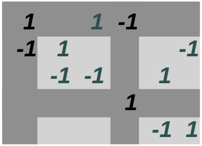
Network detection – Single-Commodity



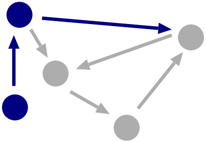
Row Scanning Addition [BixbyFourer '88]

- Start with empty set of rows
- Add adjacent flow row so that the subset remains a network
  $\longrightarrow$ Valid network submatrix after every step
- If necessary scale and/or reflect rows

Network detection – Single-Commodity



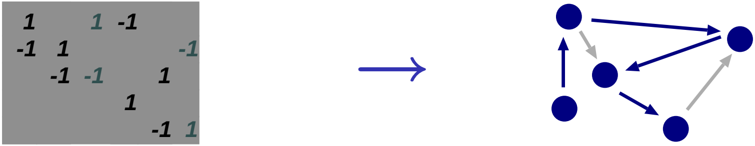
$\longrightarrow$



Row Scanning Addition [BixbyFourer '88]

- Start with empty set of rows
- Add adjacent flow row so that the subset remains a network
  $\longrightarrow$ Valid network submatrix after every step
- If necessary scale and/or reflect rows

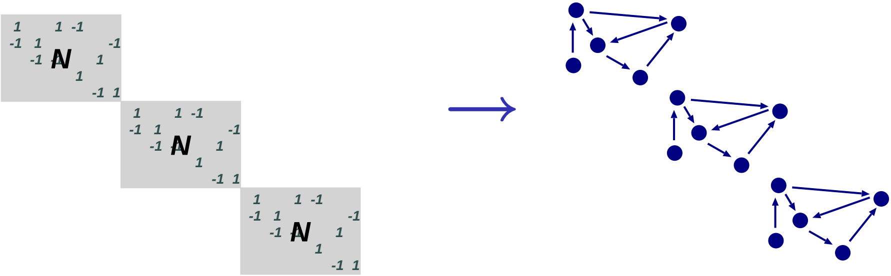
Network detection – Single-Commodity



Row Scanning Addition [BixbyFourer '88]

- Start with empty set of rows
- Add adjacent flow row so that the subset remains a network
  $\longrightarrow$ Valid network submatrix after every step
- If necessary scale and/or reflect rows

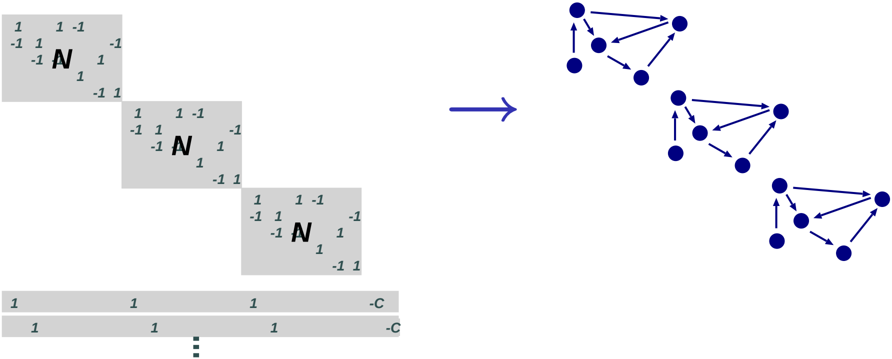
Network Detection – Multi-Commodity



- Row Scanning Addition $\longrightarrow$ one graph, several components
- How can we detect isomorphism of components ?

$\longrightarrow$ Bad News: Complexity of GraphIsomorphism unknown

Network Detection – Multi-Commodity

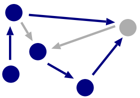


- Row Scanning Addition ⟶ one graph, several components
- How can we detect isomorphism of components ?

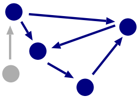
⟶ Bad News: Complexity of GraphIsomorphism unknown
⟶ Good News: We can hopefully use the coupling constraints !!

Network Detection – Challenges

- User preprocessing:
  Omitting one flow row per commodity
  $\Longrightarrow$ different node missing per commodity
  No flow into source nodes
  $\Longrightarrow$ different arcs missing per commodity



- Solver preprocessing:
  Fixing, Substituting
  $\Longrightarrow$ deletes loosely connected nodes
  (in some commodities)



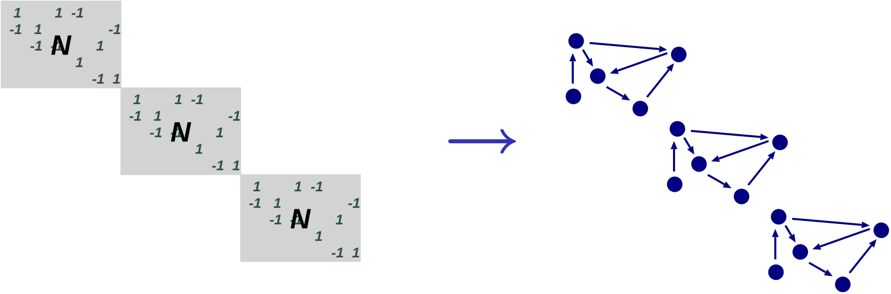
- Various model formulations
  (directed, undirected, single path,...)
- Additional side constraints

Network Detection – Algorithm

1. Flow Detection
   - Identify and sort flow row candidates
   - Row Scanning Addition
   - Throw away trash (small components)

   Result: Flow system with several components

   $\longrightarrow$ flow variables $\leftrightarrow$ commodity-id, flow row $\leftrightarrow$ commodity-id

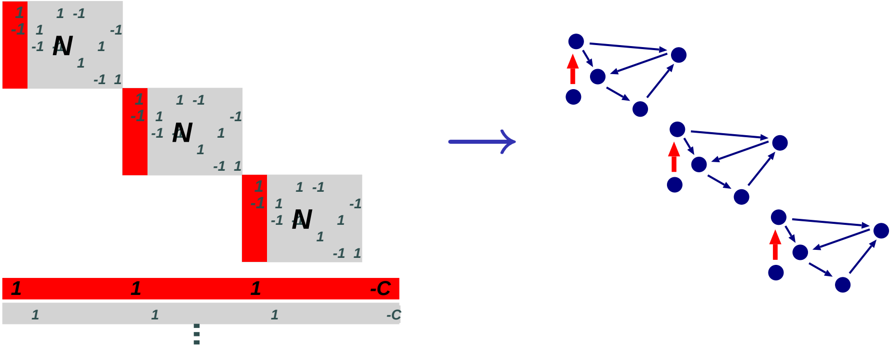


Network Detection – Algorithm

2. Arc Detection
   - Identify and sort capacity row candidates
   - capacity row should have entry in most of the commodities
   - Assign arc-id to capacity row and corresponding flow variables

   Result: Arcs known

   $\longrightarrow$ flow variable $\leftrightarrow$ arc-id, capacity row $\leftrightarrow$ arc-id

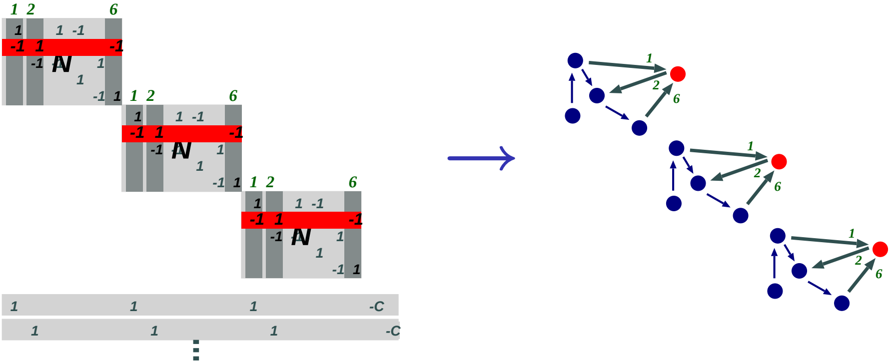


Network Detection – Algorithm

3. Node Detection
   - Assign node-id to flow rows (in different commodities) with similar incidence pattern w.r.t arc-ids

Result: Nodes known

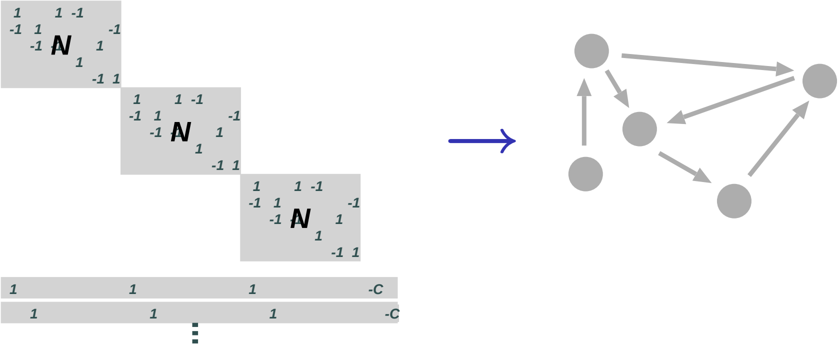
$\longrightarrow$ flow row $\leftrightarrow$ node-id



Network Detection – Algorithm

4. Construct MCF network
   - Construct incidence function of graph
   - Ask all flow variables of an arc for source (target)
   - Majority vote wins
   - **inconsistency count** $+=$ sum of minority votes

Result: MCF network + measure for quality of detection



Network Detection – Results

set	#	origin	description
arc.set	35	A. Atamtürk	MCF, unsplittable and splittable, binary cap
avub	60	A. Atamtürk	randomly generated, SCF, binary caps + GUB
cut.set	15	A. Atamtürk	MCF, integer caps
fc	20	A. Atamtürk	SCF, fixed charge, binary cap
fctp	28	J. Gottlieb	SCF, complete bipartite, binary cap
sndlib	52	ZIB	MCF, integer caps or binary caps +GUB
ufcn	84	L.A. Wolsey	SCF, fixed charge, binary cap, big M

- Network known for roughly half of the instances (# nodes, # arcs, # commodities, demands, capacities)
- SCIP preprocessing off: Detection works correctly, cut.set fails **inconsistency ratio** = **0.0032** (all - cut.set), $\gg$ 1 (cut.set)
- SCIP preprocessing on: Detected works but graphs are smaller **inconsistency ratio** = **0.01** (all - cutset), $\gg$ 1 (cut.set) detected graphs have **-22% nodes, -15% arcs**
- inconsistency ratio = # inconsistencies / # arcs / # coms

Introduction
Network Detection
Separation

Separation – Approach

Given:

- MCF network
- flow row $\leftrightarrow$ node/commodity, capacity row $\leftrightarrow$ arc

Idea:

- Use well known machinery for network design problems
- Classical cutting planes, known successful separation routines
- Separate cut based inequalities (e.g. cutset ineqs)

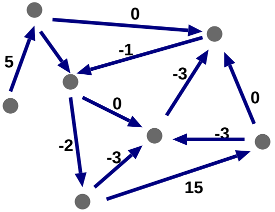
Difference:

- We cannot directly work on the graph
- Modify general c-Mixed Integer Rounding framework (c-MIR – Marchand & Wolsey [98])
- Use network based row aggregation heuristic
- Switch on separation only if inconsistency ratio small (< 0.2) !

Separation – Finding network cuts

Basic Idea: Bienstock et. al [98], Günlük [99]

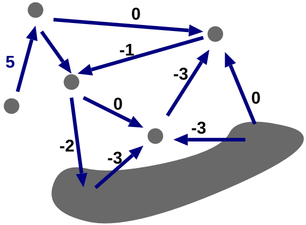
- Find tight cut $\rightarrow$ Capacity(cut) $=$ Flow(cut)
- Motivation: tight base inequalities $\rightarrow$ violated MIR inequalities
- For arc a define weight $w_a = \text{slack}(a) - |\text{dual}(a)|$ w.r.t. capacity constraint of a
- Contract arcs with large weight to get small network partition (e.g. size 2-8, we used size 4)



Separation – Finding network cuts

Basic Idea: Bienstock et. al [98], Günlük [99]

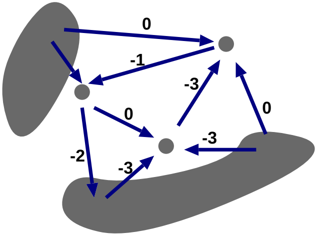
- Find tight cut $\rightarrow$ Capacity(cut) = Flow(cut)
- Motivation: tight base inequalities $\rightarrow$ violated MIR inequalities
- For arc a define weight $w_a = \text{slack}(a) - |\text{dual}(a)|$ w.r.t. capacity constraint of a
- Contract arcs with large weight to get small network partition (e.g. size 2-8, we used size 4)



Separation – Finding network cuts

Basic Idea: Bienstock et. al [98], Günlük [99]

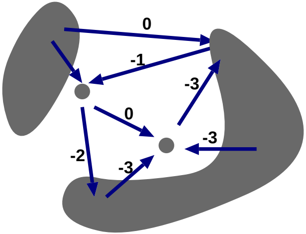
- Find tight cut $\longrightarrow$ Capacity(cut) = Flow(cut)
- Motivation: tight base inequalities $\longrightarrow$ violated MIR inequalities
- For arc a define weight $w_a = \text{slack}(a) - |\text{dual}(a)|$ w.r.t. capacity constraint of a
- Contract arcs with large weight to get small network partition (e.g. size 2-8, we used size 4)



Separation – Finding network cuts

Basic Idea: Bienstock et. al [98], Günlük [99]

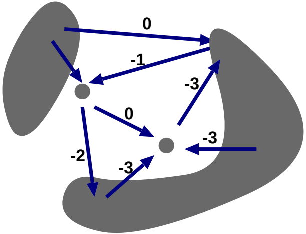
- Find tight cut $\longrightarrow$ Capacity(cut) $=$ Flow(cut)
- Motivation: tight base inequalities $\longrightarrow$ violated MIR inequalities
- For arc a define weight $w_a = \mathrm{slack}(a) - |\mathrm{dual}(a)|$
  w.r.t. capacity constraint of a
- Contract arcs with large weight to get small network partition (e.g. size 2-8, we used size 4)



Separation – Finding network cuts

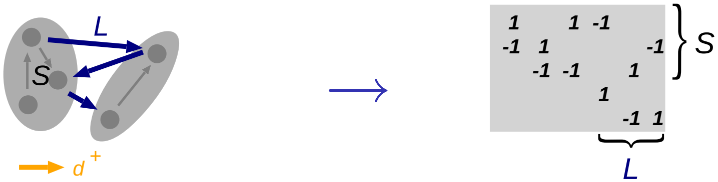
Basic Idea: Bienstock et. al [98], Günlük [99]

- Find tight cut $\longrightarrow$ Capacity(cut) = Flow(cut)
- Motivation: tight base inequalities $\longrightarrow$ violated MIR inequalities
- For arc a define weight $w_a = \text{slack}(a) - |\text{dual}(a)|$ w.r.t. capacity constraint of a
- Contract arcs with large weight to get small network partition (e.g. size 2-8, we used size 4)



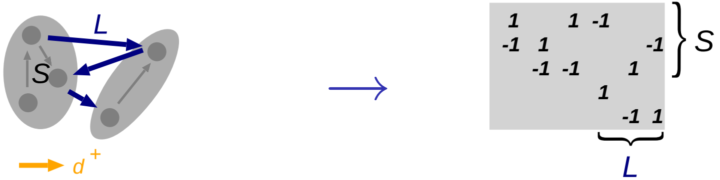
Enumerate all cuts in the resulting partition

Separation – Row aggregation and MIR



Given $S \subset V$ and corresponding cut $L = L^+ \cup L^-$.

Separation – Row aggregation and MIR



Given $S \subset V$ and corresponding cut $L = L^{+} \cup L^{-}$.

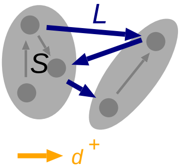
- Add all flow rows w.r.t. S (for commodities with source in S).
  $\rightarrow \quad f(L^{+}) - f(L^{-}) = d^{+} > 0$ Cancellation!

Separation – Row aggregation and MIR

Given $S \subset V$ and corresponding cut $L = L^+ \cup L^-$.

- Add all flow rows w.r.t. S (for commodities with source in S).
  $\rightarrow \quad f(L^+) - f(L^-) = d^+ > 0$ Cancellation!
- Add all capacity constraints for L^+: $Cx(L^+) - f(L^+) \geq 0$
  $\rightarrow \quad Cx(L^+) - s \geq d^+$ (base inequality)

Separation – Row aggregation and MIR



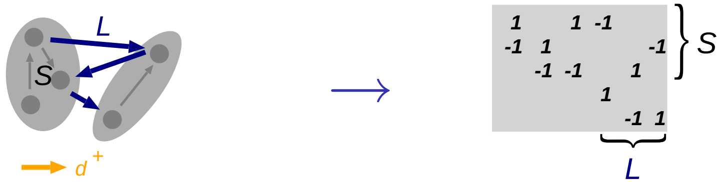
$\longrightarrow$

$$\begin{array}{cccccc|c}
1 & & 1 & -1 & & & \\
-1 & 1 & & & & -1 & S \\
& -1 & -1 & & 1 & & \\
\hline
& & & 1 & & & \\
& & & & -1 & 1 & \\
& & & & L & &
\end{array}$$

Given $S \subset V$ and corresponding cut $L = L^+ \cup L^-$.

- Add all flow rows w.r.t. S (for commodities with source in S).
  $\rightarrow \quad f(L^+) - f(L^-) = d^+ > 0$ Cancellation!
- Add all capacity constraints for L^+: $Cx(L^+) - f(L^+) \geq 0$
  $\rightarrow \quad Cx(L^+) - s \geq d^+$ (base inequality)
- Divide by $C > 0$ (one of the coeffs) and apply MIR
  $\rightarrow \quad x(L^+) \geq \left\lceil \frac{d^+}{C} \right\rceil$ (MIR cutset inequality)

Separation – Row aggregation and MIR



Given $S \subset V$ and corresponding cut $L = L^+ \cup L^-$.

- Add all flow rows w.r.t. S (for commodities with source in S).
  $\rightarrow \quad f(L^+) - f(L^-) = d^+ > 0$ Cancellation!
- Add all capacity constraints for L^+: $Cx(L^+) - f(L^+) \geq 0$
  $\rightarrow \quad Cx(L^+) - s \geq d^+$ (base inequality)
- Divide by $C > 0$ (one of the coeffs) and apply MIR
  $\rightarrow \quad x(L^+) \geq \left\lceil \frac{d^+}{C} \right\rceil$ (MIR cutset inequality)

Aggregation of many rows, nevertheless sparse inequality

Separation – Results

- 2 testsets, instances solvable within 1 hour with SCIP 1.1
- Network Design instances: 180, SCIP team testset: 329

	ND	SCIP team
size	180	329
network found	177	246
small inconsistency	165	80
violated ineqs	157	44
time ratio	0.63	0.95
node ratio	0.55	0.79

- ratios: geometric mean of $\frac{\text{time\_mcf} + 1}{\text{time\_default} + 1}$ and $\frac{\text{nodes\_mcf} + 50}{\text{nodes\_default} + 50}$
- geometric mean over instances with separated inequalities
- no time increase for the rest $\longrightarrow$ fast detection, fast separation