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
Outline

Introduction
Network Detection
Separation
Introduction

\[ \begin{align*}
\text{min } c x \\
\text{s.t. } A x &\leq b, \quad x \in \mathbb{Z}^I \times \mathbb{R}^C \end{align*} \]

\text{(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

\[ \begin{align*}
\min & \quad c^T x \\
\text{s.t.} & \quad Ax \leq b, \quad x \in \mathbb{Z}^I \times \mathbb{R}^C
\end{align*} \tag{MIP} \]

Idea

- Tons of polyhedral studies for special problems
  → 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
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 → 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 → 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
  → 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 → 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 → 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 → Valid network submatrix after every step
- If necessary scale and/or reflect rows
Network Detection – Multi-Commodity

- Row Scanning Addition → one graph, several components
- How can we detect isomorphism of components?
  → Bad News: Complexity of Graph Isomorphism unknown
Network Detection – Multi-Commodity

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

$\rightarrow$ Bad News: Complexity of Graph Isomorphism unknown
$\rightarrow$ Good News: We can hopefully use the coupling constraints!!
Network Detection – Challenges

- **User preprocessing:**
  Omitting one flow row per commodity
  \[\Rightarrow\] different node missing per commodity
  No flow into source nodes
  \[\Rightarrow\] different arcs missing per commodity

- **Solver preprocessing:**
  Fixing, Substituting
  \[\Rightarrow\] 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
→ flow variables ↔ commodity-id, flow row ↔ 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
→ flow variable ↔ arc-id, capacity row ↔ arc-id
3. Node Detection

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

Result: Nodes known

→ flow row ↔ 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 $\pm =$ sum of minority votes

Result: MCF network + measure for quality of detection
Network Detection – Results

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

- 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), ≫ 1 (cut.set)
- **SCIP preprocessing on**: Detected works but graphs are smaller
  inconsistency ratio = 0.01 (all - cutset), ≫ 1 (cut.set)
  detected graphs have -22% nodes, -15% arcs
- inconsistency ratio = # inconsistencies / # arcs / # coms
Separation – Approach

Given:

- MCF network
- flow row ↔ node/commodity, capacity row ↔ 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)!
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 $\rightarrow$ Capacity(cut) = Flow(cut)
- Motivation: tight base inequalities $\rightarrow$ violated MIR inequalities
- For arc $a$ define weight $w_a = \text{slack}(a) - |\text{dual}(a)|$
- 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 $\rightarrow$ 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)

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 $S$).
  
  $$f(L^+) - f(L^-) = d^+ > 0$$

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 $S$).
  $\rightarrow f(L^+) - f(L^-) = d^+ > 0$  
  Cancellation!

- Add all capacity constraints for $L^+$: $Cx(L^+) - f(L^+) \geq 0$
  $\rightarrow Cx(L^+) - s \geq d^+$ (base 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 \( S \)).
  \[ f(L^+) - f(L^-) = d^+ > 0 \]
  **Cancellation!**

- Add all capacity constraints for \( L^+ \):
  \[ Cx(L^+) - f(L^+) \geq 0 \]
  \[ Cx(L^+) - s \geq d^+ \]
  (base inequality)

- Divide by \( C > 0 \) (one of the coeffs) and apply MIR
  \[ x(L^+) \geq \left\lceil \frac{d^+}{C} \right\rceil \]
  (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 $S$).
  $\rightarrow f(L^+) - f(L^-) = d^+ > 0$  \hspace{1cm} \text{Cancellation!}$

- Add all capacity constraints for $L^+$: $Cx(L^+) - f(L^+) \geq 0$
  $\rightarrow Cx(L^+) - s \geq d^+$ \hspace{1cm} \text{(base inequality)}$

- Divide by $C > 0$ (one of the coeffs) and apply MIR
  $\rightarrow x(L^+) \geq \left\lceil \frac{d^+}{C} \right\rceil$ \hspace{1cm} \text{(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

<table>
<thead>
<tr>
<th></th>
<th>ND</th>
<th>SCIP team</th>
</tr>
</thead>
<tbody>
<tr>
<td>size</td>
<td>180</td>
<td>329</td>
</tr>
<tr>
<td>network found</td>
<td>177</td>
<td>246</td>
</tr>
<tr>
<td>small inconsistency</td>
<td>165</td>
<td>80</td>
</tr>
<tr>
<td>violated ineqs</td>
<td>157</td>
<td>44</td>
</tr>
<tr>
<td>time ratio</td>
<td>0.63</td>
<td>0.95</td>
</tr>
<tr>
<td>node ratio</td>
<td>0.55</td>
<td>0.79</td>
</tr>
</tbody>
</table>

- ratios: geometric mean of \( \frac{\text{time}_\text{mcf} + 1}{\text{time}_\text{default} + 1} \) and \( \frac{\text{nodes}_\text{mcf} + 50}{\text{nodes}_\text{default} + 50} \)
- geometric mean over instances with separated inequalities
- no time increase for the rest → fast detection, fast separation