Piecewise Linear Multicommodity Flow Problems

Bernard Gendron\textsuperscript{1}

December 2016

LAAS, Toulouse

\textsuperscript{1} CIRRELT and DIRO, U. Montréal
Outline

Problem description

Extended model and polyhedral results

Structured Dantzig-Wolfe decomposition

Computational results
Multicommodity flow problem

- Directed network $G = (N, A)$ with node set $N$ and arc set $A$
- Commodity set $K$: origins $O(k)$, destinations $D(k)$ and transshipment nodes $T(k)$ for each $k \in K$
- Demand $d^k_i$ for commodity $k$ at node $i$:
  
  $> 0$ if $i \in O(k)$
  $< 0$ if $i \in D(k)$
  $= 0$ if $i \in T(k)$

- Capacity $u_a$ on each arc $a$
Mathematical formulation

\( x_a^k \): flow of commodity \( k \) on arc \( a \)

\[
\sum_{a \in F_i} x_a^k - \sum_{a \in B_i} x_a^k = d_i^k, \quad i \in N, \ k \in K, \quad (1)
\]

\[
x_a^k \geq 0, \quad a \in A, \ k \in K, \quad (2)
\]

\[
x_a \equiv \sum_{k \in K} x_a^k \leq u_a, \quad a \in A.
\]

\( F_i = \{ a \in A | a = (i, j), j \in N \}, \ B_i = \{ a \in A | a = (j, i), j \in N \} \)
The objective is to minimize the sum of $|A|$ piecewise linear cost functions $g_a(x_a)$ such that:

- $g_a(0) = 0$
- $g_a(x_a)$ is lower semi-continuous
- $g_a(x_a)$ is non-decreasing

$S_a$: segments of $g_a(x_a)$

- $0 = b_a^0 < b_a^1 < \cdots < b_a^{|S_a|} = u_a$ (breakpoints)
- $c_a^s \geq 0$: linear cost (slope) of segment $s \in S_a$
- $f_a^s \geq 0$: fixed cost (intercept) of segment $s \in S_a$

Applications in transportation:

- Number of trucks used in TL (truckload) transportation
- Economies of scale in LTL (less-than-truckload) transportation
Basic multiple choice model

\( x_a^s \): total flow \( x_a \), if it falls in \((b_a^{s-1}, b_a^s]\); 0, otherwise

\( y_a^s \): 1, if \( x_a \) falls in \((b_a^{s-1}, b_a^s]\); 0, otherwise

\[ v(B) = \min \sum_{a \in A} \sum_{s \in S_a} (c_a^s x_a^s + f_a^s y_a^s) \]

subject to (1), (2)

\[ \sum_{k \in K} x_a^k = \sum_{s \in S_a} x_a^s, \quad a \in A, \]

\[ b_a^{s-1} y_a^s \leq x_a^s \leq b_a^s y_a^s, \quad a \in A, s \in S_a, \]

\[ \sum_{s \in S_a} y_a^s \leq 1, \quad a \in A, \]

\[ y_a^s \in \{0, 1\}, \quad a \in A, s \in S_a. \]
Flow disaggregation

- $M_a^k$: upper bound on the flow of commodity $k$ on arc $a$
- $x_a^{ks}$: the commodity flow $x_a^k$, if $x_a^s > 0$; 0, otherwise

\[
\sum_{k \in K} x_a^{ks} = x_a^s, \quad a \in A, s \in S_a,
\]

\[
\sum_{s \in S_a} x_a^{ks} = x_a^k, \quad a \in A, k \in K.
\]

- Add the extended forcing constraints:

\[
x_a^{ks} \leq M_a^k y_a^s, \quad a \in A, k \in K, s \in S_a.
\]
Extended multiple choice model

\[ v(E) = \min \sum_{a \in A} \sum_{s \in S_a} \left( c_a^s \sum_{k \in K} x_a^{ks} + f_a^s y_a^s \right) \]

\[ \sum_{a \in F_i} \sum_{s \in S_a} x_a^{ks} - \sum_{a \in B_i} \sum_{s \in S_a} x_a^{ks} = d_i^k, \quad i \in N, k \in K, \quad (3) \]

\[ b_a^{s-1} y_a^s \leq \sum_{k \in K} x_a^{ks} \leq b_a^s y_a^s, \quad a \in A, s \in S_a, \]

\[ 0 \leq x_a^{ks} \leq M_a^k y_a^s, \quad a \in A, k \in K, s \in S_a, \]

\[ \sum_{s \in S_a} y_a^s \leq 1, \quad a \in A, \quad (4) \]

\[ y_a^s \in \{0, 1\}, \quad a \in A, s \in S_a. \quad (5) \]
Lagrangian relaxation

- Relax flow conservation equations (3) in a Lagrangian way (Balakrishnan, Graves 1989)

- In the Lagrangian subproblem: $\sum_{k \in K} x_{a}^{ks} > 0$ iff $y_{a}^{s} = 1$

- Hence, first solve for each arc $a$ and segment $s$, the following continuous knapsack problem:

$$v(P_{a}^{s}(\pi)) = \min \sum_{k \in K} (c_{a}^{s} - \pi_{t(a)}^{k} + \pi_{h(a)}^{k})x_{a}^{ks}$$

$$b_{a}^{s-1} \leq \sum_{k \in K} x_{a}^{ks} \leq b_{a}^{s},$$

$$0 \leq x_{a}^{ks} \leq M_{a}^{k}, \quad k \in K.$$

- Then, solve a trivial IP subproblem in variables $y_{a}^{s}$ only:

$$v(R(\pi)) = \min \sum_{a \in A} \sum_{s \in S_{a}} (v(P_{a}^{s}(\pi)) + f_{a}^{s})y_{a}^{s}$$

subject to (4), (5).
Polyhedral results

- **Notation:**
  - $v(M)$: optimal value for model $M$
  - $F(M)$: feasible set for model $M$
  - $\text{conv}(F(M))$: convex hull of $F(M)$
  - $\overline{M}$: LP relaxation for model $M$
  - $R(\pi)$ : Lagrangian subproblem for Lagrange multipliers $\pi$
  - $D$: Lagrangian dual
    
    \begin{equation}
    v(D) = \max_{\pi} \{ v(R(\pi)) + \sum_{i \in N} \sum_{k \in K} \pi_i^k d_i^k \} 
    \end{equation}
  
  - $F(R(\pi)) = \text{conv}(F(R(\pi)))$
    
    (Croxton, Gendron, Magnanti 2007)
  
  - $v(B) \leq v(E) = v(D) \leq v(B)$
Appr0aches for computing $\nu(\overline{E})$

- Use state-of-the-art LP/MIP solver (with its additional cuts): hopeless for large $|K|$
- Solve $D$ by (subgradient optimization +) Dantzig-Wolfe decomposition (+ stabilization): interesting
- Use column-and-row generation to solve $E$ directly: structured Dantzig-Wolfe decomposition (SDW) (Frangioni, Gendron 2013)
Reformulation and decomposition

▶ “Structured” MIP:

\[(P) \quad \min_x \{ cx : Ax = b, \ x \in X \} \]

where \((R(\alpha))\) \[v(R(\alpha)) = \min_x \{ cx + \alpha(b - Ax) : x \in X \}\]

is “significantly easier” than \((P)\)

▶ Lagrangian dual:

\[(D) \quad \max_{\alpha} \{ v(R(\alpha)) \} = \min_x \{ cx : Ax = b, \ x \in conv(X) \}(\overline{P})\]

▶ Reformulation:

\[conv(X) = \{ x = C\theta : \Gamma\theta \leq \gamma \}\]

▶ Examples:
  ▶ Dantzig-Wolfe reformulation
  ▶ Extended multiple choice model \((E)\)
SDW assumptions

- Assumption 1 (*reformulation*):
  \[ \text{conv}(X) = \{ x = C\theta : \Gamma\theta \leq \gamma \} \]

- Assumption 2 (*padding with zeroes*):
  \[ \Gamma_B \bar{\theta}_B \leq \gamma_B \Rightarrow \Gamma [\bar{\theta}_B, 0] \leq \gamma \]
  \[ \Rightarrow X_B = \{ x = C_B\theta_B : \Gamma_B\theta_B \leq \gamma_B \} \subseteq \text{conv}(X) \]

- Assumption 3 (*easy update of variables and constraints*):
  Given \( B, \bar{x} \in \text{conv}(X) \) s.t. \( \bar{x} \notin X_B \),
  it is "easy" to find \( B' \supset B \) and \( \Gamma_{B'}, \gamma_{B'} \) such that
  \( \exists B'' \supseteq B' \) such that \( \bar{x} \in X_{B''} \).
SDW algorithm

\[
\langle \text{initialize } B \rangle; \\
\text{repeat} \\
\langle \text{solve } (\bar{P}_B) \text{ for } \tilde{x}, \tilde{\alpha}; \tilde{v} = c\tilde{x} \rangle; \\
\tilde{x} = \arg\min_x \{ (c - \tilde{\alpha}A)x : x \in X \}; \quad /* (R(\tilde{\alpha})) */ \\
\text{if } (\tilde{v} = c\tilde{x} + \tilde{\alpha}(b - A\tilde{x})) \\
\quad \text{then STOP; } \quad /* \tilde{x} \text{ optimal } */ \\
\quad \text{else } \langle \text{update } B \text{ as in Assumption 3} \rangle; \\
\text{until } \sim \text{ STOP}
\]

- Finitely terminates with an optimal solution of \( \bar{P} \)
- ...even if (proper) removal of indices from \( B \) is allowed
Variant 1 of SDW for computing $\nu(\overline{E})$

- Initialization: solve $\overline{B}$; $\nu^{last} = \nu(\overline{B})$

- Column generation iterations:
  - For each arc $a$:
    1) let $s^* = \arg \min_{s \in S_a} \{ \nu(P^s_a(\pi)) + f^s_a \}$;
    2) if $\nu(P^s_a(\pi)) + f^s_a < 0$, generate variable $y^s_{a^*}$ and, for each $k \in K$, variables $x^{ks}_{a^*}$. If columns have been generated, solve $\overline{E_B}$ and iterate to generate more columns
  - If $\nu(\overline{E_B}) - \nu^{last} < \epsilon$, STOP; else, $\nu^{last} = \nu(\overline{E_B})$

- Row generation iterations:
  - Add violated extended forcing constraints
  - If rows have been generated, solve $\overline{E_B}$ and iterate to generate more rows
  - Go to column generation iterations
Equivalent formulation for $\bar{E}$

$$v(\bar{E}) = \min \sum_{a \in A} \sum_{s \in S_a} \left( c_a^s \sum_{k \in K} x_{a}^{ks} + f_a^s y_s^a \right)$$

$$\sum_{a \in F; s \in S_a} x_{a}^{ks} - \sum_{a \in B; s \in S_a} x_{a}^{ks} = d_i^k, \quad i \in N, k \in K, \quad (\pi_i^k)$$

$$\sum_{k \in K} x_{a}^{ks} \leq b_a^s y_s^a, \quad a \in A, s \in S_a, \quad (\alpha_a^s)$$

$$0 \leq x_{a}^{ks} \leq M_a^k y_s^a, \quad a \in A, k \in K, s \in S_a,$$

$$\sum_{s \in S_a} y_s^a \leq 1, \quad a \in A, \quad (\gamma_a)$$

$$y_s^a \geq 0, \quad a \in A, s \in S_a.$$
Variant 2 of SDW for computing $\nu(\bar{E})$

- Identical to Variant 1, except for column generation iterations
- For each arc $a$:
  1) let $s^* = \arg \min_{s \in S_a} \{ f_a^s(\pi, \alpha) \}$, where $f_a^s(\pi, \alpha) = f_a^s - b_a^s \alpha_a^s + \sum_{k \in K} c_a^{ks}(\pi, \alpha)$ and $c_a^{ks}(\pi, \alpha) = \min\{0, c_a^s - \pi_k t(a) + \pi_k h(a) + \alpha_a^s\}$;
  2) if $f_a^{s*}(\pi, \alpha) < -\gamma_a$, then, for each $k$ such that $c_a^{ks*}(\pi, \alpha) < 0$, generate variables $x_a^{ks*}$;
  3) for each $s$ such that $y_a^s > 0$, for each $k$ such that $c_a^{ks}(\pi, \alpha) < 0$, generate variables $x_a^{ks}$
- This column generation method corresponds to the application of SDW based on the **Lagrangian relaxation of flow conservation equations and capacity constraints**
Implementation and data instances

- **Network design** instances ($|K| = 40, 100, 200, 400)$:
  - Commodity $k =$ O-D pair with demand $d^k$
  - $b^s_a = sb_a$, $f^s_a = sf_a$, $s = 0, 1, \ldots, \lceil \sum_k d^k / b_a \rceil$
  - $c^s_x^s_a$ replaced by $\sum_{k \in K} c^k_a x^k_a$

- All IPs and LPs solved with CPLEX 12.4

- $B$ and $E$ solved with CPLEX

- Variant 1 (Frangioni, Gendron 2009): solving the root relaxation, then freezing the formulation + CPLEX polishing for one hour

- Variant 2 (El Filali 2014): implemented with the B&P&C in SCIP 3.0.1 (Achterberg 2009)

- gap = final gap (%), cpu = time (limit = 3h)
Results at the root

Best known upper bound for each instance used to compute gaps
No lower bound after 3h: gap = 100%

<table>
<thead>
<tr>
<th>K</th>
<th>B</th>
<th>E</th>
<th>SDW1</th>
<th>SDW2</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>2.4, 221</td>
<td>0.2, 14</td>
<td>0.4, 0</td>
<td>0.4, 3</td>
</tr>
<tr>
<td>100</td>
<td>16.3, 10751</td>
<td>25.6, 4095</td>
<td>1.8, 181</td>
<td>1.8, 229</td>
</tr>
<tr>
<td>200</td>
<td>22.4, 10781</td>
<td>88.2, 10782</td>
<td>5.7, 1711</td>
<td>5.7, 674</td>
</tr>
<tr>
<td>400</td>
<td>31.7, 9049</td>
<td>100.0, 10683</td>
<td>6.8, 10862</td>
<td>6.8, 7159</td>
</tr>
</tbody>
</table>

Table: Lower bounds at the root: gap (%), cpu (s)
Comparing the two SDW variants

No initial upper bound

Unlike SDW2, solved by B&P&C, frozen formulations in SDW1 implementation may not contain optimal solution

| $|K|$ | SDW1 | SDW2 |
|-----|------|------|
| 40  | 0.4, 0 | 0.0, 182 |
| 100 | 1.8, 1551 | 1.2, 9050 |
| 200 | 5.7, 5174 | 4.7, 10800 |
| 400 | 6.8, 11250 | 6.8, 10803 |

Table: SDW1 versus SDW2: gap (%), cpu (s)
Conclusions

- Frangioni, Gendron (2009) show SDW1 is more efficient than:
  - Solving $D$ with an aggregated bundle method
  - Solving the special case of network design by a cutting-plane method based on residual capacity inequalities (Magnanti, Mirchandani, Vachani 1993; Atamtürk, Rajan 2002)

- SDW2 is as effective as SDW1 and more efficient when $|K|$ gets larger

- Forthcoming:
  - SDW3: generate both $y$ and $x$ variables simultaneously
  - Compare the three variants under the same B&P&C interface
  - Add stabilization techniques to speedup column generation (Frangioni, Gendron 2013)