One of Bernard’s life-long (scientific) love stories: playing ping-pong between (multicommodity flow) models and (decomposition) algorithms

Antonio Frangioni  Bernard Gendron  Enrico Gorgone

Dipartimento di Informatica, Università di Pisa

Hommage au Professeur Bernard Gendron

Montreal, February the 23rd, 2023
It all started with the classical Multicommodity flow model

- Graph $G = (N, A)$, classical Multicommodity flow model
  \[
  \begin{align*}
  \min & \sum_{k \in K} \sum_{(i,j) \in A} c_{ij}^k x_{ij}^k \\
  \text{s.t.} & \sum_{(i,j) \in A} x_{ij}^k - \sum_{(j,i) \in A} x_{ji}^k = b_i^k, \quad i \in N, k \in K \\
  & \sum_{k \in K} x_{ij}^k \leq u_{ij}, \quad (i,j) \in A \\
  & 0 \leq x_{ij}^k \leq u_{ij}, \quad (i,j) \in A, k \in K
  \end{align*}
  \]  

- Often $b_i^k \equiv (s^k, t^k, d^k)$, i.e., commodities $K \equiv$ O-D pairs, possibly with $x_{ij} \rightarrow d^k x_{ij}$, $x_{ij} \in \{0, 1\}$ (unsplittable routing)

- Pervasive structure in logistic and transportation, often very large (time-space $\Rightarrow$ acyclic) $G$, “few” commodities

- Common in many other areas (telecommunications, energy, . . . ), possibly “small” (undirected) $G$, “many” commodities

- Interesting links with many hard problems (e.g. Max-Cut)

- “Hard” even if continuous: very-large-scale LPs
The paradise of decomposition

- Many sources of structure $\implies$ the paradise of decomposition\(^1,2\)
- Lagrangian relaxation\(^3\) of linking constraints:
  - (3) $\implies$ flow (shortest path) relaxation
  - (2) $\implies$ knapsack relaxation
  - others possible (will see)
- Benders’ decomposition\(^4\) of linking variables:
  - Linking variables can be artificially added (resource decomposition)\(^5\)
    \[
    x^k_{ij} \leq u^k_{ij}, \quad \sum_{k \in K} u^k_{ij} \leq u_{ij}
    \]
- I did mostly Lagrange, but many ideas can be applied to Benders\(^6\) and Bernard did work on Benders (for network design, will see)\(^7\)

---

6. van Ackooij, F., de Oliveira “Inexact Stabilized Benders’ Decomposition Approaches, with Application [. . .]” *CO&A*, 2016
The general form of structure we consider:

\[
\max \{ cx : Ax = b, \ x \in X \}
\]

\(Ax = b\) “complicating” \(\equiv\) optimizing upon \(X\) “easy” \(\equiv\) convex

Almost always \(X = \bigotimes_{h \in \mathcal{K}} X^h (\mathcal{K} \neq \mathcal{K}) \equiv Ax = b\) linking constraints

Our \(X\) compact, represent it by vertices (otherwise just add extreme rays)

\[
X = \{ x = \sum_{\tilde{x} \in X} \tilde{x}\theta_{\tilde{x}} : \sum_{\tilde{x} \in X} \theta_{\tilde{x}} = 1, \ \theta_{\tilde{x}} \geq 0 \ \ \tilde{x} \in X \}
\]

\(\Rightarrow\) Dantzig-Wolfe reformulation\(^2\) of \((\Pi)\):

\[
\begin{align*}
\max & \quad c \left( \sum_{\tilde{x} \in X} \tilde{x}\theta_{\tilde{x}} \right) \\
A \left( \sum_{\tilde{x} \in X} \tilde{x}\theta_{\tilde{x}} \right) & = b \\
\sum_{\tilde{x} \in X} \theta_{\tilde{x}} & = 1, \ \theta_{\tilde{x}} \geq 0 \ \ \tilde{x} \in X
\end{align*}
\]

\(X\) nonconvex \(\Rightarrow\) solving “best” convex relaxation

\[
\max \{ cx : Ax = b, \ x \in \text{conv}(X) \}
\]
D-W decomposition $\equiv$ Lagrangian relaxation

- $B \subset X \text{ (small)},$ solve master problem restricted to $B$
  \[
  \begin{align*}
  (\Pi_B) \quad & \max \left\{ c x : A x = b, \quad x \in \text{conv}(B) \right\} \\
  \end{align*}
  \]

- feed (partial) dual optimal solution $\lambda^*$ (of $A x = b$) to pricing problem
  \[
  \begin{align*}
  (\Pi_{\lambda^*}) \quad & \max \left\{ (c - \lambda^* A)x : x \in X \right\} \quad [ + \lambda^* b ]
  \end{align*}
  \]

(Lagrangian relaxation), optimal solution $\bar{x}$ of $(\Pi_{\lambda^*}) \to B$

- Dual: $(\Delta_B) \quad \min \left\{ f_B(\lambda) = \max \left\{ c x + \lambda(b - Ax) : x \in B \right\} \right\}$

- $f_B = \text{lower approximation of "true" Lagrangian function}$
  \[
  f(\lambda) = \max \left\{ c x + \lambda(b - Ax) : x \in X \right\}
  \]
  \[
  \equiv (\Delta_B) \text{ outer approximation of Lagrangian dual } \equiv (\Pi)
  \]
  \[
  \begin{align*}
  (\Delta) \quad & \min \left\{ f(\lambda) = \max \left\{ c x + \lambda(b - Ax) : x \in X \right\} \right\} \quad (6)
  \end{align*}
  \]

- Dantzig-Wolfe decomposition $\equiv$ Cutting Plane approach to $(\Delta)^8$

---

All well and nice, but does it work well?
All well and nice, but does it work well?

- By-the-book? **Not really**

A mysterious threshold is hit and “real” convergence begins

≡ having good initial point not much useful

\[ \text{\( \lambda \) immediately shoots much farther from optimum than initial point} \]
All well and nice, but does it work well?

- By-the-book? Not really

- $\lambda^*$ immediately shoots much farther from optimum than initial point
  $\equiv$ having good initial point not much useful

- No apparent improvement for a long time as information slowly accrues

- A mysterious threshold is hit and “real” convergence begins
How to deal with instability

- $\lambda_{k+1}^*$ can be very far from $\lambda_k^*$, where $f_B$ is a "bad model" of $f$

- If $\{\lambda_k^*\}$ is unstable, then stabilize it around stability centre $\bar{\lambda}$

- Stabilizing term $D_t$ with parameter $t$, stabilized master problems

$$
\begin{align*}
(\Delta _B,\bar{\lambda},D_t) & \min \left\{ f_B(\bar{\lambda} + d) + D_t(d) \right\} \\
(\Pi _B,\bar{\lambda},D_t) & \max \left\{ cx + \bar{\lambda}(b - Ax) - D_t^*(Ax - b) : x \in \text{conv}(B) \right\}
\end{align*}
$$

("*" = Fenchel’s conjugate): a generalized augmented Lagrangian

- Change $\bar{\lambda}$ when $f(\bar{\lambda} + d^*) \ll f(\bar{\lambda})$, appropriate $D \Longrightarrow$ converges\(^9\)

- Choosing $t$ nontrivial

- Aggregation trick: right $D \Longrightarrow$ still converges with “poorman bundle” $B = \{x^*\}$ (although rather slowly\(^10\) $\approx$ volume\(^11\) $\equiv$ subgradient)

---

\(^9\) F. “Generalized Bundle Methods” *SIOPT*, 2002


What is an appropriate stabilization?

- Simplest: $D_t \equiv \|d\|_\infty \leq t$, $D_t^* = t\|\cdot\|_2^2$ (“boxstep”)

- Better: $D_t = \frac{1}{2t}\|\cdot\|_2^2$, $D_t^* = \frac{1}{2}t\|\cdot\|_2^2$ (may use specialized QP solvers)

- Keep LP master: \textit{piecewise-linear approximations}

- Several other ideas (level stabilization, centres, better “Hessian”, . . .)

---

12 Marsten, Hogan, Blankenship “The Boxstep Method for Large-scale Optimization” OR, 1975


All well and nice, but does it work well?

--

17 Nemirovsky, Yudin “Problem Complexity and Method Efficiency in Optimization” Wiley, 1983
All well and nice, but does it work well?

- It depends on what “well” means, but **surely better**

Black-box nonsmooth optimization is $\Omega(\frac{1}{\varepsilon^2})$ in general\(^{17}\)

- **Convergence slow-ish** (but at least some) until mysterious threshold hit

- At least, **better information accrued sooner** $\implies$ “quick tail” starts sooner

- **Can** make a huge difference in applications

\(^{17}\) Nemirovsky, Yudin “Problem Complexity and Method Efficiency in Optimization” Wiley, 1983
Indeed, it worked well enough for Multicommodity flows

<table>
<thead>
<tr>
<th>(k)</th>
<th>(n)</th>
<th>(m)</th>
<th>(b)</th>
<th>(Size)</th>
<th>MMCFB</th>
<th>Cplex</th>
<th>PPRN</th>
<th>IPM</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>64</td>
<td>362</td>
<td>148</td>
<td>1.4e+3</td>
<td>0.07</td>
<td>0.22</td>
<td>0.13</td>
<td>1.44</td>
</tr>
<tr>
<td>8</td>
<td>64</td>
<td>371</td>
<td>183</td>
<td>3.0e+3</td>
<td>0.26</td>
<td>0.50</td>
<td>0.52</td>
<td>4.26</td>
</tr>
<tr>
<td>16</td>
<td>64</td>
<td>356</td>
<td>191</td>
<td>5.7e+3</td>
<td>1.08</td>
<td>2.01</td>
<td>3.41</td>
<td>16.03</td>
</tr>
<tr>
<td>32</td>
<td>64</td>
<td>362</td>
<td>208</td>
<td>1.2e+4</td>
<td>3.42</td>
<td>12.99</td>
<td>22.04</td>
<td>43.27</td>
</tr>
<tr>
<td>64</td>
<td>64</td>
<td>361</td>
<td>213</td>
<td>2.3e+4</td>
<td>8.53</td>
<td>115.99</td>
<td>147.10</td>
<td>114.19</td>
</tr>
<tr>
<td>4</td>
<td>128</td>
<td>694</td>
<td>293</td>
<td>2.8e+3</td>
<td>0.58</td>
<td>0.54</td>
<td>0.85</td>
<td>6.45</td>
</tr>
<tr>
<td>8</td>
<td>128</td>
<td>735</td>
<td>363</td>
<td>5.9e+3</td>
<td>2.57</td>
<td>1.81</td>
<td>4.79</td>
<td>26.32</td>
</tr>
<tr>
<td>16</td>
<td>128</td>
<td>766</td>
<td>424</td>
<td>1.2e+4</td>
<td>11.30</td>
<td>17.31</td>
<td>40.57</td>
<td>116.26</td>
</tr>
<tr>
<td>32</td>
<td>128</td>
<td>779</td>
<td>445</td>
<td>2.5e+4</td>
<td>27.72</td>
<td>212.09</td>
<td>503.48</td>
<td>346.91</td>
</tr>
<tr>
<td>64</td>
<td>128</td>
<td>784</td>
<td>469</td>
<td>5.0e+4</td>
<td>44.04</td>
<td>1137.05</td>
<td>2215.48</td>
<td>719.69</td>
</tr>
<tr>
<td>128</td>
<td>128</td>
<td>808</td>
<td>485</td>
<td>1.0e+5</td>
<td>52.15</td>
<td>5816.54</td>
<td>6521.94</td>
<td>1546.91</td>
</tr>
<tr>
<td>4</td>
<td>256</td>
<td>1401</td>
<td>570</td>
<td>5.6e+3</td>
<td>7.54</td>
<td>2.38</td>
<td>9.88</td>
<td>51.00</td>
</tr>
<tr>
<td>8</td>
<td>256</td>
<td>1486</td>
<td>743</td>
<td>1.2e+4</td>
<td>25.09</td>
<td>15.48</td>
<td>105.89</td>
<td>208.10</td>
</tr>
<tr>
<td>16</td>
<td>256</td>
<td>1553</td>
<td>854</td>
<td>2.5e+4</td>
<td>60.85</td>
<td>180.06</td>
<td>955.20</td>
<td>844.09</td>
</tr>
<tr>
<td>32</td>
<td>256</td>
<td>1572</td>
<td>907</td>
<td>5.0e+4</td>
<td>107.54</td>
<td>1339.46</td>
<td>6605.45</td>
<td>1782.47</td>
</tr>
<tr>
<td>64</td>
<td>256</td>
<td>1573</td>
<td>931</td>
<td>1.0e+5</td>
<td>144.75</td>
<td>7463.14</td>
<td>18467.73</td>
<td>3441.62</td>
</tr>
<tr>
<td>128</td>
<td>256</td>
<td>1581</td>
<td>932</td>
<td>2.0e+5</td>
<td>223.13</td>
<td>35891.37</td>
<td>61522.94</td>
<td>9074.31</td>
</tr>
<tr>
<td>256</td>
<td>256</td>
<td>1503</td>
<td>902</td>
<td>3.8e+5</td>
<td>445.81</td>
<td>110897+</td>
<td>187156+</td>
<td>17279.00</td>
</tr>
</tbody>
</table>

- We could handily beat the state-of-the-art Cplex 3.0 and others\(^{18}\)

---

18 F., Gallo “A Bundle Type Dual-Ascent Approach to Linear Multicommodity Min Cost Flow Problems” *IJOC*, 1999

Indeed, it worked well enough for Multicommodity flows

<table>
<thead>
<tr>
<th>Group</th>
<th>$T_1$</th>
<th>$s%$</th>
<th>$T_4$</th>
<th>$T_{16}$</th>
<th>$T_{64}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>64-64</td>
<td>21.31</td>
<td>1.10</td>
<td>5.98</td>
<td>2.08</td>
<td>1.00</td>
</tr>
<tr>
<td>128-64</td>
<td>123.66</td>
<td>1.25</td>
<td>35.70</td>
<td>13.16</td>
<td>7.01</td>
</tr>
<tr>
<td>128-128</td>
<td>159.78</td>
<td>0.66</td>
<td>42.04</td>
<td>12.65</td>
<td>4.95</td>
</tr>
<tr>
<td>256-64</td>
<td>466.35</td>
<td>1.51</td>
<td>129.75</td>
<td>44.69</td>
<td>21.89</td>
</tr>
<tr>
<td>256-128</td>
<td>718.35</td>
<td>0.62</td>
<td>188.96</td>
<td>57.23</td>
<td>22.99</td>
</tr>
<tr>
<td>256-256</td>
<td>1404.48</td>
<td>0.30</td>
<td>348.46</td>
<td>98.30</td>
<td>33.85</td>
</tr>
<tr>
<td>512-512</td>
<td>15898.89</td>
<td>0.22</td>
<td>*</td>
<td>1025.26</td>
<td>291.40</td>
</tr>
</tbody>
</table>

- We could handily beat the state-of-the-art Cplex 3.0 and others\(^{18}\)
- We could even parallelise on a supercomputer with a whopping 64 CPU\(^{19}\)

---

\(^{18}\) F., Gallo “A Bundle Type Dual-Ascent Approach to Linear Multicommodity Min Cost Flow Problems” *IJoC*, 1999

\(^{19}\) Cappanera, F. “[...] Parallelization of a Cost-Decomposition Algorithm for Multi-Commodity Flow Problems” *IJoC*, 2003
Indeed, it worked well enough for Multicommodity flows

We could handily beat the state-of-the-art Cplex 3.0 and others\(^{18}\)

We could even parallelise on a supercomputer with a whopping 64 CPU\(^{19}\)

But this was not enough for Bernard . . .

\(^{18}\) F., Gallo “A Bundle Type Dual-Ascent Approach to Linear Multicommodity Min Cost Flow Problems” IJoC, 1999

\(^{19}\) Cappanera, F. “[. . . ] Parallelization of a Cost-Decomposition Algorithm for Multi-Commodity Flow Problems” IJoC, 2003

<table>
<thead>
<tr>
<th>Group</th>
<th>(T_1)</th>
<th>(s%)</th>
<th>(T_4)</th>
<th>(T_{16})</th>
<th>(T_{64})</th>
</tr>
</thead>
<tbody>
<tr>
<td>64-64</td>
<td>21.31</td>
<td>1.10</td>
<td>5.98</td>
<td>2.08</td>
<td>1.00</td>
</tr>
<tr>
<td>128-64</td>
<td>123.66</td>
<td>1.25</td>
<td>35.70</td>
<td>13.16</td>
<td>7.01</td>
</tr>
<tr>
<td>128-128</td>
<td>159.78</td>
<td>0.66</td>
<td>42.04</td>
<td>12.65</td>
<td>4.95</td>
</tr>
<tr>
<td>256-64</td>
<td>466.35</td>
<td>1.51</td>
<td>129.75</td>
<td>44.69</td>
<td>21.89</td>
</tr>
<tr>
<td>256-128</td>
<td>718.35</td>
<td>0.62</td>
<td>188.96</td>
<td>57.23</td>
<td>22.99</td>
</tr>
<tr>
<td>256-256</td>
<td>1404.48</td>
<td>0.30</td>
<td>348.46</td>
<td>98.30</td>
<td>33.85</td>
</tr>
<tr>
<td>512-512</td>
<td>15898.89</td>
<td>0.22</td>
<td>*</td>
<td>1025.26</td>
<td>291.40</td>
</tr>
</tbody>
</table>
...for he wanted to solve Multicommodity Network Design

\[
\begin{align*}
\min & \sum_{k \in K} \sum_{(i,j) \in A} c_{ij}^k x_{ij}^k + \sum_{(i,j) \in A} f_{ij} y_{ij} \\
\text{s.t.} & \sum_{(i,j) \in A} x_{ij}^k - \sum_{(j,i) \in A} x_{ji}^k = b_i^k & i \in N, k \in K \\
& \sum_{k \in K} x_{ij}^k \leq u_{ij} y_{ij} & (i,j) \in A \\
& 0 \leq x_{ij}^k \leq u_{ij}^k y_{ij} & (i,j) \in A, k \in K \\
& y \in Y \subseteq \{0, 1\}^m
\end{align*}
\]

- Reasonably good bounds but only with strong forcing constraints (9)
- Just one more subproblem, but a lot more constraints (9) to relax \(\equiv\) much larger dual space (harder) and much more costly master problem
- In fact, relaxing (2) (knapsack relaxation) competitive: less multipliers (but unconstrained), still (arc) decomposable if \(Y = \{0, 1\}^m\)
- Flow relaxation requires dynamic bundle methods\(^{20}\), many other uses\(^{21}\)

\(^{21}\) F., Lodi, Rinaldi “New approaches for optimizing over the semimetric polytope” Math. Prog., 2005
Which worked well, sort of

<table>
<thead>
<tr>
<th>Problems</th>
<th>CPXW</th>
<th>WB</th>
<th>CPXS</th>
<th>SS</th>
<th>SB</th>
<th>KS</th>
<th>KB</th>
</tr>
</thead>
<tbody>
<tr>
<td>25,100,10</td>
<td>2.3e-1</td>
<td>2.3e-1</td>
<td>0.0</td>
<td>5.3e-4</td>
<td>1.8e-4</td>
<td>7.6e-4</td>
<td>2.7e-4</td>
</tr>
<tr>
<td>(3)</td>
<td>0.1</td>
<td>0.0</td>
<td>1.3</td>
<td>1.1</td>
<td>1.0</td>
<td>0.6</td>
<td>0.8</td>
</tr>
<tr>
<td>25,100,30</td>
<td>2.2e-1</td>
<td>2.2e-1</td>
<td>0.0</td>
<td>4.0e-4</td>
<td>1.4e-4</td>
<td>9.8e-4</td>
<td>5.5e-4</td>
</tr>
<tr>
<td>(3)</td>
<td>0.6</td>
<td>0.2</td>
<td>11.3</td>
<td>3.0</td>
<td>3.5</td>
<td>1.2</td>
<td>2.3</td>
</tr>
<tr>
<td>100,400,10</td>
<td>2.8e-1</td>
<td>2.8e-1</td>
<td>0.0</td>
<td>1.1e-3</td>
<td>6.7e-4</td>
<td>1.6e-3</td>
<td>1.3e-3</td>
</tr>
<tr>
<td>(3)</td>
<td>0.3</td>
<td>0.1</td>
<td>35.9</td>
<td>4.2</td>
<td>4.8</td>
<td>1.7</td>
<td>3.0</td>
</tr>
<tr>
<td>100,400,30</td>
<td>2.9e-1</td>
<td>2.9e-1</td>
<td>0.0</td>
<td>1.0e-3</td>
<td>1.1e-3</td>
<td>1.9e-3</td>
<td>2.5e-3</td>
</tr>
<tr>
<td>(3)</td>
<td>5.9</td>
<td>2.3</td>
<td>351.9</td>
<td>14.3</td>
<td>16.7</td>
<td>4.5</td>
<td>9.1</td>
</tr>
</tbody>
</table>

Issue: > 10-100 subgradients filled our mighty 64Mb (not a typo) of RAM

⇒ never really got to the “fast tail” convergence

Yet bundle competitive with subgradient, flow and knapsack traded blows, 1e−5 to 1e−3 accuracy good enough for a B&B

Could have been better, still my most cited article ever

---

But Bernard was not happy, so we kept pushing

- **Dantzig-Wolfe reformulation**: 
  \[ S = \{ \text{(extreme) flows } s = [\bar{x}^{1,s}, \ldots, \bar{x}^{k,s}] \} \]
  \[
  \begin{align*}
  \min & \sum_{s \in S} \left( \sum_{k \in K} \sum_{(i,j) \in A} c_{ij}^{k} \bar{x}_{ij}^{k,s} \right) \theta_s \\
  \sum_{s \in S} \left( \sum_{k \in K} \bar{x}_{ij}^{k,s} - u_{ij} \right) \theta_s & \leq 0 \quad (i,j) \in A \\
  \sum_{s \in S} \theta_s & = 1 \quad, \quad \theta_s \geq 0 \quad s \in S
  \end{align*}
  \]

- **Exploit separability**: 
  \[ X = X^1 \times X^2 \times \ldots \times X^{|K|} \implies \]
  \[ \text{conv}(X) = \text{conv}(X^1) \times \text{conv}(X^2) \times \ldots \times \text{conv}(X^{|K|}) \implies \]
  a different \( \theta_k^s \) for each \( \bar{x}^{k,s} \) (aggregated \( \equiv \theta_s^k = \theta_s^h, h \neq k \), innatural)

- **Simple scaling leads to arc-path formulation** (in O-D case):
  \[ p \in P^k = \{ s^k-t^k \text{ paths } \}, \quad c_p \text{ cost}, \quad f_p(= d^k \theta_s^k) \text{ flow}, \quad P = \bigcup_{k \in K} P^k \]
  \[
  \begin{align*}
  \min & \sum_{p \in P} c_p f_p \\
  \sum_{p \in P : (i,j) \in p} f_p & \leq u_{ij} \quad (i,j) \in A \\
  \sum_{p \in P^k} f_p & = d^k \quad k \in K \\
  f_p & \geq 0 \quad p \in P
  \end{align*}
  \]
Disaggregated formulation: more columns but sparser, more rows

Master problem size $\approx$ time increases, but convergence speed increases $\equiv$ consistent improvement if you have enough RAM

Much more efficient for Multicommodity Flows$^{24}$ and others$^{25}$

But not for Network Design! So we had to understand why


$^{25}$Borghetti, F., Lacalandra, Nucci “Lagrangian Heuristics Based on Disaggregated Bundle […]” *IEEE TPWRS*, 2003
How not to do disaggregated decomposition

$S$ = extreme points of $y$ ($2^{|A|}$ vertices of the unitary hypercube):

$$\min \sum_{p \in \mathcal{P}} c_p f_p + \sum_{s \in S} \left( \sum_{(i,j) \in A} f_{ij} \bar{y}_{ij}^s \right) \theta_s$$

$$\sum_{p \in \mathcal{P} : (i,j) \in p} f_p \leq u_{ij} \sum_{s \in S} \bar{y}_{ij}^s \theta_s \quad (i,j) \in A$$

$$\sum_{p \in \mathcal{P}^k} f_p = d^k \quad k \in K$$

$$f_p \geq 0 \quad p \in \mathcal{P}$$

$$\sum_{s \in S} \theta_s = 1 \quad , \quad \theta_s \geq 0 \quad s \in S$$
How not to do disaggregated decomposition

- \( S = \) extreme points of \( y \) (\( 2^|A| \) vertices of the unitary hypercube):

\[
\min \sum_{p \in P} c_p f_p + \sum_{s \in S} \left( \sum_{(i,j) \in A} f_{ij} \bar{y}^s_{ij} \right) \theta_s \\
\sum_{p \in P : (i,j) \in p} f_p \leq u_{ij} \sum_{s \in S} \bar{y}^s_{ij} \theta_s \quad (i,j) \in A \\
\sum_{p \in \mathcal{P}^k} f_p = d^k \\
f_p \geq 0 \\
\sum_{s \in S} \theta_s = 1 \quad , \quad \theta_s \geq 0 \\
\]

- Is this sane? Arguably not: replacing a \( 2n \) formulation with a \( 2^n \) one!

- The problem on \( y \) variables is too easy, do not D-W it
How not to do disaggregated decomposition

$S = \text{extreme points of } y \ (2^{|A|} \text{ vertices of the unitary hypercube}):$

$$\min \sum_{p \in P} c_p f_p + \sum_{s \in S} \left( \sum_{(i,j) \in A} f_{ij} \bar{y}_{ij}s \right) \theta_s$$

$$\sum_{p \in P} : (i,j) \in p \ f_p \leq u_{ij} \sum_{s \in S} \bar{y}_{ij}s \theta_s \quad (i,j) \in A$$

$$\sum_{p \in P^k} f_p = d_k \quad k \in K$$

$$f_p \geq 0 \quad p \in P$$

$$\sum_{s \in S} \theta_s = 1 \ , \ \theta_s \geq 0 \quad s \in S$$

Is this sane? Arguably not: replacing a $2n$ formulation with a $2^n$ one!

The problem on $y$ variables is too easy, do not D-W it

Or D-W it more: $\{0, 1\}^m$ is a Cartesian product: why not $S^{ij} = \{0, 1\}$?

$y_{ij} \rightarrow 0 \cdot \theta^{ij,0} + 1 \cdot \theta^{ij,1} , \ \theta^{ij,0} + \theta^{ij,1} = 1 , \ \theta^{ij,0} \geq 0 , \ \theta^{ij,1} \geq 0$

$y_{ij} \in [0, 1]$
How not to do disaggregated decomposition

- $S = \text{extreme points of } y \ (2^{|A|} \ \text{vertices of the unitary hypercube}):$
  \begin{align*}
  &\min \sum_{p \in \mathcal{P}} c_p f_p + \sum_{s \in S} \left( \sum_{(i,j) \in A} f_{ij} \tilde{y}_{ij}^s \right) \theta_s \\
  &\sum_{p \in \mathcal{P}} : (i,j) \in p \ f_p \leq u_{ij} \sum_{s \in S} \tilde{y}_{ij}^s \theta_s \quad \quad (i,j) \in A \\
  &\sum_{p \in \mathcal{P}_k} f_p = d_k \quad \quad \quad \quad \quad \quad \quad \quad \quad k \in K \\
  &f_p \geq 0 \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad p \in \mathcal{P} \\
  &\sum_{s \in S} \theta_s = 1 \quad , \quad \theta_s \geq 0 \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad s \in S
  \end{align*}

- Is this sane? Arguably not: replacing a $2n$ formulation with a $2^n$ one!

- The problem on $y$ variables is too easy, do not D-W it

- Or D-W it more: $\{0, 1\}^m$ is a Cartesian product: why not $S^{ij} = \{0, 1\}$?

- $y_{ij} \rightarrow 0 \cdot \theta^{ij,0} + 1 \cdot \theta^{ij,1} \ , \ \theta^{ij,0} + \theta^{ij,1} = 1 \ , \ \theta^{ij,0} \geq 0 \ , \ \theta^{ij,1} \geq 0$

  \[ y_{ij} \in [0, 1] \quad \text{(no, \ldots really?!)} \]
How to do a disaggregated decomposition

- Arc-path formulation with original arc design variables

\[
\begin{align*}
\min & \quad \sum_{p \in P} c_p f_p + \sum_{(i, j) \in A} f_{ij} y_{ij} \\
\sum_{p \in P : (i, j) \in p} f_p & \leq u_{ij} y_{ij} \quad (i, j) \in A \\
\sum_{p \in P^k} f_p & = d_k \quad k \in K \\
f_p & \geq 0 \quad p \in P \\
y_{ij} & \in [0, 1] \quad (i, j) \in A
\end{align*}
\]

only generate the right variables, those that are too many

- But if one had (say) \( \sum_{(i, j) \in A} y_{ij} \leq r \): a linking constraint in \( Y \)

\[ \implies \] the design subproblem can no longer be disaggregated

- Yet, one could just add that constraint to the master problem

- Can this be stabilized? Of course it can\textsuperscript{26}

\textsuperscript{26}F., Gorgone “Bundle methods for sum-functions with “easy” components: [...] network design” Math. Prog., 2013
Stabilization with easy components

- Required structure: $X^1$ arbitrary, $X^2$ has compact convex formulation

\[ (\Pi) \max \left\{ c_1 x_1 + c_2(x_2) : x_1 \in X^1, \ G(x_2) \leq g, \ A_1 x_1 + A_2 x_2 = b \right\} \]

- Lagrangian function $f(\lambda) = f^1(\lambda) + f^2(\lambda)(-\lambda b)$, two components

- Primal master problem: “just plug in the easy set”

\[ (\Pi_B) \max \begin{cases} c_1 x_1 + c_2(x_2) \\ A_1 x_1 - A_2 x_2 = b \end{cases} \equiv \max \begin{cases} c_1 \left( \sum_{\bar{x}_1 \in B} \bar{x}_1 \theta_{\bar{x}_1} \right) + c_2(x_2) \\ A_1 \left( \sum_{\bar{x}_1 \in B} \bar{x}_1 \theta_{\bar{x}_1} \right) + A_2 x_2 = b \\ \sum_{\bar{x}_1 \in B} \theta_{\bar{x}_1} = 1, \ G(x_2) \leq g \end{cases} \]

- Dual master problem: $(\Delta_B) \min \left\{ \lambda b + f^1_B(\lambda) + f^2(\lambda) \right\}$

  i.e., insert “full” description of $f^2$ in the master problem

- Larger master problem at the beginning, but “perfect” information known

- Of course, stabilization + multiple easy/hard components . . .
All well and nice, but does it work well?

You bet, but you have to do it right: let information accumulate fast tail starts immediately if \(\geq 50,000\) subgradients + no harsh removals.

<table>
<thead>
<tr>
<th>Cplex</th>
<th>Easy</th>
<th>Aggregate</th>
<th>Volume</th>
<th>Dual</th>
<th>1e-6</th>
<th>1e-12</th>
<th>Time</th>
<th>It</th>
<th>Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>39</td>
<td>26</td>
<td>32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>56</td>
<td>294</td>
<td>5300</td>
<td>1e-6</td>
<td>6</td>
<td>871</td>
<td>8e-3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>26</td>
<td>5033</td>
<td>27231</td>
<td>1e-6</td>
<td>26</td>
<td>794</td>
<td>3e-3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>133</td>
<td>133</td>
<td>3122</td>
<td>14547</td>
<td>1e-6</td>
<td>51</td>
<td>760</td>
<td>4e-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>284</td>
<td>508</td>
<td>5475</td>
<td>15061</td>
<td>2e-5</td>
<td>29</td>
<td>869</td>
<td>1e-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>242</td>
<td>253</td>
<td>11863</td>
<td>13953</td>
<td>1e-6</td>
<td>109</td>
<td>765</td>
<td>2e-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>708</td>
<td>242</td>
<td>2e-5</td>
<td>58</td>
<td>817</td>
<td>2e-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2167</td>
<td>284</td>
<td>2e-5</td>
<td>131</td>
<td>2 3</td>
<td>344</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
All well and nice, but does it work well?

- You bet, but you have to do it right: let information accumulate
- Fast tail starts immediately if $\geq 50000$ subgradients + no harsh removals
All well and nice, but does it work well?

- You bet, but you have to do it **right**: let information accumulate

- **Fast tail starts immediately if** \( \geq 50000 \) subgradients **+ no harsh removals**

<table>
<thead>
<tr>
<th>Cplex dual</th>
<th>easy</th>
<th>aggregate</th>
<th>volume</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1e-6</td>
<td>1e-12</td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>26</td>
<td>32</td>
<td>322</td>
</tr>
<tr>
<td>132</td>
<td>28</td>
<td>56</td>
<td>294</td>
</tr>
<tr>
<td>301</td>
<td>21</td>
<td>26</td>
<td>5033</td>
</tr>
<tr>
<td>1930</td>
<td>133</td>
<td>133</td>
<td>3122</td>
</tr>
<tr>
<td>131</td>
<td>2</td>
<td>3</td>
<td>344</td>
</tr>
<tr>
<td>708</td>
<td>246</td>
<td>337</td>
<td>2256</td>
</tr>
<tr>
<td>2167</td>
<td>284</td>
<td>508</td>
<td>5475</td>
</tr>
<tr>
<td>8908</td>
<td>242</td>
<td>253</td>
<td>11863</td>
</tr>
</tbody>
</table>

- Much better accuracy/time than Cplex and competing decompositions

- Finally competitive even for Network Design, very happy

- Of course, meanwhile Barnard had already moved on
**Knapsack decomposition for Network Loading**

- **y general integers**, relax flow conservation constraints (2)

\[
\min \sum_{(i,j) \in A} \left( \sum_{k \in K} (d^k c^k_{ij} - \pi^k_i + \pi^k_j)x^k_{ij} + f_{ij}y_{ij} \right)
\]

\[
\sum_{k \in K} d^k x^k_{ij} \leq u_{ij}y_{ij} \quad (i,j) \in A
\]

\[
x^k_{ij} \in [0, 1] \quad (i,j) \in A, \ k \in K
\]

\[
y_{ij} \in \mathbb{N} \quad (i,j) \in A
\]

- **Decomposes by arc**, easy \((\approx 2 \text{ continuous knapsack})\) but **no integrality property** \(\implies\) **better bound** than continuous relaxation

- **Residual capacity inequalities**, separate \(\approx 2 \text{ continuous knapsack}\)^{27}

\[
a_k = d^k / u_{ij} \quad a(S) = \sum_{k \in S} a_k \quad S \subseteq K
\]

\[
\sum_{k \in S} a_k (1 - x^k_{ij}) \geq \left( a(S) - \lfloor a(S) \rfloor \right) \left( \lceil a(S) \rceil - y \right)
\]

\[
\bar{I}^+ = \text{continuous relaxation of (1)}-\text{(10)} + \text{(11)} \equiv \text{DW}^{28}
\]

---

^{27} \text{Atamtürk “On Capacitated Network Design Cut-Set Polyhedra” Math. Prog., 2002}

RG vs. StabDW, strange game: the only winning move . . .

- Large difficult instances, lightly ($C = 1$) to tightly ($C = 16$) capacitated

- Aggregated and/or non-stabilised DW too slow, only Stabilized DW “works” (but $\| \cdot \|_{\infty}$ stabilization, $\| \cdot \|_{2}$ too costly, see below)

| Problem | $|A|$ | $C$ | $\text{imp}$ | $I^+$ | $\text{StabDW}$ |
|---------|-----|-----|-------------|-------|----------------|
|         | cpu | it  | cpu         | it    |                |
|         |     |     |             |       |                |
| 229     | 185.17 | 18326 | 86 | 9261 | 132963 |
| 4       | 125.39 | 15537 | 80 | 11791 | 147879 |
| 8       | 85.31   | 9500  | 74 | 10702 | 146727 |
| 16      | 46.09   | 1900  | 52 | 7268  | 107197 |
| 287     | 198.87  | 14559 | 66 | 8815  | 120614 |
| 4       | 136.97  | 11934 | 62 | 8426  | 112308 |
| 8       | 92.94   | 9656  | 64 | 10098 | 130536 |
| 16      | 53.45   | 3579  | 54 | 6801  | 98972  |

- Trade blows depending on $C$, but basically both lose
Reformulation III: Binary formulation $B$

- Redundant upper bound constraints: $y_{ij} \leq \left\lceil \sum_{k \in K} d^k / a_{ij} \right\rceil = T_{ij}$

- Pseudo-polynomially many segments $S_{ij} = \{1, \ldots, T_{ij}\}$ for $y_{ij}$

- Reformulation in binary variables: $y_{ij} = \sum_{s \in S_{ij}} s y_{ij}^s$ (substituted away)

  \[
  y_{ij}^s = \begin{cases} 
  1 & \text{if } y_{ij} = s \\
  0 & \text{otherwise}
  \end{cases} \quad s \in S_{ij}
  \]

  \[
  x_{ij}^{ks} = \begin{cases} 
  x_{ij}^k & \text{if } y_{ij} = s \\
  0 & \text{otherwise}
  \end{cases} \quad s \in S_{ij}, \ k \in K
  \]

- $$(s - 1)a_{ij}y_{ij}^s \leq \sum_{k \in K} d^k x_{ij}^{ks} \leq sa_{ij}y_{ij}^s \quad (i, j) \in A, \ s \in S_{ij}$$

- $$\sum_{s \in S_{ij}} y_{ij}^s \leq 1 \quad (i, j) \in A$$

- + extended linking inequalities $x_{ij}^{ks} \leq y_{ij}^s \quad (i, j) \in A, \ k \in K, \ s \in S_{ij}$

  \[\implies B+ \text{ same bound as } \bar{T}+ \text{ and } \text{DW}^{29}\]

---

$^{29}$F., Gendron “0-1 reformulations of the multicommodity capacitated network design problem” DAM, 2009
In fact, binary formulation describes $\text{conv}(X^{ij}) \equiv$ integrality property
$\implies$ optimizing over $X \implies \text{conv}(X)$ easy

Pseudo-polynomial number of variables and constraints

Substantially different from both RG and DW

Need to generate both rows and columns: how?
The Structured Dantzig-Wolfe Idea

- **Assumption 1** (alternative (large) Formulation of “easy” set)
  \[ \text{conv}(X) = \left\{ x = C\theta : \Gamma\theta \leq \gamma \right\} \]

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

- **Assumption 3** (easy update of rows and columns):
  Given \( B, \bar{x} \in \text{conv}(X), \bar{x} \notin X_B \), it is “easy” to find \( B' \supseteq B \)
  \[ (\implies \Gamma_{B'}, \gamma_{B'}) \text{ such that } \exists B'' \supseteq B' \text{ such that } \bar{x} \in X_{B''}. \]

- **Structured master problem**
  \[ (\Pi_B) \quad \max \left\{ cx : Ax = b, \ x = C_B\theta_B, \ \Gamma_B\theta_B \leq \gamma_B \right\} \quad (12) \]
  \[ \equiv \text{structured model} \]
  \[ f_B(\lambda) = \max\{ (c - \lambda A)x + xb : x = C_B\theta_B, \ \Gamma_B\theta_B \leq \gamma_B \} \quad (13) \]
The Structured Dantzig-Wolfe Algorithm

\[
\langle \text{initialize } \mathcal{B} \rangle ; \\
\text{repeat} \\
\quad \langle \text{solve } (\Pi_\mathcal{B}) \text{ for } x^*, \lambda^* \text{ (duals of } Ax = b) ; v^* = cx^* \rangle ; \\
\quad \bar{x} = \text{argmin} \{ (c - \lambda^* A)x : x \in X \} ; \\
\quad \langle \text{update } \mathcal{B} \text{ as in Assumption 3} \rangle ; \\
\text{until } v^* < c\bar{x} + \lambda^*(b - A\bar{x})
\]

- Relatively easy\textsuperscript{29} to prove that:
  - finitely terminates with an optimal solution of \((\Pi)\)
  - \ldots\ even if (proper) removal from \(\mathcal{B}\) is allowed (when \(cx^*\) increases)
  - \ldots\ even if \(X\) is non compact and \(\mathcal{B} = \emptyset\) at start (Phase 0)

- The subproblem to be solved is \textbf{identical to that of DW}

- Requires (\(\implies\) \textbf{exploits}) extra information on the structure

- Master problem with \textbf{any structure}, possibly much larger
And it does work somewhat better

<table>
<thead>
<tr>
<th>Problem</th>
<th>$I+$</th>
<th>StabDW</th>
<th>StructDW</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>A</td>
<td>\times C$ imp</td>
<td>cpu gap it</td>
</tr>
<tr>
<td>229 1 185.17</td>
<td>18326 20.53 86</td>
<td>9261 132963</td>
<td>380 7.44 39</td>
</tr>
<tr>
<td>4 125.39</td>
<td>15537 18.81 80</td>
<td>11791 147879</td>
<td>612 9.36 49</td>
</tr>
<tr>
<td>8 85.31</td>
<td>9500 13.08 74</td>
<td>10702 146727</td>
<td>1647 8.87 68</td>
</tr>
<tr>
<td>16 46.09</td>
<td>1900 7.19 52</td>
<td>7268 107197</td>
<td>3167 7.99 108</td>
</tr>
<tr>
<td>287 1 198.87</td>
<td>14559 27.86 66</td>
<td>8815 120614</td>
<td>598 12.54 53</td>
</tr>
<tr>
<td>4 136.97</td>
<td>11934 22.52 62</td>
<td>8426 112308</td>
<td>603 15.07 37</td>
</tr>
<tr>
<td>8 92.94</td>
<td>9656 15.28 64</td>
<td>10098 130536</td>
<td>1221 10.38 41</td>
</tr>
<tr>
<td>16 53.45</td>
<td>3579 11.60 54</td>
<td>6801 98972</td>
<td>3515 9.06 99</td>
</tr>
</tbody>
</table>

- Save sometimes for highly capacitated instances
- Extra advantage: quickly solve reduced binary model to integer optimality ("price and branch") giving better feasible solutions than integer model
- Still likely room for improvement: stabilizing SDW seems promising
Stabilizing the Structured Dantzig-Wolfe Algorithm

- **Exactly the same** as stabilizing DW: stabilized master problem

\[
(\Delta_B, \bar{y}, D) \min \left\{ f_B(\bar{\lambda} + d) + D(d) \right\}
\]

(14)

except \( f_B \) is a different model of \( f \) (not the cutting plane one)

- Even simpler from the primal viewpoint\(^{30}\):

\[
\max \left\{ cx + \bar{\lambda} z - D^*(-z) : z = b - Ax, \ x = C_B \theta_B, \Gamma_B \theta_B \leq \gamma_B \right\}
\]

(15)

- With proper choice of \( D \), still a Linear Program; e.g.

\[
\max \ldots - (\Delta^- + \Gamma^-)z_2^- - \Delta^-z_1^- - \Delta^+z_1^+ - (\Delta^+ + \Gamma^+)z_2^+
\]

\[
z_2^- + z_1^- - z_1^+ - z_2^+ = b - Ax, \ \ldots
\]

\[
z_2^+ \geq 0, \ \varepsilon^+ \geq z_1^+ \geq 0, \ \varepsilon^- \geq z_1^- \geq 0, \ z_2^- \geq 0
\]

- Dual optimal variables of “\( z = b - Ax \)” still give \( d^* \), \ldots

- How to move \( \bar{y} \), handle \( t \), handle \( B \): basically as in\(^9\), actually even somewhat simpler because \( B \) is inherently finite

---

And it actually works a lot better

- **Can do smart warm-start** (MCF + subgradient) to improve performances

<table>
<thead>
<tr>
<th></th>
<th>StructDW</th>
<th>$S^2DW_2$</th>
<th>$S^2DW_\infty$</th>
<th>$S^2DW_\infty–ws^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>cpu</td>
<td>gap</td>
<td>it</td>
<td>cpu</td>
</tr>
<tr>
<td>1</td>
<td>380</td>
<td>7.44</td>
<td>39</td>
<td>1.0e4</td>
</tr>
<tr>
<td>4</td>
<td>612</td>
<td>9.36</td>
<td>49</td>
<td>1.3e4</td>
</tr>
<tr>
<td>8</td>
<td>1647</td>
<td>8.87</td>
<td>68</td>
<td>3.3e4</td>
</tr>
<tr>
<td>16</td>
<td>3167</td>
<td>7.99</td>
<td>108</td>
<td>7.0e4</td>
</tr>
<tr>
<td>1</td>
<td>598</td>
<td>12.54</td>
<td>53</td>
<td>2.1e4</td>
</tr>
<tr>
<td>4</td>
<td>603</td>
<td>15.07</td>
<td>37</td>
<td>1.8e4</td>
</tr>
<tr>
<td>8</td>
<td>1221</td>
<td>10.38</td>
<td>41</td>
<td>5.2e4</td>
</tr>
<tr>
<td>16</td>
<td>3515</td>
<td>9.06</td>
<td>99</td>
<td>1.3e5</td>
</tr>
</tbody>
</table>

- **Quadratic stabilization converges faster but master problem too costly**

- **Warm-started stabilised** (with $\| \cdot \|_\infty$) structured decomposition gives extremely good upper and lower bounds in (relatively) short time
Not that we entirely gave up on subgradients, either

- In fact we tested them all very thoroughly (for knapsack decomposition)\(^{31}\)
- We even tested fancy smoothed subgradient (≡ quadratic knapsack\(^{32}\)) but results were not good: ≈linear in a doubly-logarithmic chart

Subgradients faster but flatline at \(\varepsilon \approx 1e^{-4}\), smoothed does \(\varepsilon = 1e^{-6}\) but it requires \(1e+6\) iterations to get there

Exploiting information about \(f_*\) helps (black solid line) but not enough\(^{33}\)

---


\(^{32}\)F., Gorgone “A Library for Continuous Convex Separable Quadratic Knapsack Problems” EJOR, 2013

But Bernard loved models more than algorithms

... and was always capable of finding new gems in a highly mined cave
But Bernard loved models more than algorithms

... and was always capable of finding new gems in a highly mined cave.

He took the venerable knapsack relaxation and came up with three new node-based ones by playing nifty reformulation tricks.
But Bernard loved models more than algorithms

... and was always capable of finding new gems in a highly mined cave

He took the venerable knapsack relaxation and came up with three new node-based ones by playing nifty reformulation tricks

\[ K_{i}^{O/T/D} = \{ k \in K : i \text{ is origin/transhipment/destination for } i \} \]

Add redundant \( \sum_{j \in N_{i}^{+}} x_{ij}^{k} \leq g_{i}^{k} = \min \{ d^{k}, \sum_{j \in N_{i}^{-}} u_{ji} \} \) \( i \in N, k \in K_{i}^{T} \)

**Facility location relaxation**, decomposes by \( i \in N \equiv \text{node}: \)

\[
\begin{align*}
\min & \sum_{j \in N_{i}^{+}} \sum_{k \in K} c_{ij}^{k}(\pi)x_{ij}^{k} + f_{ij}y_{ij} \\
\sum_{j \in N_{i}^{+}} x_{ij}^{k} &= d^{k} \quad k \in K_{i}^{O} \\
\sum_{j \in N_{i}^{+}} x_{ij}^{k} &\leq g_{i}^{k} \quad k \in K_{i}^{T} \\
x_{ij}^{k} &= 0 \quad j \in N_{i}^{+}, k \in K_{i}^{D} \cup K_{j}^{O} \\
\sum_{k \in K} x_{ij}^{k} &\leq u_{ij}y_{ij} \quad j \in N_{i}^{+} \\
0 &\leq x_{ij}^{k} \leq d^{k}y_{ij} \quad j \in N_{i}^{+}, k \in K \\
y_{ij} &\in \{0, 1\} \quad j \in N_{i}^{+}
\end{align*}
\]
Introduce copies of design \((z)\) and flow \((v)\) variables, then link them with copy constraints (Lagrangian decomposition)

\[
\begin{align*}
    z_{ij} - y_{ij} &= 0 & (i, j) \in A & \quad (16) \\
    v^k_{ij} - x^k_{ij} &= 0 & (i, j) \in A, \; k \in K & \quad (17)
\end{align*}
\]

Add a bunch of redundant constraints

\[
\begin{align*}
    \sum_{j \in N_i^-} v^k_{ji} &= d^k & i \in N, \; k \in K_i^D \\
    v^k_{ji} &= 0 & (j, i) \in A, \; k \in K_i^O \cup K_j^D \\
    \sum_{k \in K} v^k_{ji} &\leq u_{ji}z_{ji} & (j, i) \in A \\
    0 &\leq v^k_{ji} \leq d^k z_{ji} & (j, i) \in A, \; k \in K \\
    z_{ji} &\in \{0, 1\} & (j, i) \in A \\
    \sum_{j \in N_i^-} v^k_{ji} &\leq h^k_i = \min\{d^k, \sum_{j \in N_i^+} u_{ij}\} & i \in N, \; k \in K_i^T 
\end{align*}
\]

Now relax (16) and (17) together with (2)
Behold the forward-backward facility location relaxation

- One problem (for each $i \in N$) just like before, except with
  \[
  \min \sum_{j \in N_i^+} \sum_{k \in K} c_{ij}^k(\omega, \pi)x_{ij}^k + f_{ij}(\gamma)y_{ij}
  \]

- The other (for each $i \in N$) analogous on the $(v, z)$
  \[
  \min \sum_{j \in N_i^-} \sum_{k \in K} c_{ji}^k(\omega)v_{ji}^k + f_{ji}(\gamma)z_{ji}
  \]

- Still decomposes by $i \in N \equiv$ node, but now two CFL problems

- Correspondingly, better bound than the facility location relaxation
And then yet another one

- Add to the forward-backward facility location relaxation the constraints
  \[ \sum_{j \in N_i^+} x_{ij}^k - \sum_{j \in N_i^-} v_{ji}^k = 0 \quad i \in N, \; k \in K_i^T \]

- Two subproblems \(\mapsto\) multicommodity single-node fixed-charge problem more difficult \(\Rightarrow\) better bound than forward-backward relaxation

- A whole new set of bound quality/time trade-offs to explore

<table>
<thead>
<tr>
<th></th>
<th>(Z^{LP})</th>
<th>(Z^{FW})</th>
<th>(Z^{KN})</th>
<th>(Z^{FL})</th>
<th>(Z^{FB})</th>
<th>(Z^{SN})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average gap</td>
<td>—</td>
<td>0.003</td>
<td>0.008</td>
<td>-0.508</td>
<td>-0.919</td>
<td>-1.781</td>
</tr>
<tr>
<td>Minimum gap</td>
<td>—</td>
<td>0.000</td>
<td>0.000</td>
<td>-4.767</td>
<td>-7.713</td>
<td>-20.518</td>
</tr>
<tr>
<td>Total time (sec.)</td>
<td>170.25</td>
<td>7.40</td>
<td>125.56</td>
<td>699.74</td>
<td>4073.34</td>
<td>4677.71</td>
</tr>
<tr>
<td>Number of iterations</td>
<td>—</td>
<td>20</td>
<td>5866</td>
<td>284</td>
<td>373</td>
<td>316</td>
</tr>
<tr>
<td>Lagrangian time (%)</td>
<td>—</td>
<td>5</td>
<td>18</td>
<td>28</td>
<td>10</td>
<td>65</td>
</tr>
<tr>
<td>Master problem time (%)</td>
<td>—</td>
<td>95</td>
<td>82</td>
<td>72</td>
<td>90</td>
<td>35</td>
</tr>
</tbody>
</table>

- A bunch of new Lagrangian-based math-heuristics, competitive results\(^{34}\)

- A renewed interest in incremental/inexact Bundle methods\(^{35}\)

- Lots of fun!

\(^{34}\) Kazemzadeh, Bektas, Crainic, F., Gendron, Gorgone “Node-Based Lagrangian Relaxations […]” \(DAM\), 2022

\(^{35}\) van Ackooij, F “Incremental Bundle Methods Using Upper Models” \(SIOPT\), 2018
And he was not done with knapsack relaxation either

- **Knapsack relaxation** decomposes by arc if $Y = \{ 0, 1 \}^{|A|}$
  
  $\begin{align*}
  \min & \quad \sum_{(i,j) \in A} \left( \sum_{k \in K} (c_{ij}^k - \pi_i^k + \pi_j^k)x_{ij}^k + f_{ij}y_{ij} \right) \\
  \text{s.t.} & \quad \sum_{k \in K} d_k^k x_{ij}^k \leq u_{ij} y_{ij} \\
  & \quad 0 \leq x_{ij}^k \leq u_{ij}^k y_{ij} \\
  & \quad y \in Y \\
  & \quad (i,j) \in A
  \end{align*}$

- Still solvable if $Y \subset \{ 0, 1 \}^{|A|}$ “not too nasty”: first
  
  $f_{ij}^*(\pi) = \min \sum_{k \in K} (c_{ij}^k - \pi_i^k + \pi_j^k)x_{ij}^k$
  
  $\begin{align*}
  \sum_{k \in K} d_k^k x_{ij}^k \leq u_{ij} \\
  0 \leq x_{ij}^k \leq u_{ij}^k \\
  k \in K
  \end{align*}$

  and then $\min \left\{ \sum_{(i,j) \in A} (f_{ij}^*(\pi) + f_{ij})y_{ij} : y \in Y \right\}$

- Computational cost $\approx$ same but **Lagrangian function no longer separable**

  $\implies$ wave goodbye to disaggregate master problem, easy components

- Still, the Lagrangian problem is **somewhat separable**

- We want to “show this quasi-separability to the master problem”
General setting: quasi-separable problems

Set of $N$ quasi-continuous (vector) variables $x_i$ governed by $y_i$

\[ \max dy + \sum_{i \in N} c_i x_i \tag{18} \]

\[ Dy + \sum_{i \in N} C_i x_i = b \tag{19} \]

\[ A_i x_i \leq b_i y_i \quad i \in N \tag{20} \]

\[ x_i \in X_i \quad i \in N \tag{21} \]

\[ y \in Y \tag{22} \]

$m$ linking constraints (19): Lagrangian relaxation

\[ \phi(\lambda) = \lambda b + \max \left\{ (d - \lambda D) y + \sum_{i \in N} (c_i - \lambda C_i) x_i : (20), (21), (22) \right\} \]

Two-stage solution procedure

\[ \phi_i(\lambda) = \max \left\{ (c_i - \lambda C_i)x_i : x_i \in X_i \right\} \quad i \in N \tag{23} \]

\[ \phi(\lambda) = \lambda b + \max \left\{ \sum_{i \in N} (d_i - \lambda D^i + \phi_i(\lambda)) y_i : y \in Y \right\} \tag{24} \]
Making it separable: the dumb way

- D-W reformulation is not disaggregate

\[
\begin{align*}
\max \ & \sum_{(\bar{y}, \bar{x}) \in YX} \left( d\bar{y} + \sum_{i \in N} c_i \bar{x}_i \right) \theta(\bar{y}, \bar{x}) \\
\sum_{(\bar{y}, \bar{x}) \in YX} \left( D\bar{y} + \sum_{i \in N} C_i \bar{x}_i \right) \theta(\bar{y}, \bar{x}) &= b \\
\sum_{(\bar{y}, \bar{x}) \in YX} \theta(\bar{y}, \bar{x}) &= 1, \quad \theta(\bar{y}, \bar{x}) \geq 0 \quad (\bar{y}, \bar{x}) \in YX
\end{align*}
\] (25)

- Can be made so the hard way: also relax (20) \( (\mu = [\mu_i]_{i \in N} \geq 0) \)

\[
\phi(\lambda, \mu) = \lambda b + \psi(\lambda, \mu) + \sum_{i \in N} \psi_i(\lambda, \mu_i) \quad \text{with} \quad (28)
\]

\[
\psi_i(\lambda, \mu_i) = \max \ \{ (c_i - \lambda C_i - \mu_i A_i)x_i : x_i \in X_i \} \quad (29)
\]

\[
\psi(\lambda, \mu) = \max \ \{ \sum_{i \in N} (d_i - \lambda D_i - \mu_i b_i)y_i : y \in Y \} \quad (30)
\]

- Many more multipliers \( (|K| |A| \text{ in FC-MMCF}) \)

- Can easily destroy any advantage due to separability
Making it separable: the better way

“Easy component” $Y$ version:

$$\max\ dy + \sum_{i \in N} \sum_{x_i \in X_i} (C_i \bar{x}_i) \theta \bar{x}_i$$

$$Dy + \sum_{i \in N} \sum_{x_i \in X_i} (C_i \bar{x}_i) \theta \bar{x}_i = b$$

$$\sum_{x_i \in X_i} (A_i \bar{x}_i) \theta \bar{x}_i \leq y$$

$$\sum_{x_i \in X_i} \theta \bar{x}_i = 1$$

$$y \in Y, \ \theta \bar{x}_i \geq 0$$

Nifty idea: replace (33)–(34) with

$$\sum_{x_i \in X_i} \theta \bar{x}_i = y$$

then relax (35) with multipliers $\gamma = [\gamma_i]_{i \in N} \geq 0$

Multipliers are from master problem constraints (which they are . . . )

Non-easy component version obvious

Much fewer multipliers (1 instead of $m$), much more elegant
And it also works in practice

- Results from last week (Enrico is the pit bull of numerical experiments)
- Time limit 18000 seconds (always hit if not shown)

<table>
<thead>
<tr>
<th>name</th>
<th>BKA-10 gap</th>
<th>BKA-4000</th>
<th>BKD gap</th>
<th>BQS gap</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time</td>
<td>gap</td>
<td>time</td>
<td>gap</td>
</tr>
<tr>
<td>p33</td>
<td>5.71e-06</td>
<td>227.68</td>
<td>6.58e-07</td>
<td>4.63e-02</td>
</tr>
<tr>
<td>p34</td>
<td>8.20e-06</td>
<td>233.14</td>
<td>3.47e-07</td>
<td>5.43e-02</td>
</tr>
<tr>
<td>p35</td>
<td>7.33e-06</td>
<td>260.01</td>
<td>8.63e-07</td>
<td>8.92e-02</td>
</tr>
<tr>
<td>p36</td>
<td>9.61e-06</td>
<td>57.02</td>
<td>8.48e-07</td>
<td>9.33e-02</td>
</tr>
<tr>
<td>p37</td>
<td>5.14e-04</td>
<td>—</td>
<td>3.22e-04</td>
<td>9.23e-02</td>
</tr>
<tr>
<td>p38</td>
<td>4.79e-04</td>
<td>—</td>
<td>3.24e-04</td>
<td>5.75e-02</td>
</tr>
<tr>
<td>p39</td>
<td>4.54e-06</td>
<td>—</td>
<td>2.46e-05</td>
<td>4.46e-02</td>
</tr>
<tr>
<td>p40</td>
<td>4.99e-06</td>
<td>—</td>
<td>1.45e-05</td>
<td>5.13e-02</td>
</tr>
<tr>
<td>p41</td>
<td>3.22e-06</td>
<td>212.67</td>
<td>3.13e-08</td>
<td>4.92e-02</td>
</tr>
<tr>
<td>p42</td>
<td>3.29e-06</td>
<td>130.07</td>
<td>2.58e-08</td>
<td>7.34e-02</td>
</tr>
<tr>
<td>p43</td>
<td>9.91e-06</td>
<td>193.61</td>
<td>2.97e-08</td>
<td>8.99e-02</td>
</tr>
<tr>
<td>p44</td>
<td>5.16e-06</td>
<td>134.04</td>
<td>1.28e-06</td>
<td>1.34e-01</td>
</tr>
</tbody>
</table>

Our last paper all together

---

\(^{36}\) F., Gendron, Gorgone “Separable Lagrangian Decomposition for Quasi-Separable Problems” *Bernard’s Book*, 2023

A. Frangioni (DI — UniPi) Bernard and Multicommodity Flows Bernard 2023
But Bernard’s legacy will live on, also in software

- Putting these ideas in practice: easier said than done
- Specialized implementations for one application “relatively easy”
- General implementations for all problems with same structure harder: it took $\approx 10$ years from idea to paper for easy components on top of existing, nicely structured C++ bundle code
- It’s 10 years since $S^2$DW and we still don’t have a general implementation
- Issue: extracting structure from problems
- Issue: really using this in a B&C approach
  $\approx 20$ years doing this well for Multicommodity Network Design
- Especially hard: multiple nested forms of structure, reformulation
- Current modelling/solving tools just don’t do it
- So I have been building my own
Meet SMS++

https://gitlab.com/smspp/smspp-project

“For algorithm developers, from algorithm developers”

- Open source (LGPL3)
- 1 “core” repo, 1 “umbrella” repo, 10+ problem and/or algorithmic-specific repos (public, more in development)
- Extensive Doxygen documentation https://smspp.gitlab.io
- But no real user manual as yet
What SMS++ is

- A core set of C++-17 classes implementing a **modelling system** that:
  - explicitly supports the notion of Block \(\equiv\) nested structure
  - separately provides “semantic” information from “syntactic” details (list of constraints/variables \(\equiv\) one specific formulation among many)
  - allows exploiting **specialised Solver** on Block with specific structure
  - manages **any dynamic change in the Block** beyond “just” generation of constraints/variables
  - supports reformulation/restriction/relaxation of Block
  - has built-in **parallel processing capabilities**
  - **should** be able to deal with almost anything (bilevel, PDE, . . .)

- An **hopefully growing set of specialized Block and Solver**

- **In perspective** an ecosystem fostering collaboration and code sharing: a community-building effort as much as a (suite of) software product(s)

- I believe Bernard would have loved it
And finally the really important things
And finally the really important things
And finally the really important things

VIETATO ATTRAVERSARE I BINARI
SERVIRSI DEL SOTTOPASSAGGIO
ES IST VERBOTEN ÜBER DAS GLEIS ZU GEHEN
BENUTZEN SIE BITTE DIE BAHNÜNTER ÜHRUNG
DÉ ENSE DE TRAVERSER LES BINAIRES
UTILISEZ LE PASSAGE SOUTERRAIN
DO NOT CROSS THE TRACKS
PLEASE USE THE SUBWAY
And finally the really important things
And finally the really important things