Decomposition Methods for Network Design

Bernard Gendron*

May 12, 2010

Spring School on Supply Chain and Transportation Network Design

* CIRRELT and Département d'informatique et de recherche opérationnelle, Université de Montréal, Canada

Outline

Network design

- Network with multiple commodities
- Each commodity flows between supply and demand points
- Minimization of a "complex" (non-convex) objective function
  - Tradeoff between transportation and investment costs
  - Transportation costs: not necessarily linear, can be piecewise linear
  - Investment costs: "fixed" cost for building, renting, operating "facilities" at nodes or arcs of the network
- Additional constraints: budget, capacity, topology, reliability,...
- Variants:
  - Centralized / Decentralized
  - Static / Dynamic
  - Determinist / Stochastic
  - Strategic / Tactical / Operational

Infrastructure network design: strategic planning

- Planning horizon: years
- Decisions: invest in building roads, warehouses, plants,...
- Typical assumptions:
  - Central control
  - Static network
  - Linear transportation costs
  - Fixed costs for investment decisions
  - Usually no capacities
  - Known demands based on average values
- Robustness is an issue: stochastic demands?

Service network design: tactical planning

- Planning horizon: months
- Decisions: establish or not “services” (vehicles moving between two points) + flows-inventories
- Dynamic network: space-time expansion
  - Node = location-period
  - Transportation arc = (location1-period1, location2-period2) = moving from location1 to location2 in time (period2-period1)
  - Inventory arc = (location-period, location-period+1) = holding inventory at location between two consecutive periods
- Typical assumptions:
  - Central control
  - Linear inventory-transportation costs
  - Fixed costs for service decisions
  - Service capacities
  - Known demands

Adaptive network design: operational planning

- Planning horizon: days
- Decisions: operate or not "facilities" (warehousing or parking space) for fast product delivery + how many vehicles to use on each arc
- Typical assumptions:
  - Central control
  - Dynamic network
  - Piecewise linear transportation costs
  - Fixed costs for facility decisions
  - Facility and vehicle capacities
  - Known demands

Multicommodity capacitated network design

- Directed network $G = (N, A)$, with node set N and arc set A
- Commodity set K: known demand d^k between origin $O(k)$ and destination $D(k)$ for each $k \in K$
- Unit transportation cost c_{ij} on each arc (i, j)
- Capacity u_{ij} on each arc (i, j)
- Cost f_{ij} for each capacity unit installed on arc (i, j)

Problem formulation

$$Z = \min \sum_{(i,j)\in A} \sum_{k\in K} c_{ij} d^k x_{ij}^k + \sum_{(i,j)\in A} f_{ij} y_{ij}$$

$$\sum_{j\in N_i^+} x_{ij}^k - \sum_{j\in N_i^-} x_{ji}^k = \begin{cases} & 1, \quad i = O(k) \\ - & 1, \quad i = D(k) \\ & 0, \quad i \neq O(k), D(k) \end{cases} \qquad i \in N, \; k \in K$$

$$\sum_{k\in K} d^k x_{ij}^k \leq u_{ij} y_{ij} \quad (i,j) \in A$$

$$0 \leq x_{ij}^k \leq 1 \quad (i,j) \in A, \; k \in K$$

$$y_{ij} \; integer \quad (i,j) \in A$$

Extensions

- Fixed-charge: $0 \leq y_{ij} \leq 1 \quad (i,j) \in A$
- Asset-balance constraints: $\sum_{j \in N_i^+} y_{ij} - \sum_{j \in N_i^-} y_{ji} = 0 \quad i \in N$
- Non-bifurcated flows: x_{ij}^k *integer* $\quad (i,j) \in A,\ k \in K$
- Piecewise linear arc flow costs
- Multifacility design: several facilities $t \in T_{ij}$ on each arc, each with capacity u_{ij}^t and cost f_{ij}^t

Multicommodity capacitated fixed-charge network design

- Directed network $G = (N, A)$, with node set N and arc set A
- Commodity set K: known demand d^k between origin $O(k)$ and destination $D(k)$ for each $k \in K$
- Unit transportation cost c_{ij} on each arc (i, j)
- Capacity u_{ij} on each arc (i, j)
- Fixed charge f_{ij} incurred whenever arc (i, j) is used to transport some commodity units

Problem formulation (MCND)

$$Z = \min \sum_{(i,j)\in A} \sum_{k\in K} c_{ij} x_{ij}^k + \sum_{(i,j)\in A} f_{ij} y_{ij}$$

$$\sum_{j\in N_i^+} x_{ij}^k - \sum_{j\in N_i^-} x_{ji}^k = \begin{cases} & d^k, \quad i = O(k) \\ - & d^k, \quad i = D(k) \\ & 0, \quad i \neq O(k), D(k) \end{cases} \qquad i \in N, \ k \in K$$

$$\sum_{k\in K} x_{ij}^k \leq u_{ij} y_{ij} \quad (i,j) \in A$$

$$x_{ij}^k \geq 0 \quad (i,j) \in A, \ k \in K$$

$$y_{ij} \in \{0,1\} \quad (i,j) \in A$$

Developing solution methods for MCND: why?

- Generic problem: methods can be adapted to many similar network design applications
- But why "develop solution methods": simply use a black-box solver!
- Things are not so simple:
  - LP relaxations are weak (typically, more than 20% gap w.r.t. optimal value)
  - LP relaxations can be hard to solve when the number of commodities is large: degeneracy
  - Combinatorial explosion
  - Dominant factors in increasing the complexity of a problem: high fixed charges + tight capacities + large number of commodities
- Two main classes of methods:
  - Mathematical programming
  - Metaheuristics

Overview of solution methods for MCND

- Mathematical programming
  - Lagrangian relaxation: Gendron, Crainic 1994; Gendron, Crainic, Frangioni 1998; Holmberg, Yuan 2000; Crainic, Frangioni, Gendron 2001; Sellmann, Kliewer, Koberstein 2002; Kliewer, Timajev 2005; Bektas, Crainic, Gendron 2009
  - Cutting-plane methods: Chouman, Crainic, Gendron 2009
  - Benders decomposition: Costa, Cordeau, Gendron 2009
- Metaheuristics
  - Tabu search: Crainic, Farvolden, Gendreau 2000; Crainic, Gendreau 2002; Crainic, Gendreau, Ghamlouche 2003, 2004
- Hybrid algorithms
  - Slope scaling with long-term memory: Crainic, Gendron, Hernu 2004

Strong formulation

$$Z = \min \sum_{(i,j)\in A} \sum_{k\in K} c_{ij} x_{ij}^k + \sum_{(i,j)\in A} f_{ij} y_{ij}$$

$$\sum_{j\in N_i^+} x_{ij}^k - \sum_{j\in N_i^-} x_{ji}^k = \begin{cases} d^k, & i = O(k) \\ -d^k, & i = D(k) \\ 0, & i \neq O(k), D(k) \end{cases} \qquad i \in N,\ k \in K \quad (\pi_i^k)$$

$$\sum_{k\in K} x_{ij}^k \leq u_{ij} y_{ij} \quad (i,j) \in A \quad (\alpha_{ij})$$

$$x_{ij}^k \leq b_{ij}^k y_{ij} \quad (i,j) \in A,\ k \in K \quad (\beta_{ij}^k)$$

$$x_{ij}^k \geq 0 \quad (i,j) \in A,\ k \in K$$

$$y_{ij} \in \{0,1\} \quad (i,j) \in A$$

Shortest path relaxation

$$Z(\alpha, \beta) = \min \sum_{(i,j)\in A} \sum_{k\in K} (c_{ij} + \alpha_{ij} + \beta_{ij}^k) x_{ij}^k$$

$$+ \sum_{(i,j)\in A} (f_{ij} - u_{ij}\alpha_{ij} - \sum_{k\in K} b_{ij}^k \beta_{ij}^k) y_{ij}$$

$$\sum_{j\in N_i^+} x_{ij}^k - \sum_{j\in N_i^-} x_{ji}^k = \begin{cases} d^k, & i = O(k) \\ -d^k, & i = D(k) \\ 0, & i \neq O(k), D(k) \end{cases} \qquad i \in N,\ k \in K$$

$$y_{ij} \in \{0, 1\} \quad (i,j) \in A$$

Knapsack relaxation

$$Z(\pi) = \min \sum_{(i,j)\in A} \sum_{k\in K} (c_{ij} + \pi_i^k - \pi_j^k) x_{ij}^k + \sum_{(i,j)\in A} f_{ij} y_{ij} + \sum_{k\in K} d^k (\pi_{D(k)}^k - \pi_{O(k)}^k)$$

$$\sum_{k\in K} x_{ij}^k \leq u_{ij} y_{ij} \quad (i,j) \in A$$

$$x_{ij}^k \leq b_{ij}^k y_{ij} \quad (i,j) \in A, \; k \in K$$

$$x_{ij}^k \geq 0 \quad (i,j) \in A, \; k \in K$$

$$y_{ij} \in \{0,1\} \quad (i,j) \in A$$

Theoretical and computational results

- Both Lagrangian relaxations provide the same lower bound as the strong LP relaxation
- Lower bound within 9% of optimality on average
- To find (near-)optimal Lagrangian multipliers, two classes of methods have been traditionally used:
  - Subgradient methods
  - Bundle methods
- Our computational results show that:
  - Bundle methods are much more robust
  - Bundle methods converge faster
  - Any of these two methods converge much faster than solving the strong LP relaxation with the simplex method

Cutting-plane method: motivations

- Starting with the weak LP relaxation, iteratively add violated valid inequalities:
  - To be more efficient: keep the problem size as small as possible
  - To be more effective: improve the lower bound
- But the black-box solver already does that, so why not simply use it?
- True, but we can be more efficient and more effective by exploiting the structure of MCND
- Five classes of valid inequalities:
  - Strong inequalities (SI)
  - Cover inequalities (CI)
  - Minimum cardinality inequalities (MCI)
  - Flow cover inequalities (FCI)
  - Flow pack inequalities (FPI)

Computational results

- Comparison with CPLEX

	CI			FCI			All		
	gap	cpu	cuts	gap	cpu	cuts	gap	cpu	cuts
CPLEX	5.40%	0.6%	10	23.17%	70.9%	305	23.20%	72.5%	306
Cutting-Plane	8.54%	1.4%	27	26.25%	28.6%	766	27.98%	12.6%	1537

Computational results

- Comparison with CPLEX

	CI			FCI			All		
	gap	cpu	cuts	gap	cpu	cuts	gap	cpu	cuts
CPLEX	5.40%	0.6%	10	23.17%	70.9%	305	23.20%	72.5%	306
Cutting-Plane	8.54%	1.4%	27	26.25%	28.6%	766	27.98%	12.6%	1537

- Comparison between valid inequalities

	None+		All-	
	gap	cpu	gap	cpu
$\emptyset$	0%	0%	27.98%	12.6%
SI	26.53%	7.3%	26.97%	30.0%
CI	8.54%	1.4%	27.92%	12.8%
MCI	8.00%	1.4%	27.97%	12.6%
FCI	26.25%	28.6%	27.97%	10.9%
FPI	26.75%	32.9%	27.94%	10.5%

Branch-and-cut algorithm

- Apply the cutting-plane method at every node of the B&B tree
- Generate Benders feasibility cuts along the tree
- Add pre- and post-processing at every node to reduce the size of the solution space
- Apply a variant of strong branching
- The resulting B&C algorithm is:
  - Much better than CPLEX B&C using the weak LP relaxation
  - Much better than CPLEX B&C using the strong LP relaxation
  - Competitive with CPLEX B&C using the cutting-plane LP relaxation
- How does it compare with state-of-the-art Lagrangian-based B&B?

General integer formulation (I)

$$\min \sum_{k\in K} \sum_{(i,j)\in A} d^k c_{ij} x_{ij}^k + \sum_{(i,j)\in A} f_{ij} y_{ij}$$

$$\sum_{j\in N} x_{ij}^k - \sum_{j\in N} x_{ji}^k = \begin{cases} 1, & \text{if } i = O(k) \\ -1, & \text{if } i = D(k) \\ 0, & \text{if } i \neq O(k), D(k) \end{cases} \qquad \forall\, i \in N,\ k \in K$$

$$\sum_{k\in K} d^k x_{ij}^k \leq u_{ij} y_{ij} \quad \forall\, (i,j) \in A$$

$$0 \leq x_{ij}^k \leq 1 \quad \forall\, (i,j) \in A,\ k \in K$$

$$y_{ij} \geq 0 \quad \forall\, (i,j) \in A$$

$$y_{ij} \ \text{integer} \quad \forall\, (i,j) \in A$$

Lagrangian relaxation of flow conservation

$$\min \sum_{k \in K} \sum_{(i,j) \in A} (d^k c_{ij} - \pi_i^k + \pi_j^k) x_{ij}^k + \sum_{(i,j) \in A} f_{ij} y_{ij} + \sum_{k \in K} \pi_{O(k)}^k - \pi_{D(k)}^k$$

$$\sum_{k \in K} d^k x_{ij}^k \leq u_{ij} y_{ij} \quad \forall\, (i,j) \in A$$

$$0 \leq x_{ij}^k \leq 1 \quad \forall\, (i,j) \in A,\ k \in K$$

$$y_{ij} \geq 0 \quad \forall\, (i,j) \in A$$

$$y_{ij}\ \mathit{integer} \quad \forall\, (i,j) \in A$$

- Lagrangian subproblem *decomposes by arc*
- Easy ($\approx$ 2 continuous knapsack) but *no* integrality property

Residual capacity inequalities

- For any $P \subseteq K$, define $d^P = \sum_{k \in P} d^k$
- Then, for any $(i,j) \in A$, define

$$a_{ij}^P = \frac{d^P}{u_{ij}}$$

$$q_{ij}^P = \lceil a_{ij}^P \rceil$$

$$r_{ij}^P = a_{ij}^P - \lfloor a_{ij}^P \rfloor$$

- Residual capacity inequalities

$$\sum_{k \in P} \{a_{ij}^k (1 - x_{ij}^k)\} \geq r_{ij}^P (q_{ij}^P - y_{ij}) \quad \forall (i,j) \in A,\ P \subseteq K$$

- Characterize the convex hull of solutions to the Lagrangian subproblem (Magnanti, Mirchandani, Vachani 1993)
- Separation can be performed in $O(|A||K|)$ (Atamtürk, Rajan 2002)

Multiple choice model

$$y_{ij} \leq \left\lceil \frac{\sum_{k \in K} d^k}{u_{ij}} \right\rceil = T_{ij}$$

$$S_{ij} = \{1, \ldots, T_{ij}\}$$

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

$$x_{ij}^s = \begin{cases} \sum_{k \in K} d^k x_{ij}^k, & \text{if } y_{ij} = s \\ 0, & \text{otherwise} \end{cases} \quad \forall\ s \in S_{ij}$$

Binary formulation (B)

$$y_{ij} = \sum_{s \in S_{ij}} s y_{ij}^s \quad \forall (i,j) \in A$$

$$\sum_{k \in K} d^k x_{ij}^k = \sum_{s \in S_{ij}} x_{ij}^s \quad \forall (i,j) \in A$$

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

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

$$y_{ij}^s \geq 0 \quad (i,j) \in A, s \in S_{ij}$$

$$y_{ij}^s \ integer \quad (i,j) \in A, s \in S_{ij}$$

Variable disaggregation and extended formulation (B^+)

- Extended auxiliary variables

$$x_{ij}^{ks} = \begin{cases} x_{ij}^{k}, & \textit{if } y_{ij} = s \\ 0, & \textit{otherwise} \end{cases} \qquad \forall\ s \in S_{ij}$$

$$x_{ij}^{k} = \sum_{s \in S_{ij}} x_{ij}^{ks} \quad \forall (i,j) \in A, k \in K$$

$$x_{ij}^{s} = \sum_{k \in K} d^k x_{ij}^{ks} \quad \forall (i,j) \in A, s \in S_{ij}$$

- Extended linking inequalities

$$x_{ij}^{ks} \leq y_{ij}^{s} \quad \forall (i,j) \in A, k \in K, s \in S_{ij}$$

Polyhedral results: notation

- $F(M)$: feasible set for model M
- $conv(F(M))$: convex hull of $F(M)$
- $LP(M)$: LP relaxation for model M
- $LS(M)$: Lagrangian subproblem
  (relaxation of flow conservation constraints)
- $LD(M)$: Lagrangian dual for $LS(M)$

Polyhedral results

- $LD(I)$ and $LD(B^+)$ are equivalent
- $F(LP(LS(B^+))) = conv(F(LS(B^+)))$
  (Croxton, Gendron, Magnanti 2007)
- $LP(B^+)$ and $LD(B^+)$ are equivalent
- $LP(B^+)$ *and* $LD(I)$ *are equivalent*
- $I^+ = I +$ residual capacity inequalities
- $LP(B^+)$ *and* $LP(I^+)$ *are equivalent* (Frangioni, Gendron 2009)

Reformulations and decomposition

- "Structured" MIP:

$$(P) \qquad \min_x \{\ cx :\ Ax = b\ ,\ x \in X\ \}$$

where $\qquad (P_\alpha) \qquad Z(\alpha) = \min_x \{\ cx + \alpha(b - Ax)\ :\ x \in X\ \}$

"significantly easier" than (P)

- Lagrangian dual:

$$(LD)\ \max_\alpha \{\ Z(\alpha)\ \} = \min_x \{\ cx :\ Ax = b\ ,\ x \in conv(X)\ \} (LP)$$

- Reformulation:

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

- Examples:
  - Dantzig-Wolfe Reformulation (DW)
  - Extended Formulation (B^+)

Structured DW decomposition: assumptions

- Assumption 1 (*reformulation*):

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

- Assumption 2 (*padding with zeroes*):

$$\Gamma_{\mathcal{B}}\bar{\theta}_{\mathcal{B}} \leq \gamma_{\mathcal{B}} \quad \Rightarrow \Gamma\left[\bar{\theta}_{\mathcal{B}}, 0\right] \leq \gamma$$

$$\Rightarrow X_{\mathcal{B}} = \{\ x = C_{\mathcal{B}}\theta_{\mathcal{B}} \ :\ \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \gamma_{\mathcal{B}}\ \} \subseteq conv(X)$$

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

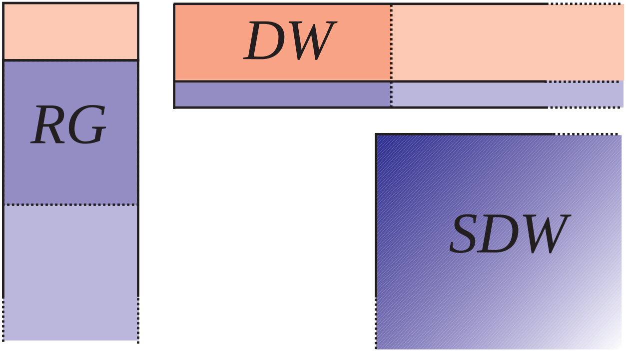
Structured DW decomposition: algorithm

$\langle$ initialize $\mathcal{B}$ $\rangle$;
repeat
    $\langle$ solve $(LP_{\mathcal{B}})$ for $\tilde{x}$, $\tilde{\alpha}$; $\tilde{v} = c\tilde{x}$ $\rangle$;
    $\bar{x} = \mathrm{argmin}_x\{\ (c - \tilde{\alpha}A)x : x \in X\ \}$;   /* $(P_{\tilde{\alpha}})$ */
    if($\tilde{v} = c\bar{x} + \tilde{\alpha}(b - A\bar{x})$)
        then STOP;   /* $\tilde{x}$ optimal */
        else $\langle$ update $\mathcal{B}$ as in Assumption 3 $\rangle$;
until $\sim$ **STOP**

- Finitely terminates with an optimal solution of (LP)
- . . . even if (proper) removal of indices from $\mathcal{B}$ is allowed

Structured DW and other decomposition methods

- Generalizes DW, whose unstructured model is identical for all applications (except when exploiting disaggregation)
- Substantially different from both RG (Row Generation) and DW

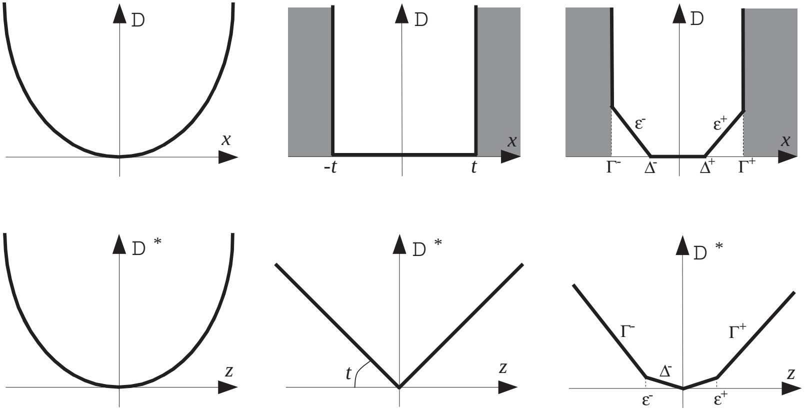


Stability issues in (structured)DW

- The next $\tilde{\alpha}$ can be very far from the current one
- In general, the sequence of $\tilde{\alpha}$ is unstable, has no locality properties and convergence speed does not improve near the optimum
- Counter-measure: use a Proximal Point method defined by a *stabilizing term* $\mathcal{D}_t$, depending on the current $\tilde{\alpha}$ and proximal parameter(s) t

Some stabilizing terms

a penalty — a trust region — or both



$$\mathcal{D}_t = \frac{1}{2t}\|\cdot\|_2^2 \qquad \mathcal{D}_t = I_{B_\infty(t)} \qquad \mathcal{D}_{\Gamma^\pm,\Delta^\pm,\varepsilon^\pm} = \dots$$

$$\mathcal{D}_t^* = \frac{1}{2}t\|\cdot\|_2^2 \qquad \mathcal{D}_t^* = t\|\cdot\|_1 \qquad \mathcal{D}^*_{\Gamma^\pm,\Delta^\pm,\varepsilon^\pm} = \dots$$

Stabilized structured DW algorithm

- Exactly the same as stabilizing DW!
- Stabilized DW = Proximal Point + Column Generation (= Bundle, Frangioni 2002)
- Even simpler from the primal viewpoint:

$$\min \left\{ cx - \tilde{\alpha}z + \mathcal{D}_t^*(-z) \; : \; z = Ax - b \, , \; x = C_{\mathcal{B}}\theta_{\mathcal{B}} \, , \; \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \gamma_{\mathcal{B}} \right\}$$

- With proper choice of $\mathcal{D}_t^*$, this is still a linear program
- Dual optimal variables of "$z = Ax - b$" still give $\tilde{\alpha}$
- Convergence theory basically the same as in (Frangioni 2002)

Summary of Approaches

- I^+: Cutting-plane with exponential number of constraints, but easy separation
- *StabDW*: Bundle for *DW* with exponential number of variables, but easy pricing
- *StructDW*: Structured DW for $LP(B^+)$ with pseudo-polynomial number of variables and constraints
- S_2DW_2: Stabilized Structured DW with quadratic penalty
- S_2DW_1: Stabilized Structured DW with trust region
- S_2DW_1–ws^2: Stabilized Structured DW with trust region and subgradient optimization warmstart

Computational experiments

- Large-scale instances ($|K| \in \{100, 200, 400\}$), very difficult
- $C = 1 \Rightarrow$ lightly capacitated, $C = 16 \Rightarrow$ tightly capacitated
- Solving the root relaxation, then freezing the formulation
  $+$ `CPLEX` polishing for one hour
- Unlike $I+$, frozen $B+$ formulations may not contain optimal solution
  $\Rightarrow$ final gap $\approx$ quality of obtained formulation
- imp $=$ lower bound improvement (equal for all)
  gap $=$ final gap (%), cpu $=$ time, it $=$ iterations

Sample computational results ($|K| = 100$)

Problem			$I+$			StabDW		StructDW		
$\|A\|$	C	imp	cpu	gap	it	cpu	it	cpu	gap	it
517	1	187.00	348	5.78	26	4323	88144	**296**	6.94	55
	4	138.22	362	6.42	25	3581	79390	**312**	7.48	44
	8	100.08	**305**	6.12	21	4054	88807	633	6.11	61
	16	60.49	**249**	6.20	21	3015	71651	1138	6.45	87
517	1	155.19	**140**	3.95	23	2899	69500	188	4.70	60
	4	122.84	194	3.87	26	2799	65229	**147**	4.15	39
	8	93.00	**151**	3.96	20	2824	66025	355	4.31	67
	16	59.68	**116**	4.72	18	2172	56184	551	4.94	70
669	1	114.50	80	0.50	26	330	11273	**36**	0.46	32
	4	97.32	78	0.46	22	327	10951	**66**	0.46	50
	8	79.62	68	0.46	19	323	11173	**55**	0.46	33
	16	56.19	**58**	0.74	19	275	9979	164	0.81	65

Sample computational results ($|K| = 200$)

Problem			$I+$			StabDW		StructDW		
$\lvert A \rvert$	C	imp	cpu	gap	it	cpu	it	cpu	gap	it
229	1	205.67	49081	28.16	109	11748	154821	**525**	10.50	44
	4	131.24	30899	25.40	91	9132	131674	**807**	13.58	45
	8	84.61	16502	21.80	87	12682	162766	**1593**	10.17	44
	16	42.78	**2090**	**5.59**	54	6541	97952	2630	**9.20**	73
229	1	185.17	18326	20.53	86	9261	132963	**380**	7.44	39
	4	125.39	15537	18.81	80	11791	147879	**612**	9.36	49
	8	85.31	9500	13.08	74	10702	146727	**1647**	8.87	68
	16	46.09	**1900**	**7.19**	52	7268	107197	3167	**7.99**	108
287	1	198.87	14559	27.86	66	8815	120614	**598**	12.54	53
	4	136.97	11934	22.52	62	8426	112308	**603**	15.07	37
	8	92.94	9656	15.28	64	10098	130536	**1221**	10.38	41
	16	53.45	3579	11.60	54	6801	98972	**3515**	9.06	99

Sample computational results ($|K| = 400$)

Problem			StabDW		StructDW		
$\|A\|$	C	imp	cpu	it	cpu	gap	it
519	1	100.83	87695	248746	9839	9.96	157
	4	92.54	88031	247864	9087	11.25	140
	8	82.16	88918	258266	11613	8.47	143
	16	65.53	85384	238945	38617	10.26	242
519	1	125.07	93065	258054	22246	14.90	165
	4	111.02	90573	250854	17976	18.22	131
	8	94.82	93418	256884	30460	18.18	159
	16	71.31	93567	265663	74447	16.50	176
668	1	126.02	98789	246702	23771	11.89	149
	4	115.29	99014	247620	28567	10.97	176
	8	102.03	104481	258636	27871	12.07	130
	16	80.96	103011	278905	58363	13.95	156

Some preliminary conclusions

- DW unbearably slow, and disaggregating does not help enough
- Stabilized DW $\equiv$ bundle much better, but only aggregated
- SDW worsens as C grows (tighter capacities), RG the converse
- SDW generally better, but times and gaps are still large $\Rightarrow$ Stabilized SDW seems promising

Computational experiments on stabilized SDW

- No removal/aggregation for $\mathcal{B}$, fixed t (class-specific tuning)
- Different stabilizing terms: quadratic penalty vs trust region (QP vs LP)
- Different warm-start: "standard" MCF initialization (used for all) vs MCF + subgradient warm-start (few iterations, class-specific tuning)
- gap = final gap (%), cpu = time, it = iterations, ss = serious steps

Sample computational results ($|K| = 100$)

	StructDW			S^2DW_2				S^2DW_1				S^2DW_1–ws^2			
C	cpu	gap	it	cpu	gap	it	ss	cpu	gap	it	ss	cpu	gap	it	ss
1	296	6.94	55	16380	6.57	51	15	**223**	2.97	66	58	357	1.52	91	84
4	312	7.48	44	17091	5.87	47	12	298	2.72	70	54	**270**	1.48	69	60
8	633	6.11	61	22176	7.16	37	14	280	2.70	64	34	**277**	1.44	65	47
16	1138	6.45	87	27033	6.08	43	18	190	2.78	60	21	**119**	1.52	40	18
1	**188**	4.70	60	5802	4.01	42	13	205	2.56	71	57	222	1.43	85	71
4	147	4.15	39	6453	4.32	39	15	215	2.43	79	40	**91**	1.39	41	36
8	354	4.31	67	5752	4.40	31	12	167	2.38	62	25	**124**	1.42	50	21
16	551	4.94	70	10154	5.07	40	14	163	2.76	61	20	**113**	1.53	50	19
1	**36**	0.46	32	2405	0.46	47	15	84	0.41	76	48	78	0.33	72	66
4	**66**	0.46	50	1964	0.46	45	14	67	0.41	74	24	81	0.33	73	56
8	55	0.46	33	1974	0.46	44	15	50	0.41	57	18	**40**	0.33	49	20
16	164	0.81	65	1408	0.80	38	17	47	0.61	52	16	**44**	0.40	52	22

Sample computational results ($|K| = 200$)

	StructDW			S^2DW_2				S^2DW_1				S^2DW_1–ws^2			
C	cpu	gap	it	cpu	gap	it	ss	cpu	gap	it	ss	cpu	gap	it	ss
1	**525**	10.50	44	1.8e4	12.11	32	17	860	4.16	76	73	907	1.32	129	119
4	**807**	13.58	45	2.7e4	10.20	29	15	1091	2.79	89	87	1460	1.23	126	118
8	1593	10.17	44	8.3e4	10.12	40	17	**1027**	3.03	78	61	1237	1.20	99	77
16	2630	9.20	73	1.1e5	9.21	54	16	**399**	2.12	65	31	804	1.02	114	73
1	**380**	7.44	39	1.0e4	****	29	14	557	2.61	80	71	592	1.30	101	95
4	**612**	9.36	49	1.3e4	10.33	25	15	755	2.87	80	68	930	1.22	98	95
8	1647	8.87	68	3.3e4	10.61	30	14	**468**	2.75	50	43	761	1.33	83	66
16	3167	7.99	108	7.0e4	8.32	47	17	**476**	2.22	67	30	357	1.10	53	39
1	**598**	12.54	53	2.1e4	16.31	39	15	1019	3.92	98	93	1327	1.65	149	143
4	**603**	15.07	37	1.8e4	13.78	27	15	1001	3.72	90	79	891	1.60	98	94
8	1221	10.38	41	5.2e4	11.81	29	14	**909**	3.68	73	50	1040	1.63	102	96
16	3515	9.06	99	1.3e5	10.11	54	17	**513**	2.93	59	25	555	1.26	62	45

Sample computational results ($|K| = 400$)

	StructDW			S^2DW_1				S^2DW_1–ws^2			
C	cpu	gap	it	cpu	gap	it	ss	cpu	gap	it	ss
1	9839	9.96	157	2473	2.23	76	55	**1857**	2.31	53	38
4	9087	11.25	140	**2140**	2.33	68	54	2487	2.36	66	44
8	11613	8.47	143	2338	2.45	66	45	**1813**	2.30	52	30
16	38617	10.26	242	3403	2.66	77	39	**2570**	2.26	58	23
1	22246	14.90	165	4811	3.31	87	76	**4668**	3.06	66	55
4	17976	18.22	131	**4324**	2.57	77	64	4373	3.19	66	45
8	30460	18.18	159	5224	3.14	85	60	**4209**	2.86	57	36
16	74447	16.50	176	5532	3.14	67	46	**5191**	3.02	64	23
1	23771	11.89	149	9215	2.96	97	78	**6815**	3.01	69	56
4	28567	10.97	176	6766	2.99	79	63	**6506**	3.07	69	45
8	27871	12.07	130	7560	2.67	87	56	**5765**	2.78	61	37
16	58363	13.95	156	8626	3.14	83	45	**3764**	2.95	41	18

Current research and future trends

- Adaptive network design (Gendron, Semet 2009)
- Integrating uncertainty: stochastic programming (Crainic, Gendreau, Rei, Wallace 2009)
- Decentralized / collaborative network design
- On the methodological side:
  - Alternative formulations based on paths/circuits and (multi)-cutsets
  - Decomposition methods involving column and cut generation within B&B: B&C&P
  - Hybrid algorithms combining mathematical programming and metaheuristics
  - Parallel computing (especially for large-scale adaptive and stochastic network design)