Not-only-column Generation: the (Stabilized) Structured Dantzig-Wolfe Method

Antonio Frangioni

Dipartimento di Informatica, Università di Pisa

joint work with

Bernard Gendron

CIRRELT, Université de Montréal

ColGen 2012, Bromont
June 11^{th}, 2012

Outline

1. A (not uncommon) Tale of Modeling and Reformulations
   - Integer Formulation
   - Row Generation
   - (Stabilized) Column Generation
   - Computational results: Row vs. (Stabilized) Column Generation

Outline

1. A (not uncommon) Tale of Modeling and Reformulations
   - Integer Formulation
   - Row Generation
   - (Stabilized) Column Generation
   - Computational results: Row vs. (Stabilized) Column Generation
2. Binary formulation: Row & Column Generation

Outline

Outline

1. A (not uncommon) Tale of Modeling and Reformulations
   - Integer Formulation
   - Row Generation
   - (Stabilized) Column Generation
   - Computational results: Row vs. (Stabilized) Column Generation
2. Binary formulation: Row & Column Generation
3. The Structured Dantzig-Wolfe Approach
4. Computational results for StructDW

Outline

Outline

1. A (not uncommon) Tale of Modeling and Reformulations
   - Integer Formulation
   - Row Generation
   - (Stabilized) Column Generation
   - Computational results: Row vs. (Stabilized) Column Generation
2. Binary formulation: Row & Column Generation
3. The Structured Dantzig-Wolfe Approach
4. Computational results for StructDW
5. Stabilizing the Structured Dantzig-Wolfe Algorithm
6. Computational results for S^2DW

Outline

Multicommodity Capacitated Network Design

- Multiple flows (commodities) (s^k, t^k, d^k) $k \in K$, facility costs f_{ij}
- Standard integer formulation I ($\bar{I}$ = continuous relaxation)

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

$$\sum_{(i,j) \in A} u^k_{ij} - \sum_{(j,i) \in A} u^k_{ji} = \begin{cases} 1 & \text{if } i = s^k \\ -1 & \text{if } i = t^k \\ 0 & \text{otherwise} \end{cases} \qquad i \in N \, , \; k \in K$$

$$\sum_{k \in K} d^k u^k_{ij} \leq a_{ij} y_{ij} \qquad (i,j) \in A$$

$$0 \leq x^k_{ij} \leq 1 \qquad (i,j) \in A \, , \; k \in K$$

$$y_{ij} \in \mathbb{N} \qquad (i,j) \in A$$

Multicommodity Capacitated Network Design

- Multiple flows (commodities) (s^k, t^k, d^k) $k \in K$, facility costs f_{ij}
- Standard integer formulation I ($\bar{I}$ = continuous relaxation)

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

$$\sum_{(i,j) \in A} u^k_{ij} - \sum_{(j,i) \in A} u^k_{ji} = \begin{cases} 1 & \text{if } i = s^k \\ -1 & \text{if } i = t^k \\ 0 & \text{otherwise} \end{cases} \qquad i \in N \ , \ k \in K$$

$$\sum_{k \in K} d^k u^k_{ij} \leq a_{ij} y_{ij} \qquad (i,j) \in A$$

$$0 \leq x^k_{ij} \leq 1 \qquad (i,j) \in A \ , \ k \in K$$

$$y_{ij} \in \mathbb{N} \qquad (i,j) \in A$$

- Efficiently optimize on mutiflows + construct the graph
- $\mathcal{NP}$-hard, loads of applications, very difficult in practice because large-scale (= slow) relaxation but weak bound from $\bar{I}$

Reformulation I: Polyhedral Methods = Row Generation

- (Exponentially many) Residual capacity inequalities [Atamturk, 2002]

$$\sum_{k \in S} a_k(1 - u_{ij}^k) \geq (a(S) - \lfloor a(S) \rfloor)(\lceil a(S) \rceil - y_{ij}) \quad S \subseteq K \qquad (1)$$

  $(a_k = d^k / a_{ij},\ a(S) = \sum_{k \in S} a_k)$

- Separation easy ($\approx$ 2 continuous knapsack), bound improves
- Standard B&C tools
- Re-solve $\bar{I}$ (large already) many times per node

Reformulation II: Lagrangian Dual = Column Generation

- Relax the flow conservation constraints, (many) multipliers $x = [x_i^k]$
- Lagrangian Relaxation decomposes by arc

$$\min \quad \sum_{(i,j)\in A} \Big(\sum_{k\in K} (d^k c_{ij}^k - x_i^k + x_j^k) u_{ij}^k + f_{ij} y_{ij} \Big)$$

$$U_{ij} \equiv \left\{ \begin{array}{ll} \sum_{k\in K} d^k u_{ij}^k \leq a_{ij} y_{ij} & \\ 0 \leq u_{ij}^k \leq 1 & k \in K \\ y_{ij} \in \mathbb{N} & \end{array} \right\} \qquad (i,j) \in A$$

- Easy ($\approx$ 2 continuous knapsack) but no integrality property
  $\Rightarrow$ better bound than continuous relaxation
- Residual capacity inequalities (1) have $\approx$ separation cost and describe $conv(U_{ij})$ [Atamturk, 2002] $\Rightarrow$ $v(LD) = v(\bar{I}+)$
- Have to find optimal multipliers x^*

Lagrangian Dual = Column Generation = Dantzig-Wolfe

- Compact notation: decomposable $U = \times_{k \in K} U^k$, $u = [u^k]_{k \in K}$

$$(\Pi) \qquad \max \{ \, cu \; : \; Au = b \, , \; u \in conv(U) \, \}$$

- We can efficiently optimize upon $U \Rightarrow$ generate vertices of $U \Rightarrow$ represent $conv(U)$ by extreme points ($\bar{U} = \operatorname{ext} U$) instead of by faces

$$conv(U) = \left\{ \, u = \sum_{\bar{u} \in \bar{U}} \bar{u}\theta_{\bar{u}} \; : \; \sum_{\bar{u} \in \bar{U}} \theta_{\bar{u}} = 1 \, , \; \theta_{\bar{u}} \geq 0 \quad \bar{u} \in \bar{U} \, \right\}$$

$\Rightarrow$ reformulate (Π) in terms of the convex multipliers θ

$$\begin{array}{lll} \max & c \left(\sum_{\bar{u} \in \bar{U}} \bar{u}\theta_{\bar{u}} \right) & \\ & A \left(\sum_{\bar{u} \in \bar{U}} \bar{u}\theta_{\bar{u}} \right) & = b \\ & \sum_{\bar{u} \in \bar{U}} \theta_{\bar{u}} & = 1 \quad \theta_{\bar{u}} \geq 0 \quad \bar{u} \in \bar{U} \end{array}$$

- Too large to be solved directly $\Rightarrow$ Column Generation

Master Problems

- $\mathcal{B} \subset \bar{U}$ (small), solve restriction of (Π) with $\bar{U} \to \mathcal{B}$, i.e.,

$$(\Pi_{\mathcal{B}}) \qquad \max \{ \, cu \; : \; Au = b \, , \; u \in conv(\mathcal{B}) \, \}$$

feed (partial) dual optimal solution x^* (of $Au = b$) to pricing problem

$$(\Pi_x) \qquad f(x) = \max \{ \, (c - xA)u \; : \; u \in U \, \} \; + \; xb$$

($\equiv$ compute Lagrangian function $f(x)$) to get new $\bar{u} \in \bar{U}$, $\bar{u} \to \mathcal{B}$

- Dual of $(\Pi_{\mathcal{B}})$: $\quad \min \{ \, f_{\mathcal{B}}(x) = \max \{ \, (c - xA)u + xb \, , \; u \in \mathcal{B} \, \} \, \}$
$f_{\mathcal{B}}(x)$ = lower approximation of "true" Lagrangian function $f(x)$
= cutting-plane model

- Column Generation $\equiv$ Row Generation in the dual $\equiv$
solve Lagrangian Dual by Cutting Plane method

Better Master Problems I: Disaggregation

- Better: disaggregated primal master problem

$$\max \left\{ \sum_{k \in K} c^k u^k \;:\; \sum_{k \in K} A^k u^k = b \;,\;\; u^k \in U^k_{\mathcal{B}} = conv(\mathcal{B}^k) \;\; k \in K \right\}$$

(in practice, a different multiplier $\theta^k_{\bar{u}}$ for each $\bar{u}^k$, previously $\theta^k_{\bar{u}} = \theta^h_{\bar{u}}$)
$\equiv$ disaggregated cutting-plane model

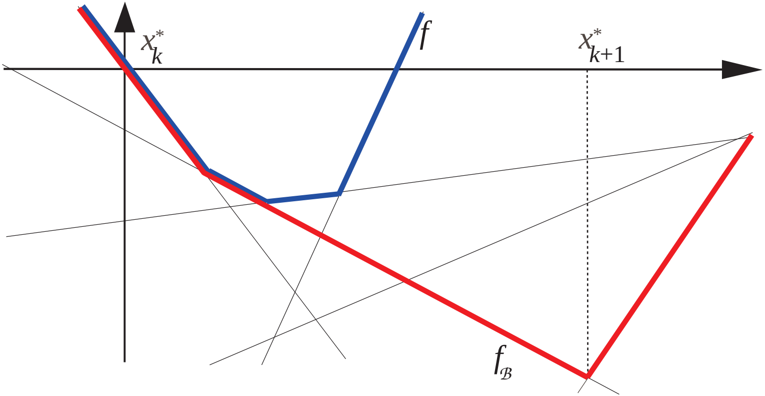
$$f_{\mathcal{B}}(x) = xb \;+\; \textstyle\sum_{k \in K} \left(\; f^k_{\mathcal{B}}(x) = \max \; \{ \; (c^k - xA^k)u^k \;:\; u^k \in U^k_{\mathcal{B}} \; \} \; \right)$$

- $|K|$ times larger master problem, but better use of information
  $\Rightarrow$ faster convergence (e.g. [Jones et al. 1993] for multicommodity)

- Convergence can be slow, less well-supported than Row Generation, a few nontrivial issues (branching, ...)

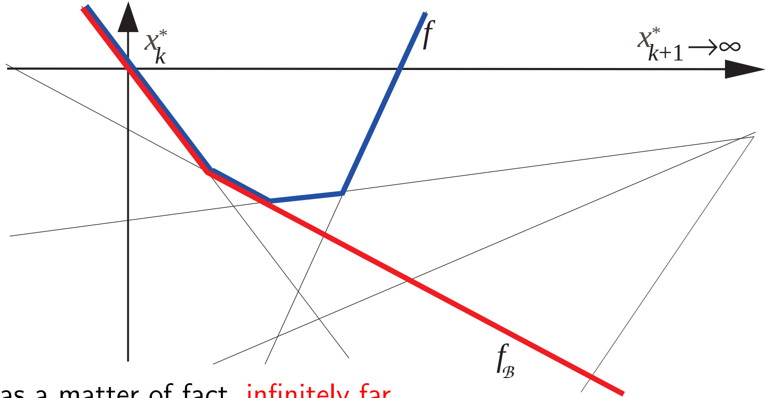
Instability in CG/DW

- x^*_{k+1} can be very far from x^*_k, where $f_{\mathcal{B}}$ is a "bad model" of f



Instability in CG/DW

- x^*_{k+1} can be very far from x^*_k, where $f_{\mathcal{B}}$ is a "bad model" of f



...as a matter of fact, infinitely far

- $(\Pi_{\mathcal{B}})$ empty $\equiv$ $(\Delta_{\mathcal{B}})$ unbounded $\Rightarrow$ Phase 0 / Phase 1 approach
- More in general: $\{x^*_k\}$ is unstable, has no locality properties
  $\Rightarrow$ convergence speed does not improve near the optimum

Better Master Problems II: Stabilization

- Current point $\bar{x}$, stabilizing term $\mathcal{D}_t \geq 0$, proximal parameter(s) t, stabilized dual problem

$$(\Delta_{\mathcal{B},\bar{x},\mathcal{D},t}) \quad \min \left\{ \, f_{\mathcal{B}}(x) + \mathcal{D}_t(x - \bar{x}) \, \right\}$$

Just avoid that iterates "go too far from $\bar{x}$"

Better Master Problems II: Stabilization

- Current point $\bar{x}$, stabilizing term $\mathcal{D}_t \geq 0$, proximal parameter(s) t, stabilized dual problem

$$(\Delta_{\mathcal{B},\bar{x},\mathcal{D},t}) \quad \min \left\{ f_{\mathcal{B}}(x) + \mathcal{D}_t(x - \bar{x}) \right\}$$

  Just avoid that iterates "go too far from $\bar{x}$"

- Very simple stabilized primal problem

$$\max \left\{ cu + \bar{x}z - \mathcal{D}_t^*(-z) \ : \ z = b - Au \ , \ u \in conv(\mathcal{B}) \right\}$$

  add slacks z, penalize them ("1^{st}-order" and "2^{nd}-order" terms)

- Funny general form for NDO lovers: Fenchel's dual of $(\Delta_{\bar{x},\mathcal{D},t})$

$$- \min \left\{ f^*(z) - z\bar{y} + \mathcal{D}_t^*(-z) \right\}$$

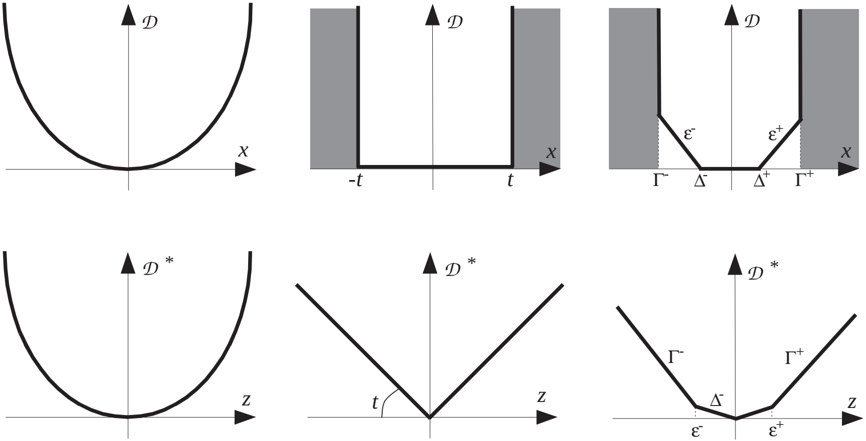
  [F., 2002] "*" = Fenchel's conjugate

Stabilizing Terms

- Few general properties:
  i) $\mathcal{D}_t \geq 0$ convex, $\mathcal{D}_t(0) = 0$ | i) + ii) hold for $\mathcal{D}_t$
  ii) $S_\delta(\mathcal{D}_t)$ compact and full-dimensional $\forall \delta > 0$ | $\iff$ hold for $\mathcal{D}_t^*$
  iii) $\mathcal{D}_t$ differentiable in 0 $\iff$ $\mathcal{D}_t^*$ strictly convex in 0
  iv) $\lim_{\|x\| \to \infty} D_t(x)/\|x\| = +\infty \iff \mathcal{D}_t^* < +\infty$
  v) $\mathcal{D}_t$ ($\mathcal{D}_t^*$) de(inc)reasing in t, $\mathcal{D}_t \to 0$ ($\mathcal{D}_t^* \to I_{\{0\}}$) as $t \to \infty$
- iv) only serve to have $(\Delta_{\bar{x},\mathcal{D},t})$ bounded (other means possible)
  iii) can be relaxed somewhat, albeit at a cost
- Simple and robust choice: $\| \cdot \|_2^2$ [Lemaréchal et al, 2006]
- Reasonable choices: piecewise-linear functions $\Rightarrow$ $(\Delta_{\bar{x},\mathcal{D},t})$ is a LP
  - 1-piece = boxstep [Marsten et al, 1975]
  - 2-pieces [Kim et al, 1995]
  - 3-pieces [Du Merle et al, 1999]
  - 5-pieces [Ben Hamor et al, 2009]

In practice

a penalty | a trust region | or both



$\mathcal{D}_t = \frac{1}{2t}\|\cdot\|_2^2$

$\mathcal{D}_t^* = \frac{1}{2}t\|\cdot\|_2^2$

$\mathcal{D}_t = I_{B_\infty(t)}$

$\mathcal{D}_t^* = t\|\cdot\|_1$

$\mathcal{D}_{\Gamma^\pm,\Delta^\pm,\varepsilon^\pm} = \ldots$

$\mathcal{D}^*_{\Gamma^\pm,\Delta^\pm,\varepsilon^\pm} = \ldots$

Computational results: RG vs. StabDW

- Intel Xeon X7350@2.93GHz, 64Gb RAM, Suse Linux, `CPLEX 11.1`
- Large-scale instances ($|K| \in \{100, 200, 400\}$), very difficult
- $C = 1 \Rightarrow$ lightly capacitated, $C = 16 \Rightarrow$ tightly capacitated

Computational results: RG vs. StabDW

- Intel Xeon X7350@2.93GHz, 64Gb RAM, Suse Linux, `CPLEX 11.1`
- Large-scale instances ($|K| \in \{100, 200, 400\}$), very difficult
- $C = 1 \Rightarrow$ lightly capacitated, $C = 16 \Rightarrow$ tightly capacitated
- DW unbearably slow, disaggregating does not help (enough)
- Stabilized DW much better, but only if disaggregated

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

Problem			$I+$		StabDW	
$\lvert A\rvert$	C	imp	cpu	it	cpu	it
517	1	187.00	348	26	4323	88144
	4	138.22	362	25	3581	79390
	8	100.08	305	21	4054	88807
	16	60.49	249	21	3015	71651
517	1	155.19	140	23	2899	69500
	4	122.84	194	26	2799	65229
	8	93.00	151	20	2824	66025
	16	59.68	116	18	2172	56184
669	1	114.50	80	26	330	11273
	4	97.32	78	22	327	10951
	8	79.62	68	19	323	11173
	16	56.19	58	19	275	9979

- RG always better than StabCG

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

Problem			$I+$		StabDW	
$\|A\|$	C	imp	cpu	it	cpu	it
229	1	205.67	49081	109	11748	154821
	4	131.24	30899	91	9132	131674
	8	84.61	16502	87	12682	162766
	16	42.78	2090	54	6541	97952
229	1	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	1	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

- RG wins only for large C, 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-polinomially many segments $S_{ij} = \{ 1, \ldots, T_{ij} \}$ for y_{ij}

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-polinomially many segments $S_{ij} = \{ 1, \ldots, T_{ij} \}$ for y_{ij}
- Reformulation in binary variables: $y_{ij} = \sum_{s \in S_{ij}} y_{ij}^s$

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

$$u_{ij}^{ks} = \begin{cases} u_{ij}^k & \text{if } y_{ij} = s \\ 0 & \text{otherwise} \end{cases} \qquad s \in S_{ij} \ , \ k \in K$$

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

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

... then original variables can be removed

- Up to now, continuous relaxation bound has not improved

Improved binary formulation $B+$

- Extended linking inequalities:

$$u_{ij}^{ks} \leq y_{ij}^{s} \qquad (i,j) \in A \;\;,\;\; k \in K \;\;,\;\; s \in S_{ij}$$

- Improved continuous relaxation bound: $v(\bar{B}+) = v(\bar{I}+) = v(DW)$
[F., Gendron, 2009] using [Croxton et al., 2003]

Improved binary formulation $B+$

- Extended linking inequalities:

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

- Improved continuous relaxation bound: $v(\bar{B}+) = v(\bar{I}+) = v(DW)$ [F., Gendron, 2009] using [Croxton et al., 2003]

- In particular, binary formulation describes $conv(U)$: continuous relaxation has integrality property

- Optimizing over $U \Rightarrow conv(U)$ easy

Improved binary formulation $B+$

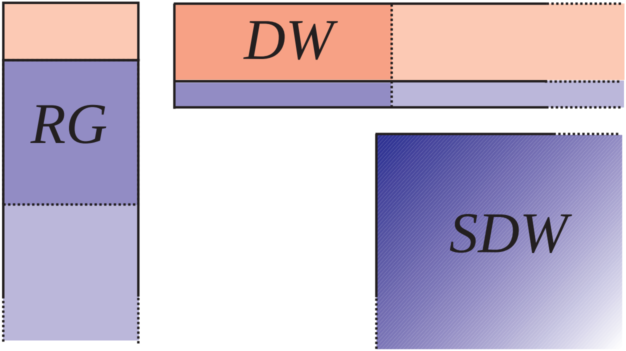
- Extended linking inequalities:

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

- Improved continuous relaxation bound: $v(\bar{B}+) = v(\bar{I}+) = v(DW)$ [F., Gendron, 2009] using [Croxton et al., 2003]
- In particular, binary formulation describes $conv(U)$: continuous relaxation has integrality property
- Optimizing over $U \Rightarrow conv(U)$ easy
- Pseudo-polynomial number of variables and constraints
- How can we exploit it?

The main issue

- Substantially different from both RG and DW



- Need to generate both rows and columns

The Structured Dantzig-Wolfe Idea

- Assumption 1: Alternative Formulation of "easy" set

$$conv(U) = \{ \ u = C\theta \ : \ \Gamma\theta \leq \gamma \ \}$$

The Structured Dantzig-Wolfe Idea

- Assumption 1: Alternative Formulation of "easy" set

$$conv(U) = \{ \; u = C\theta \; : \; \Gamma\theta \leq \gamma \; \}$$

- Assumption 2: padding with zeroes

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

$$\Rightarrow U_{\mathcal{B}} = \left\{ \; u = C_{\mathcal{B}}\theta_{\mathcal{B}} \; : \; \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \gamma_{\mathcal{B}} \; \right\} \subseteq conv(U)$$

The Structured Dantzig-Wolfe Idea

- Assumption 1: Alternative Formulation of "easy" set

$$conv(U) = \{\ u = C\theta\ :\ \Gamma\theta \leq \gamma\ \}$$

- Assumption 2: padding with zeroes

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

$$\Rightarrow U_{\mathcal{B}} = \left\{\ u = C_{\mathcal{B}}\theta_{\mathcal{B}}\ :\ \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \gamma_{\mathcal{B}}\ \right\} \subseteq conv(U)$$

- Assumption 3: easy update of rows and columns

Given $\mathcal{B}$, $\bar{u} \in conv(U)$, $\bar{u} \notin U_{\mathcal{B}}$, it is "easy" to find $\mathcal{B}' \supset \mathcal{B}$ ($\Rightarrow \Gamma_{\mathcal{B}'}$, $\gamma_{\mathcal{B}'}$) such that $\exists\ \mathcal{B}'' \supseteq \mathcal{B}'$ such that $\bar{u} \in U_{\mathcal{B}''}$.

The Structured Dantzig-Wolfe Idea

- Assumption 1: Alternative Formulation of "easy" set

$$conv(U) = \{\ u = C\theta\ :\ \Gamma\theta \leq \gamma\ \}$$

- Assumption 2: padding with zeroes

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

$$\Rightarrow U_{\mathcal{B}} = \left\{\ u = C_{\mathcal{B}}\theta_{\mathcal{B}}\ :\ \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \gamma_{\mathcal{B}}\ \right\} \subseteq conv(U)$$

- Assumption 3: easy update of rows and columns

Given $\mathcal{B}$, $\bar{u} \in conv(U)$, $\bar{u} \notin U_{\mathcal{B}}$, it is "easy" to find $\mathcal{B}' \supset \mathcal{B}$

($\Rightarrow \Gamma_{\mathcal{B}'}, \gamma_{\mathcal{B}'}$) such that $\exists\ \mathcal{B}'' \supseteq \mathcal{B}'$ such that $\bar{u} \in U_{\mathcal{B}''}$.

- Structured master problem

$$(\Pi_{\mathcal{B}}) \qquad \max\left\{\ cu\ :\ Au = b\ ,\ u = C_{\mathcal{B}}\theta_{\mathcal{B}}\ ,\ \ \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \gamma_{\mathcal{B}}\ \right\}$$

$\equiv$ structured model

$$f_{\mathcal{B}}(x) \ = \ \max\{\ (c - xA)u + ub\ ,\ u = C_{\mathcal{B}}\theta_{\mathcal{B}}\ ,\ \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \gamma_{\mathcal{B}}\ \}$$

The Structured Dantzig-Wolfe Algorithm

$\langle$ initialize $\mathcal{B}$ $\rangle$;
```
repeat
```
  $\langle$ solve $(\Pi_{\mathcal{B}})$ for u^*, x^* (duals of $Au = b$); $v^* = cu^*$ $\rangle$;
  $\bar{u} = \text{argmin} \{ (c - x^*A)u : u \in U \}$;
  $\langle$ update $\mathcal{B}$ as in Assumption 3 $\rangle$;
`until` $v^* < c\bar{u} + x^*(b - A\bar{u})$

The Structured Dantzig-Wolfe Algorithm

```
⟨ initialize 𝓑 ⟩;
repeat
    ⟨ solve (Π_𝓑) for u*, x* (duals of Au = b); v* = cu* ⟩;
    ū = argmin { (c − x*A)u : u ∈ U };
    ⟨ update 𝓑 as in Assumption 3 ⟩;
until v* < cū + x*(b − Aū)
```

- Relatively easy [F., Gendron, 2009] to prove that:
  - finitely terminates with an optimal solution of (Π)
  - …even if (proper) removal from $\mathcal{B}$ is allowed (when cu^* increases)
  - …even if U is non compact and $\mathcal{B} = \emptyset$ at start (Phase 0)

The Structured Dantzig-Wolfe Algorithm

$\langle$ initialize $\mathcal{B}$ $\rangle$;
`repeat`
  $\langle$ solve $(\Pi_{\mathcal{B}})$ for u^*, x^* (duals of $Au = b$); $v^* = cu^*$ $\rangle$;
  $\bar{u} = \text{argmin } \{ (c - x^* A)u : u \in U \}$;
  $\langle$ update $\mathcal{B}$ as in Assumption 3 $\rangle$;
`until` $v^* < c\bar{u} + x^*(b - A\bar{u})$

- Relatively easy [F., Gendron, 2009] to prove that:
  - finitely terminates with an optimal solution of (Π)
  - ...even if (proper) removal from $\mathcal{B}$ is allowed (when cu^* increases)
  - ...even if U is non compact and $\mathcal{B} = \emptyset$ at start (Phase 0)
- The subproblem to be solved is identical to that of DW
- Requires ($\Rightarrow$ exploits) extra information on the structure
- Master problem with any structure, possibly much larger

Computational results for StructDW

- Same machine/instances as before
- 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

- SDW worsens as C grows (tighter capacities), RG the converse

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

Problem			$l+$			StabDW		StructDW		
$\|A\|$	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

- Same trend, but RG better only for $C = 16$

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

- SWD always better, stabilizing SDW seems promising

Stabilizing the Structured Dantzig-Wolfe Algorithm

- Exactly the same as stabilizing DW: stabilized master problem

$$(\Delta_{\mathcal{B},\bar{x},\mathcal{D},t}) \qquad \min \left\{ f_{\mathcal{B}}(x) + \mathcal{D}_t(x - \bar{x}) \right\}$$

except $f_{\mathcal{B}}$ is a different model of f (not the cutting plane one)

Stabilizing the Structured Dantzig-Wolfe Algorithm

- Exactly the same as stabilizing DW: stabilized master problem

$$(\Delta_{\mathcal{B},\bar{x},\mathcal{D},t}) \qquad \min\left\{\ f_{\mathcal{B}}(x) + \mathcal{D}_t(x-\bar{x})\ \right\}$$

  except $f_{\mathcal{B}}$ is a different model of f (not the cutting plane one)

- Even simpler from the primal viewpoint:

$$\max\left\{\ cu + \bar{x}z - \mathcal{D}^*(-z)\ :\ z = b - Au\ ,\ u = C_{\mathcal{B}}\theta_{\mathcal{B}}\ ,\ \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \gamma_{\mathcal{B}}\ \right\}$$

- With proper choice of $\mathcal{D}_t$, still a Linear Program; e.g.

$$\begin{array}{rl} \max & \ldots - (\Delta^- + \Gamma^-)z_2^- - \Delta^- z_1^- - \Delta^+ z_1^+ - (\Delta^+ + \Gamma^+)z_2^+ \\ & z_2^- + z_1^- - z_1^+ - z_2^+ = b - Au\ ,\ \ldots \\ & z_2^+ \geq 0\ ,\ \varepsilon^+ \geq z_1^+ \geq 0\ ,\ \varepsilon^- \geq z_1^- \geq 0\ ,\ z_2^- \geq 0 \end{array}$$

- Dual optimal variables of "$z = b - Au$" still give x^*

Stabilizing the Structured Dantzig-Wolfe Algorithm

- Exactly the same as stabilizing DW: stabilized master problem

$$(\Delta_{\mathcal{B},\bar{x},\mathcal{D},t}) \qquad \min \left\{ \ f_{\mathcal{B}}(x) + \mathcal{D}_t(x - \bar{x}) \ \right\}$$

  except $f_{\mathcal{B}}$ is a different model of f (not the cutting plane one)

- Even simpler from the primal viewpoint:

$$\max \left\{ \ cu + \bar{x}z - \mathcal{D}^*(-z) \ : \ z = b - Au \ , \ u = C_{\mathcal{B}}\theta_{\mathcal{B}} \ , \ \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \gamma_{\mathcal{B}} \ \right\}$$

- With proper choice of $\mathcal{D}_t$, still a Linear Program; e.g.

$$\begin{array}{ll} \max & \ldots - (\Delta^- + \Gamma^-)z_2^- - \Delta^- z_1^- - \Delta^+ z_1^+ - (\Delta^+ + \Gamma^+)z_2^+ \\ & z_2^- + z_1^- - z_1^+ - z_2^+ = b - Au \ , \ \ldots \\ & z_2^+ \geq 0 \ , \ \varepsilon^+ \geq z_1^+ \geq 0 \ , \ \varepsilon^- \geq z_1^- \geq 0 \ , \ z_2^- \geq 0 \end{array}$$

- Dual optimal variables of "$z = b - Au$" still give x^*
- Convergence theory basically the same as in [F., 2002]
  even somewhat simpler because $\mathcal{B}$ is inherently finite
- NS/SS decision, handling of t, handling of $\mathcal{B}$

Aggregation & the Structured Dantzig-Wolfe Algorithm

- Aggregation is $\mathcal{B} = \mathcal{B} \cup \{ u^* \}$ ($\mathcal{B} = \{ u^* \} \equiv$ "poorman" method)
- Aggregation is contrary to the spirit of S^2DW, anyway it is impossible

Aggregation & the Structured Dantzig-Wolfe Algorithm

- Aggregation is $\mathcal{B} = \mathcal{B} \cup \{\ u^*\ \}$ ($\mathcal{B} = \{\ u^*\ \} \equiv$ "poorman" method)
- Aggregation is contrary to the spirit of S^{2}DW, anyway it is impossible
  ... or is it? Actually, not!
- $\bar{f}_{\mathcal{B}} \ = \ \max\{\ f_{\mathcal{B}}\ ,\ f_{u^*}(x) = cu^* + x(b - Au^*)\ \}$ is a model of f

Aggregation & the Structured Dantzig-Wolfe Algorithm

- Aggregation is $\mathcal{B} = \mathcal{B} \cup \{ u^* \}$ ($\mathcal{B} = \{ u^* \} \equiv$ "poorman" method)
- Aggregation is contrary to the spirit of S^{2}DW, anyway it is impossible . . . or is it? Actually, not!
- $\bar{f}_{\mathcal{B}} = \max\{ f_{\mathcal{B}} , f_{u^*}(x) = cu^* + x(b - Au^*) \}$ is a model of f
- Stabilized master problem with $\bar{f}_{\mathcal{B}}$

$$\max \left\{ \begin{array}{l} cu + (1-\rho)cu^* + \bar{x}z - \mathcal{D}_t^*(-z) \\ u = C_{\mathcal{B}}\theta_{\mathcal{B}} \ , \ \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \rho\gamma_{\mathcal{B}} \\ z = Au + (1-\rho)Au^* - b \ , \ \ \rho \in [0,1] \end{array} \right.$$

  if $conv(U)$ compact and constraints linear

- "Knob": $\rho = 0 \Rightarrow \gamma_{\mathcal{B}} = 0 \Rightarrow u = u^*$, $\rho = 1 \Rightarrow u \in U_{\mathcal{B}}$

Aggregation & the Structured Dantzig-Wolfe Algorithm

- Aggregation is $\mathcal{B} = \mathcal{B} \cup \{\ u^*\ \}$ ($\mathcal{B} = \{\ u^*\ \} \equiv$ "poorman" method)
- Aggregation is contrary to the spirit of S^{2}DW, anyway it is impossible ...or is it? Actually, not!
- $\bar{f}_{\mathcal{B}}\ =\ \max\{\ f_{\mathcal{B}}\ ,\ f_{u^*}(x) = cu^* + x(b - Au^*)\ \}$ is a model of f
- Stabilized master problem with $\bar{f}_{\mathcal{B}}$

$$\max \left\{ \begin{array}{l} cu + (1-\rho)cu^* + \bar{x}z - \mathcal{D}^*_t(-z) \\ u = C_{\mathcal{B}}\theta_{\mathcal{B}}\ ,\ \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \rho\gamma_{\mathcal{B}} \\ z = Au + (1-\rho)Au^* - b\ \ ,\ \ \rho \in [0,1] \end{array} \right.$$

  if $conv(U)$ compact and constraints linear
- "Knob": $\rho = 0\ \Rightarrow\ \gamma_{\mathcal{B}} = 0\ \Rightarrow\ u = u^*$, $\rho = 1\ \Rightarrow\ u \in U_{\mathcal{B}}$
- Possible use: avoid Phase 0 when $\mathcal{D}_t$ "not steep" given $u^* \in conv(U)$ (e.g. $u^* \in U$) such that $Au^* = b$

Computational results

- Same machine/instances as before
- Comparing SDW with S^{2}DW
- No removal/aggregation for $\mathcal{B}$, fixed t (class-specific tuning)
- Different stabilizing terms: $\mathcal{D}_t = \frac{1}{2t}\| \cdot \|_2^2$ vs $\mathcal{D}_t = I_{B_\infty(t)}$ (QP vs LP, Lemaréchal vs Marsten)
- 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_∞				S^2DW_∞–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

- S^2DW_2 converges faster but slow, ws^2 best in gap and often time

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

	StructDW			S^2DW_2				S^2DW_∞				S^2DW_∞–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

- S^2DW_2 exceedingly slow, ws^2 best in gap, not always time

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

	StructDW			S^2DW_∞				S^2DW_∞–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

- SDW always slower, ws^2 most often faster, S^2DW gaps much better

Conclusions

- Column Generation very useful, too often too slow
- Stabilized CG helpful, too often not enough

Conclusions

- Column Generation very useful, too often too slow
- Stabilized CG helpful, too often not enough
- Structured Dantzig-Wolfe another item in our bag-of-tricks:
  generate structures at a finer granularity than columns

Conclusions

- Column Generation very useful, too often too slow
- Stabilized CG helpful, too often not enough
- Structured Dantzig-Wolfe another item in our bag-of-tricks: generate structures at a finer granularity than columns
- Stabilizing SDW possible, little theoretical issues
- Implementation non straightforward but possible
- Computational results quite promising

Conclusions

- Column Generation very useful, too often too slow
- Stabilized CG helpful, too often not enough
- Structured Dantzig-Wolfe another item in our bag-of-tricks:
  generate structures at a finer granularity than columns
- Stabilizing SDW possible, little theoretical issues
- Implementation non straightforward but possible
- Computational results quite promising
- To do: implement generic version (`FiOracle` class)
- To do: application to other interesting problems
- To do: something better than `CPLEX` to solve the quadratic version