Automatic Structure Detection for Decomposition-based Methods in Mixed Integer Linear Programs

Mixed Integer Programming Workshop (2018) — Clemson University

Matthew Galati, SAS Institute Inc.
Robert Pratt, SAS Institute Inc.
Taghi Khaniyev, University of Waterloo
Samir Elhedhli, University of Waterloo

SOFTWARE — HISTORY

- Frameworks supporting branch-and-price, as far back as 1994:
  - MINTO, ABACUS, COIN/SYMPHONY, COIN/BCP
  - Issue — user must derive most algorithmic components for their application

INTRODUCTION | SOFTWARE — HISTORY

- Frameworks supporting branch-and-price, as far back as 1994:
  - MINTO, ABACUS, COIN/SYMPHONY, COIN/BCP
  - Issue — user must derive most algorithmic components for their application
- *Generalize and automate* these frameworks:
  - 2000 by Vanderbeck (BaPCod) and Galati/Ralphs (COIN/DIP) — user defines the blocks
  - 2010 by Gamrath/Lübbecke (SCIP/GCG) and Wang/Ralphs (COIN/DIP) — automated detection of blocks

SOFTWARE — HISTORY

- Frameworks supporting branch-and-price, as far back as 1994:
  - MINTO, ABACUS, COIN/SYMPHONY, COIN/BCP
  - Issue — user must derive most algorithmic components for their application
- *Generalize and automate* these frameworks:
  - 2000 by Vanderbeck (BaPCod) and Galati/Ralphs (COIN/DIP) — user defines the blocks
  - 2010 by Gamrath/Lübbecke (SCIP/GCG) and Wang/Ralphs (COIN/DIP) — automated detection of blocks
- SAS DECOMP is the first/only commercial implementation
  - 2012 — user-defined blocks (MPS or PROC OPTMODEL) and automated (network/concomp/set)
  - 2014 — general automated detection of blocks (auto)

AUTOMATED DANTZIG-WOLFE DECOMPOSITION

- **Pros:**
  - (Dual side) often improves the bound obtained by CPM
  - (Primal side) helps to find good integer solutions faster
  - Relaxation separability provides a natural parallelization of the bounding method
  - Isomorphic subproblems can be aggregated to eliminate symmetry
- **Cons:**
  - The work to obtain the bound is much more extensive
  - Convergence of dual space is slow (stabilization techniques can help)
  - Numeric issues mapping from the DW space to the original space
- Question: How do we automatically permute (and *stretch*) the original matrix into SBDF?

$$\begin{pmatrix} D^1 & & & & F^1 \\ & D^2 & & & F^2 \\ & & \ddots & & \vdots \\ & & & D^K & F^K \\ A^1 & A^2 & \cdots & A^K & G \end{pmatrix}$$

Doubly-bordered block-diagonal form (DBDF)

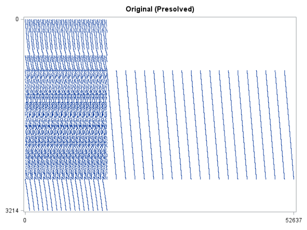
$$\begin{pmatrix} D^1 & & & \\ & D^2 & & \\ & & \ddots & \\ & & & D^K \\ A^1 & A^2 & \cdots & A^K \end{pmatrix}$$

Singly-bordered block-diagonal form (SBDF)

CUSTOMER EXAMPLE — PHARMACEUTICALS

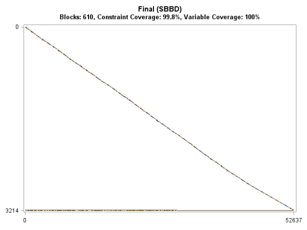
MILP Branch & Cut

```
NOTE: The presolved problem has 52638 variables, 3215 constraints, and 131250 constraint coe
          Node    Active    Sols    BestInteger      BestBound      Gap     Time
             0         1       3    6151.1464478    8590.4503506   28.40%        0
-- snip --
             0         1       4    6151.1466160    7045.9724210   12.70%      782
-- snip --
        175772    173251      11    6871.8766247    7044.1201668    2.45%     3599
NOTE: Real time limit reached.
```



MILP Branch & Price — DECOMP

```
NOTE: The problem has a decomposable structure with 610 blocks.
      The largest block covers 0.2488% of the constraints in the problem.
NOTE: The decomposition subproblems cover 52638 (100%) variables and 3207 (99.75%) constrai
      Iter         Best       Master          Best        LP        IP  CPU  Real
                  Bound    Objective       Integer       Gap       Gap Time  Time
         .    7963.9759    6467.2136     6467.2136    18.79%    18.79%   13     8
         2    7267.7239    6467.2136     6467.2136    11.01%    11.01%   26    13
         3    7147.9955    6878.4375     6467.2136     3.77%     9.52%   51    21
         5    6986.1299    6960.5400     6960.5400     0.37%     0.37%   74    30
         6    6986.1299    6965.5335     6965.5335     0.29%     0.29%   84    33
         7    6972.3310    6972.3309     6972.3309     0.00%     0.00%   87    34
         Node  Active    Sols         Best          Best       Gap     CPU    Real
                                   Integer         Bound              Time    Time
            0       0       9    6972.3309     6972.3310     0.00%      87      34
NOTE: The Decomposition algorithm time is 34.61 seconds.
NOTE: Optimal within relative gap.
```



SOFTWARE — BRANCH & CUT VS BRANCH & PRICE

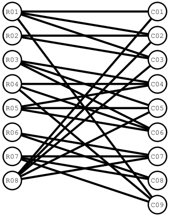
Is automated (black-box) Branch & Price worth pursuing? Yes.

Source	Model/Industry	Instance	Branch & Cut Time	Branch & Cut Gap	Branch & Price Time	Branch & Price Gap	△
MIPLIB/NEOS	Unknown	noswot	17	OPT	0.2	OPT	86x
MIPLIB/NEOS	Unknown	noes_942830	198	OPT	6	OPT	33x
MIPLIB/NEOS	Unknown	ns1208400	252	OPT	35	OPT	7.2x
Research	Machine Reassignment	modela15	103	OPT	15	OPT	6.9x
Research	Cutting Stock	test0055	33	OPT	10	OPT	3.3x
MIPLIB/NEOS	Unknown	neos_885524	523	OPT	169	OPT	3.1x
MIPLIB/NEOS	Unknown	neos_885086	85	OPT	30	OPT	2.8x
Research	Grid Load Management	partition_trim47	261	OPT	99	OPT	2.6x
Customer	Finance	bank_bad	301	OPT	123	OPT	2.4x
Customer	Revenue Management	golfrm_500_471123	12	OPT	5	OPT	2.4x
Customer	Finance	atmorig	T	∞	363	OPT	∞
MIPLIB/NEOS	Unknown	ns3974959	T	∞	257	INF	∞
Customer	Finance	design3_miqp	T	165%	60	OPT	165%
MIPLIB/NEOS	Unknown	chrom_1024	T	33%	984	OPT	33%
MIPLIB/NEOS	Unknown	neos_787933	T	30%	271	OPT	30%
MIPLIB/NEOS	Unknown	ns903616	T	24%	1,650	OPT	24%
MIPLIB/NEOS	Unknown	neos_631694	T	8%	3	OPT	8%
MIPLIB/NEOS	Unknown	neos_799838	T	6%	31	OPT	6%
MIPLIB/NEOS	Unknown	neos_1426662	T	5%	2	OPT	5%
MIPLIB/NEOS	Unknown	neos_826650	T	4%	62	OPT	4%
MIPLIB/NEOS	Unknown	neos_826841	T	4%	71	OPT	4%
MIPLIB/NEOS	Unknown	neos_911880	T	3%	3	OPT	3%
Customer	Pharmaceuticals	pharma	T	2%	949	OPT	2%
MIPLIB/NEOS	Unknown	neos9	T	1%	64	OPT	1%
MIPLIB/NEOS	Unknown	neos19	T	0.3%	283	OPT	0.3%
Research	Resource Allocation	rap_i71	T	0.1%	28	OPT	0.1%
Customer	Hotel Management	roomassign30	T	35%	T	9%	26%
Research	Unit Commitment	unitcommitment	T	26%	T	7%	19%

AUTO DETECTION | APC — AYKANAT ET AL., 2004

- Search for a fixed number $\hat{K}$ of partitions (blocks)
- Hypergraph Model for $A \rightarrow A_{\mathrm{SBDF}}$ — SCIP/GCG uses hMETIS
  - Bergner et al., 2015 — root node only
- Bipartite Graph Model for $A \rightarrow A_{\mathrm{DBDF}} \rightarrow A_{\mathrm{SBDF}}$ — SAS/DECOMP ($\hat{K} =$ NumCores)
  - $A \rightarrow A_{\mathrm{DBDF}}$ — Graph Partitioning with Vertex Separator (GPVS)
  - $A_{\mathrm{DBDF}} \rightarrow A_{\mathrm{SBDF}}$ — Column-splitting (Lagrangian decomposition)
  - APC Goals: minimize border ($\mathcal{S}$) and balance clusters (blocks)
  - Forced balance (good for LP, bad for MILP) — further break down disjoint blocks

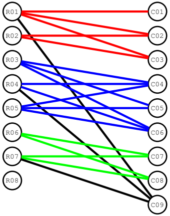
	C01	C02	C03	C04	C05	C06	C07	C08	C09
R01	●	●	●						●
R02		●	●						
R03				●	●	●			
R04				●		●			●
R05				●	●	●			
R06							●	●	
R07							●	●	●
R08	●	●	●		●		●		



APC — AYKANAT ET AL., 2004

- Search for a fixed number $\hat{K}$ of partitions (blocks)
- Hypergraph Model for $A \rightarrow A_{\mathrm{SBDF}}$ — SCIP/GCG uses hMETIS
  - Bergner et al., 2015 — root node only
- Bipartite Graph Model for $A \rightarrow A_{\mathrm{DBDF}} \rightarrow A_{\mathrm{SBDF}}$ — SAS/DECOMP ($\hat{K} =$ NumCores)
  - $A \rightarrow A_{\mathrm{DBDF}}$ — Graph Partitioning with Vertex Separator (GPVS)
  - $A_{\mathrm{DBDF}} \rightarrow A_{\mathrm{SBDF}}$ — Column-splitting (Lagrangian decomposition)
  - APC Goals: minimize border ($\mathcal{S}$) and balance clusters (blocks)
  - Forced balance (good for LP, bad for MILP) — further break down disjoint blocks

	C01	C02	C03	C04	C05	C06	C07	C08	C09
R01	•	•	•						•
R02		•	•						
R03				•	•	•			
R04				•		•			•
R05				•	•	•			
R06							•	•	
R07							•	•	•
R08	•	•	•		•		•		



APC — AYKANAT ET AL., 2004

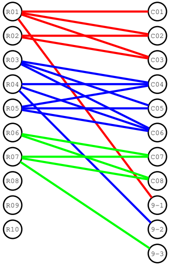
- Search for a fixed number $\hat{K}$ of partitions (blocks)
- Hypergraph Model for $A \rightarrow A_{\mathrm{SBDF}}$ — SCIP/GCG uses hMETIS
  - Bergner et al., 2015 — root node only
- Bipartite Graph Model for $A \rightarrow A_{\mathrm{DBDF}} \rightarrow A_{\mathrm{SBDF}}$ — SAS/DECOMP ($\hat{K}$ = NumCores)
  - $A \rightarrow A_{\mathrm{DBDF}}$ — Graph Partitioning with Vertex Separator (GPVS)
  - $A_{\mathrm{DBDF}} \rightarrow A_{\mathrm{SBDF}}$ — Column-splitting (Lagrangian decomposition)
  - APC Goals: minimize border ($\mathcal{S}$) and balance clusters (blocks)
  - Forced balance (good for LP, bad for MILP) — further break down disjoint blocks

$K = 3$

$B_r = 10.0\%$

$B_c = 20.0\%$

		C01	C02	C03	9-1	C04	C05	C06	9-2	C07	C08	9-3
	R01	●	●	●	●							
	R02		●	●								
	R03					●	●	●				
	R04					●		●	●			
	R05					●	●	●				
	R06									●	●	
	R07									●	●	●
$\mathcal{S}$	R08	●	●	●			●			●		
$\mathcal{S}$	R09				1				−1			
$\mathcal{S}$	R10								1			−1



KEE — KHANIYEV, ET AL., 2018

- Key Idea: Remove the restriction for a fixed number of blocks
- Goals (conflicting objectives):
  - Minimize the border area — same as APC
  - Maximize the *quality* of the diagonal for a fixed border — modularity
- Modularity measures subgraph cohesion compared to a random graph with same degree distribution

$$M(\mathbf{A}^{\pi}) = \sum_{k \in K} \left[\frac{e_k}{m} - \left(\frac{d_k}{2m} \right)^2 \right]$$

- DECOMP/MILP intuition:
  - Subproblems (subsets of constraints) with strong interconnectivity (variables) relative to a random partition

$$Q(\mathbf{A}^{\pi}) = \sum_{k \in K} \frac{e_k}{m} \left(1 - \frac{e_k}{m} \right)$$

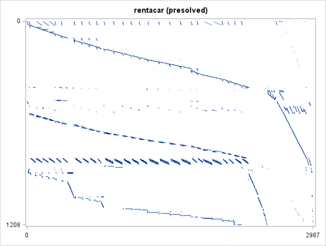
$$P_r(\mathbf{A}^{\pi}, \alpha) = e^{-\alpha B_r}$$

$$P_c(\mathbf{A}^{\pi}, \alpha) = e^{-\alpha B_c}$$

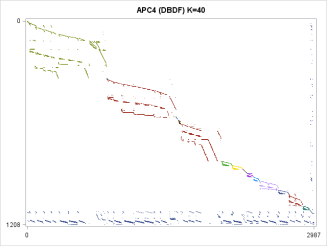
$$\Gamma(\mathbf{A}^{\pi}, \alpha_r, \alpha_c) = Q(\mathbf{A}^{\pi}) P_r(\mathbf{A}^{\pi}, \alpha_r) P_c(\mathbf{A}^{\pi}, \alpha_c)$$

GOODNESS FUNCTION

- $\Gamma \in [0, 1)$
  - $\Gamma = 0$ (single block, or empty diagonal)
  - $\Gamma \to 1$ (empty border, balanced blocks, as $K \uparrow$)

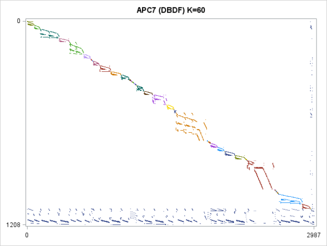


$\Gamma = 0.63$



$Q = 0.82, B_r = 6.7\%, B_c = 0.97\%$

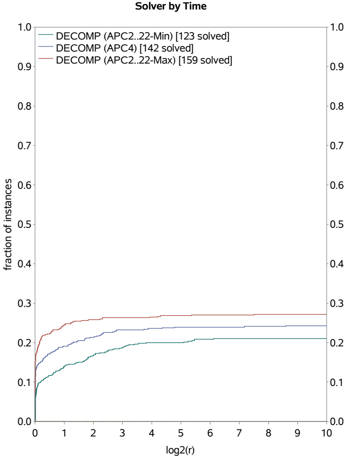
$\Gamma = 0.72$

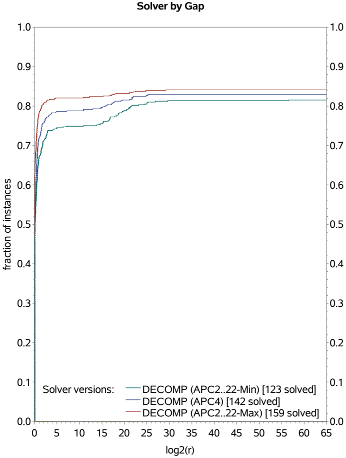


$Q = 0.94, B_r = 7.9\%, B_c = 0.93\%$

GOODNESS FUNCTION – RESULTS

Does choosing a decomposition based on (higher) Γ improve B&P performance? Yes.





585 Instances

APC2..22 vs APC4

- +17 instances solved (+25/-8)
- Shifted Geo-Mean Speedup (>1s)
  - 1.13 (122 instances)

KEE IN SAS/DECOMP

- Run APC for $\hat{K} = 2..22$ (threaded) and choose max goodness (Γ^{APC})
- Define the initial border rows ($\mathcal{S}$) from APC directly (SBDF) or greedily (DBDF)
- Iterative two-phase routine — record $\mathcal{S}$ with max goodness (Γ^{KEE})
    - **REMOVE:** At each sub-iteration
        - Move vertices (rows) into $\mathcal{S}$ based on community structure (maximizing modularity)
    - **MERGE:** At each sub-iteration
        - Move vertices (rows) out of $\mathcal{S}$ that maximize $\Delta\Gamma$ (merging blocks / components)
- Return $\mathcal{S}$ corresponding to $\max(\Gamma^{\mathrm{KEE}}, \Gamma^{\mathrm{APC}})$

AUTO DETECTION | KEE IN SAS/DECOMP — REMOVE

- Find communities (blocks) in $G = [V \backslash \mathcal{S}]$ which maximizes modularity using Louvain (heuristic)
- Move vertices (rows) into $\mathcal{S}$ with the highest number of inter-community edges
- Evaluate Γ with blocks defined as the connected components of $G = [V \backslash \mathcal{S}]$
- STOP when there are no inter-community edges

$\Gamma =$

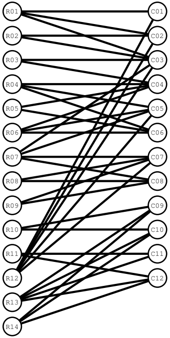
$K =$

$Q =$

$B_r =$

$M =$

	C01	C02	C03	C04	C05	C06	C07	C08	C09	C10	C11	C12
R01	●	●	●									
R02		●	●									
R03			●	●								
R04				●	●	●						
R05				●		●						
R06				●	●	●						
R07			●		●		●	●				
R08							●	●				
R09							●	●				
R10									●	●		
R11											●	●
R12	●	●	●		●		●					
R13									●	●	●	●
R14									●	●		●



AUTO DETECTION | KEE IN SAS/DECOMP — REMOVE

- Find communities (blocks) in $G = [V \setminus \mathcal{S}]$ which maximizes modularity using Louvain (heuristic)
- Move vertices (rows) into $\mathcal{S}$ with the highest number of inter-community edges
- Evaluate Γ with blocks defined as the connected components of $G = [V \setminus \mathcal{S}]$
- STOP when there are no inter-community edges

$\Gamma =$

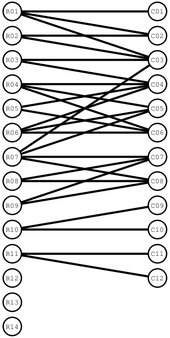
$K =$

$Q =$

$B_r =$

$M =$

		C01	C02	C03	C04	C05	C06	C07	C08	C09	C10	C11	C12
	R01	•	•	•									
	R02		•	•									
	R03			•	•								
	R04				•	•	•						
	R05				•		•						
	R06				•	•	•						
	R07			•		•		•	•				
	R08							•	•				
	R09							•	•				
	R10									•	•		
	R11											•	•
$\mathcal{S}$	R12	•	•	•		•		•					
$\mathcal{S}$	R13									•	•	•	•
$\mathcal{S}$	R14									•	•		•



AUTO DETECTION | KEE IN SAS/DECOMP — REMOVE

- Find communities (blocks) in $G = [V \backslash \mathcal{S}]$ which maximizes modularity using Louvain (heuristic)
- Move vertices (rows) into $\mathcal{S}$ with the highest number of inter-community edges
- Evaluate Γ with blocks defined as the connected components of $G = [V \backslash \mathcal{S}]$
- STOP when there are no inter-community edges

$\Gamma = 0.21$

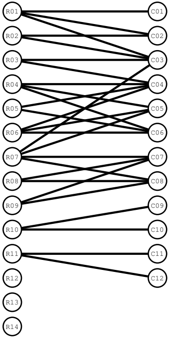
$K = 3$

$Q = 0.26$

$B_r = 21.4\%$

$M =$

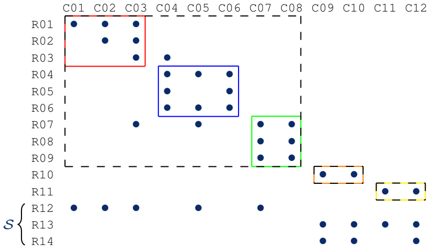
		C01	C02	C03	C04	C05	C06	C07	C08	C09	C10	C11	C12
	R01	•	•	•									
	R02		•	•									
	R03			•	•								
	R04				•	•	•						
	R05				•		•						
	R06				•	•	•						
	R07			•		•		•	•				
	R08							•	•				
	R09							•	•				
	R10									•	•		
	R11											•	•
$\mathcal{S}$	R12	•	•	•		•		•					
$\mathcal{S}$	R13									•	•	•	•
$\mathcal{S}$	R14									•	•		•

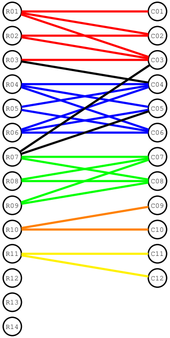


AUTO DETECTION | KEE IN SAS/DECOMP — REMOVE

- Find communities (blocks) in $G = [V \setminus \mathcal{S}]$ which maximizes modularity using Louvain (heuristic)
- Move vertices (rows) into $\mathcal{S}$ with the highest number of inter-community edges
- Evaluate Γ with blocks defined as the connected components of $G = [V \setminus \mathcal{S}]$
- STOP when there are no inter-community edges

$$\Gamma = 0.21$$
$$K = 3$$
$$Q = 0.26$$
$$B_r = 21.4\%$$
$$M = 0.63$$





AUTO DETECTION | KEE IN SAS/DECOMP — REMOVE

- Find communities (blocks) in $G = [V \setminus \mathcal{S}]$ which maximizes modularity using Louvain (heuristic)
- Move vertices (rows) into $\mathcal{S}$ with the highest number of inter-community edges
- Evaluate Γ with blocks defined as the connected components of $G = [V \setminus \mathcal{S}]$
- STOP when there are no inter-community edges

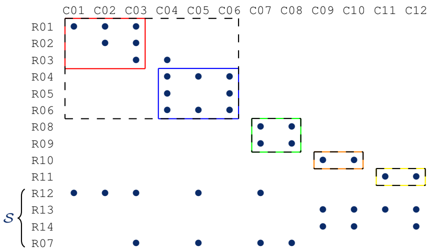
$\Gamma = 0.40$

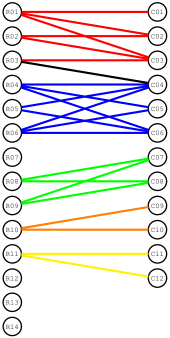
$K = 4$

$Q = 0.53$

$B_r = 28.6\%$

$M = 0.69$





AUTO DETECTION | KEE IN SAS/DECOMP — REMOVE

- Find communities (blocks) in $G = [V \backslash \mathcal{S}]$ which maximizes modularity using Louvain (heuristic)
- Move vertices (rows) into $\mathcal{S}$ with the highest number of inter-community edges
- Evaluate Γ with blocks defined as the connected components of $G = [V \backslash \mathcal{S}]$
- STOP when there are no inter-community edges

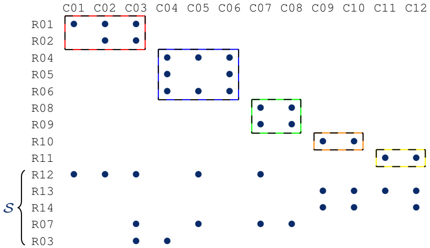
$\Gamma = 0.52$

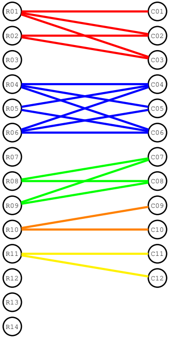
$K = 5$

$Q = 0.74$

$B_r = 35.7\%$

$M = 0.74$





KEE IN SAS/DECOMP — MERGE

- Move vertices (rows) out of $\mathcal{S}$ that maximize $\Delta\Gamma$ (merging blocks / components)
- STOP when the number of blocks is 1

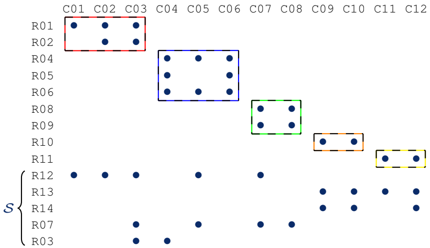
$\Gamma = 0.52$

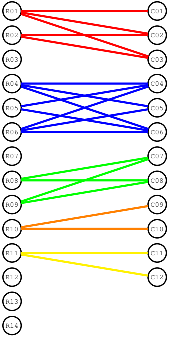
$K = 5$

$Q = 0.74$

$B_r = 35.7\%$

$M = 0.74$

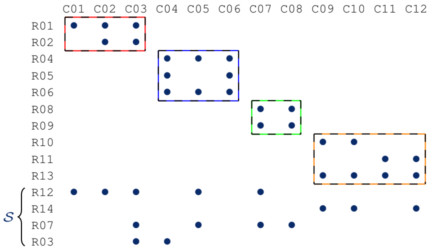


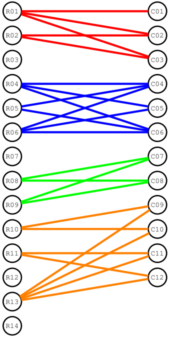


KEE IN SAS/DECOMP — MERGE

- Move vertices (rows) out of $\mathcal{S}$ that maximize $\Delta\Gamma$ (merging blocks / components)
- STOP when the number of blocks is 1

$$\Gamma = 0.55$$
$$K = 4$$
$$Q = 0.73$$
$$B_r = 28.6\%$$
$$M = 0.73$$





AUTO DETECTION | KEE IN SAS/DECOMP — MERGE

- Move vertices (rows) out of $\mathcal{S}$ that maximize $\Delta\Gamma$ (merging blocks / components)
- STOP when the number of blocks is 1

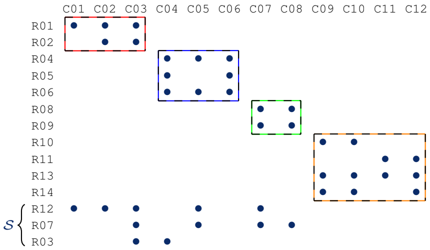
$\Gamma = 0.57$

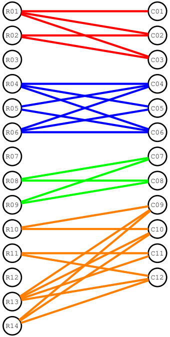
$K = 4$

$Q = 0.71$

$B_r = 21.4\%$

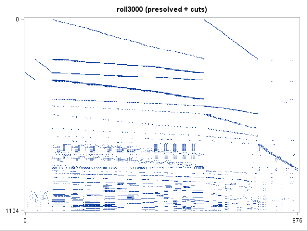
$M = 0.71$



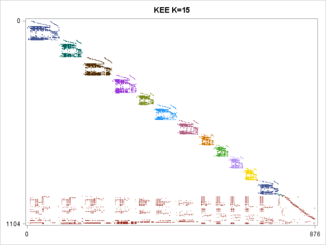


TUNING α

- Fix $\alpha_c = 20$, and vary α_r
  - $\alpha_r \uparrow$ emphasizes minimization of the (row) border
  - $\alpha_r \downarrow$ emphasizes quality of the diagonal

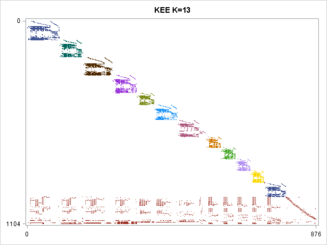


$\alpha_r = 0$ (Solve time 186 secs)



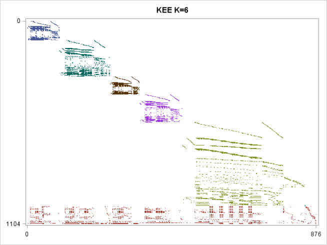
$Q = 0.907$, $B_r = 13.6\%$

$\alpha_r = 1$ (Solve time 83 secs)



$Q = 0.906$, $B_r = 13.4\%$

$\alpha_r = 5$ (Solve time 1401 secs)



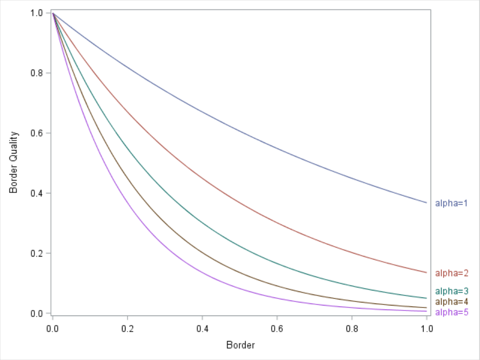
$Q = 0.76$, $B_r = 8.7\%$

TUNING α

Comparison to $\alpha_r = 1$

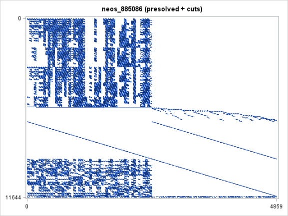
α_r	Solved		ShGeo-Mean
0.0	168	−6 (+17/−23)	0.89 (140)
0.5	175	+1 (+13/−12)	0.94 (150)
1.0	174		1.00
1.5	177	+3 (+11/−8)	0.94 (154)
2.0	178	+4 (+15/−11)	0.91 (151)
3.0	176	+2 (+16/−14)	0.90 (148)
4.0	176	+2 (+17/−15)	0.84 (147)
5.0	169	−5 (+15/−20)	0.82 (142)
Best	213	+39	1.26 (162)

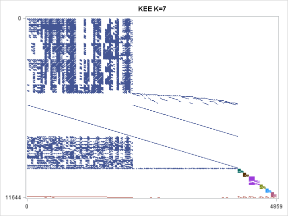
`Best` — could be a share-nothing (gridded) nondeterministic concurrent



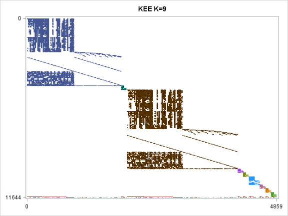
AUTO DETECTION | Γ VS B&P PERFORMANCE

Does choosing a decomposition based on maximizing Γ improve B&P performance? Yes.

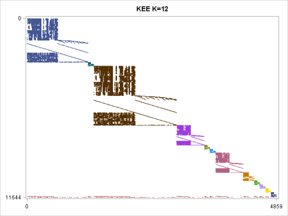




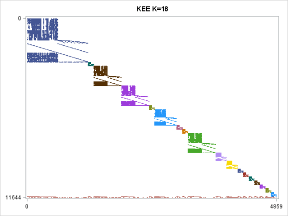
$\Gamma = 0.28$ (Timeout 1.23% Gap)



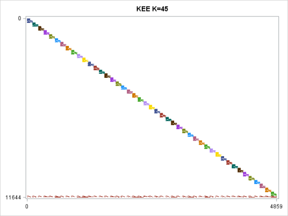
$\Gamma = 0.65$ (Timeout 1.23% Gap)



$\Gamma = 0.79$ (Solve time 303 secs)



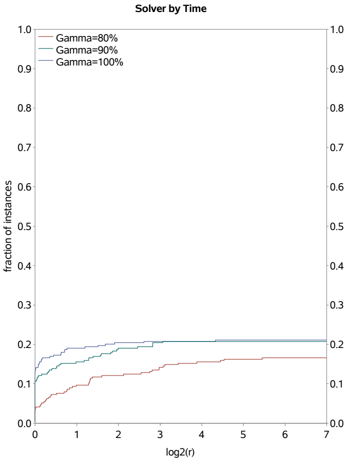
$\Gamma = 0.88$ (Solve time 63 secs)

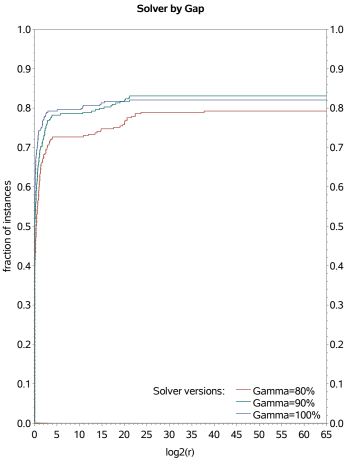


$\Gamma = 0.97$ (Solve time 30 secs)

AUTO DETECTION | Γ VS B&P PERFORMANCE

Does choosing a decomposition based on maximizing Γ improve B&P performance? Yes.





289 Instances

$\Gamma_{\max}$ vs $0.9\Gamma_{\max}$

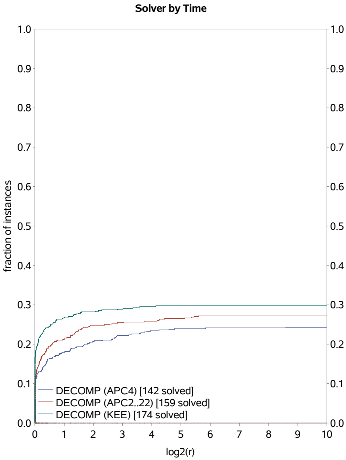
- +1 instances solved (+10/-9)
- Shifted Geo-Mean Speedup
  - 1.20 (50 instances)

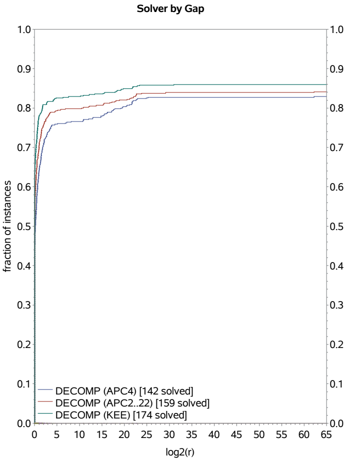
$\Gamma_{\max}$ vs $0.8\Gamma_{\max}$

- +13 instances solved (+19/-6)
- Shifted Geo-Mean Speedup
  - 1.77 (41 instances)

KEE — RESULTS

Does choosing a decomposition based on maximizing Γ improve B&P performance? Yes.





585 Instances

KEE vs APC2..22

- +15 instances solved (+29/-14)
- Shifted Geo-Mean Speedup
  - 1.17 (135 instances)

KEE vs APC4

- +32 instances solved (+49/-17)
- Shifted Geo-Mean Speedup
  - 1.40 (115 instances)

SOFTWARE — BRANCH & CUT VS BRANCH & PRICE

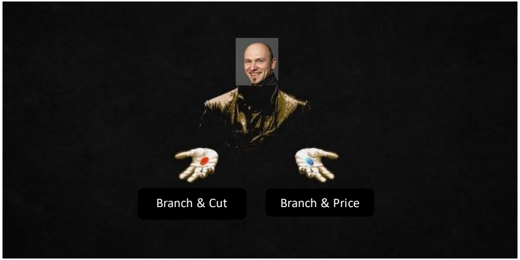
- Reality check (automated methods) — 85% of the time B&C *wins* (faster or better gap)
- How do we get the best of both (by default)?
  - Concurrent
  - Machine Learning, of course!... ask Marco — this research may provide a new feature (Γ)

	NumSolved	NumWins	Shifted Geo-Mean
Branch & Cut	345/585 (59%)	415/487 (85%)	3.3x (145)
Branch & Price	174/585 (30%)		

SOFTWARE — BRANCH & CUT VS BRANCH & PRICE

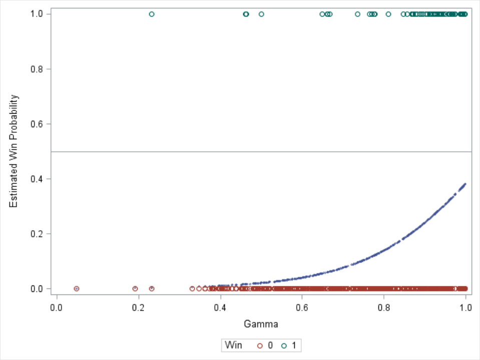
- Reality check (automated methods) — 85% of the time B&C *wins* (faster or better gap)
- How do we get the best of both (by default)?
  - Concurrent
  - Machine Learning, of course!… ask Marco — this research may provide a new feature (Γ)

	NumSolved	NumWins	Shifted Geo-Mean
Branch & Cut	345/585 (59%)	415/487 (85%)	3.3x (145)
Branch & Price	174/585 (30%)		



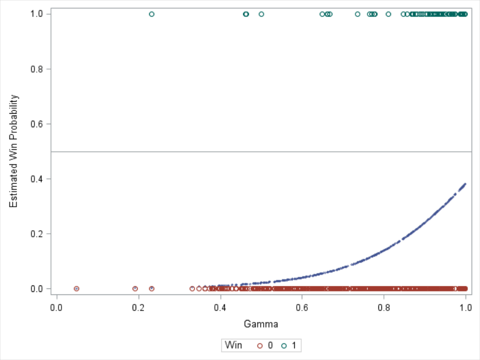
SOFTWARE — BRANCH & CUT AND BRANCH & PRICE

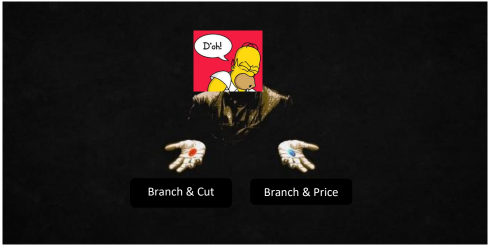
Does Γ provide a decision rule for when to use B&P over B&C?



SOFTWARE — BRANCH & CUT AND BRANCH & PRICE

Does Γ provide a decision rule for when to use B&P over B&C? No.



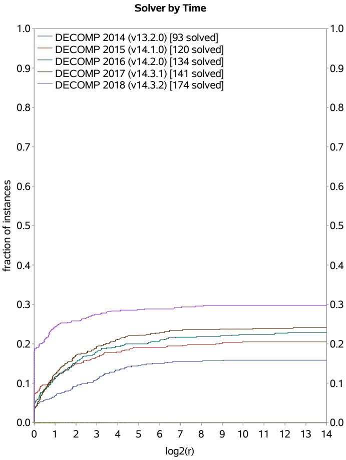


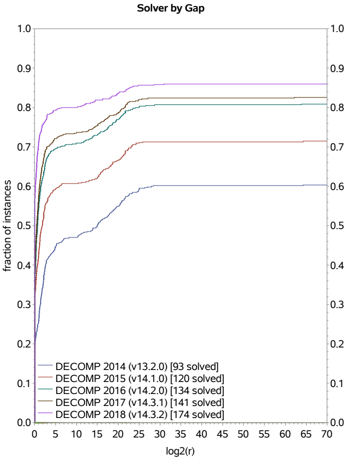
FUTURE DIRECTIONS

- Build ML model to choose among options (e.g., for various α) — features: $Q, B_r, B_c, \ldots$
  - Played *data scientist* for a week — Fail
  - What exactly is the target output? How do you score (or rank) solved versus not solved cases?
  - How do you handle cases that are *close*? Doesn't that depend on the user's value function?
- Improve the form of $\Gamma(\,)$ directly — regression $f(Q, B_r, B_c, \ldots)$
- Analyze the effect of cuts (and presolve) hiding structures — run KEE before and after
  - Especially important for isomorphic structures (which KEE indirectly encourages)
- DBDF
  - KEE on A_{DBDF} from APC — preliminary results are poor
  - Generalize the KEE algorithm directly
  - Compare APC to hMETIS (also requires a fixed number of blocks)
- Speed of KEE algorithm
  - Median = 0.3 secs, but 95% quantile = 175 secs
  - Need a better stopping criterion — do we know a tight bound on max Γ?

support.sas.com/or

SOFTWARE — BRANCH & PRICE (AUTO) PROGRESS





Default (Hybrid) — 4 Threads

- 585 Instances — bench minus:
  - Solved in presolver
  - Solved in B&C root
  - Auto-detect failed
- Max Time = 3600 seconds