DECOMP: A Framework for Decomposition in Integer Programming

Matthew V. Galati
Ted K. Ralphs

SAS Institute - Analytical Solutions - Operations Research and Development, Cary, NC
Lehigh University - Department of Industrial and Systems Engineering, Bethlehem, PA

COR@L - Computational Optimization Research at Lehigh
http://sagan.ie.lehigh.edu/mgalati

Outline

- Decomposition Methods
  - Cutting Plane Method
  - Dantzig-Wolfe Method
  - Lagrangian Method
- Integrated Decomposition Methods
  - Price and Cut
  - Relax and Cut
- Structured Separation and Motivation
- Decomp and Cut
- DECOMP Framework

Decomposition Methods

Preliminaries

- Consider the following integer linear program (ILP):

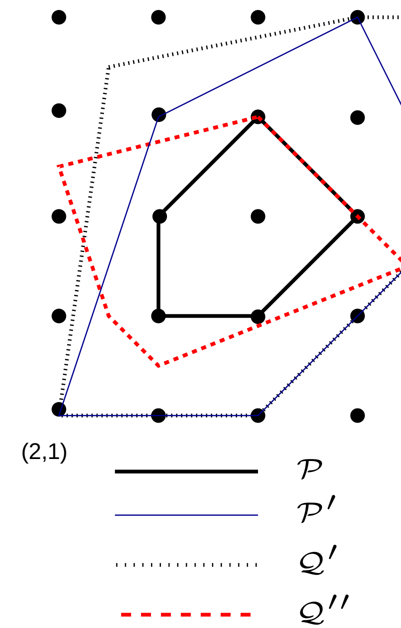
$$z_{IP} = \min_{x \in \mathcal{F}} \{c^\top x\} = \min_{x \in \mathcal{P}} \{c^\top x\} = \min_{x \in \mathbb{Z}^n} \{c^\top x \mid Ax \geq b\}$$

where

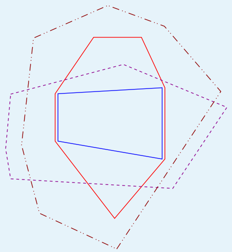
$$\mathcal{F} = \{x \in \mathbb{Z}^n \mid A'x \geq b', A''x \geq b''\} \quad \mathcal{Q} = \{x \in \mathbb{R}^n \mid A'x \geq b', A''x \geq b''\}$$

$$\mathcal{F}' = \{x \in \mathbb{Z}^n \mid A'x \geq b'\} \quad \mathcal{Q}' = \{x \in \mathbb{R}^n \mid A'x \geq b'\}$$

$$\mathcal{Q}'' = \{x \in \mathbb{R}^n \mid A''x \geq b''\}$$



- Denote $\mathcal{P} = conv(\mathcal{F})$ and $\mathcal{P}' = conv(\mathcal{F}')$.
- $OPT(c, X)$: Subroutine returns $x \in X$ that minimizes $c^\top x$.
- $SEP(x, X)$: Subroutine returns (a, β) which separates x from X (if exists).



Preliminaries

- **Assumption:**
  - $OPT(c, \mathcal{P})$ and $SEP(x, \mathcal{P})$ are *"hard"*.
  - $OPT(c, \mathcal{P}')$ and $SEP(x, \mathcal{P}')$ are *"easy"*.
  - $\mathcal{Q}''$ can be represented explicitly (description has polynomial size).
  - $\mathcal{P}'$ must be represented implicitly (description has exponential size).

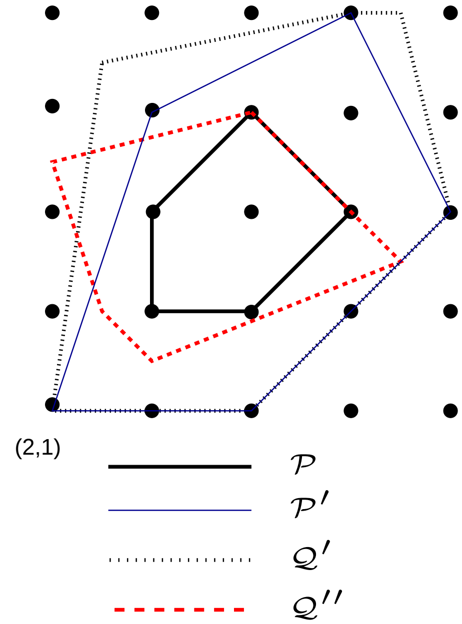
- Classical Example - Traveling Salesman Problem

$$
\begin{array}{rcll}
x(\delta(\{u\})) & = & 2 & \forall u \in V \\
x(E(S)) & \leq & |S| - 1 & \forall S \subset V,\ 3 \leq |S| \leq |V| - 1 \\
x_e \in \{0,1\} & & & \forall e \in E
\end{array}
$$

- One classical decomposition of TSP is to look for a spanning subgraph with $|V|$ edges ($\mathcal{P}'$ = 1-Tree) that satisfies the 2-degree constraints ($\mathcal{Q}''$).

Example - Polyhedra

$$
\begin{array}{lrcll}
\min & x_1 & & & \\
 & 7x_1 - x_2 & \geq & 13 & (1) \\
 & x_2 & \geq & 1 & (2) \\
 & -x_1 + x_2 & \geq & -3 & (3) \\
 & -4x_1 - x_2 & \geq & -27 & (4) \\
 & -x_2 & \geq & -5 & (5) \\
 & 0.2x_1 - x_2 & \geq & -4 & (6) \\
 & -x_1 - x_2 & \geq & -8 & (7) \\
 & -0.4x_1 + x_2 & \geq & 0.3 & (8) \\
 & x_1 + x_2 & \geq & 4.5 & (9) \\
 & 3x_1 + x_2 & \geq & 9.5 & (10) \\
 & 0.25x_1 - x_2 & \geq & -3 & (11) \\
 & & & x \in \mathbb{Z}^2 & (12)
\end{array}
$$



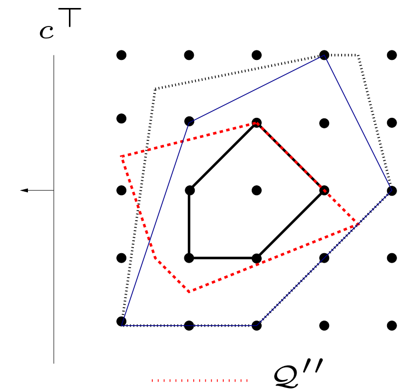
$\mathcal{Q}' = \{x \in \mathbb{R}^n \mid x$ satisfies $(1) - (6)\}$

$\mathcal{Q}'' = \{x \in \mathbb{R}^n \mid x$ satisfies $(7) - (11)\}$

$\mathcal{P}' = conv(\mathcal{Q}' \cap \mathbb{Z}^n)$

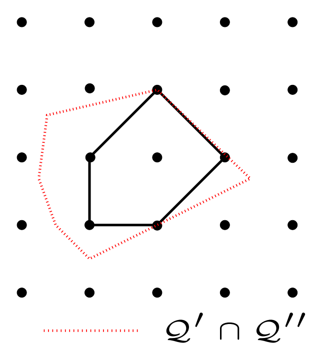
Bounding

- **Goal**: Compute a lower bound on z_{IP} by building an approximation to $\mathcal{P}$.
- The most straightforward approach is to use the continuous approximation



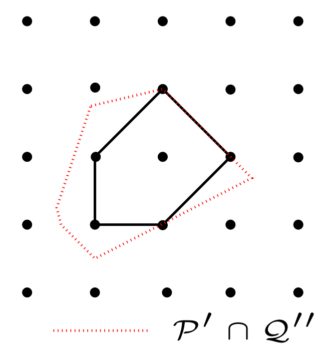
$$z_{LP} = \min_{x \in \mathcal{Q}} \{c^\top x\} = \min_{x \in \mathbb{R}^n} \{c^\top x \mid A'x \geq b', A''x \geq b''\}$$

- Decomposition approaches attempt to improve on this bound by utilizing the fact that $OPT(c, \mathcal{P}')$ or $SEP(x, \mathcal{P}')$ is *easy*.



$$z_D = \min_{x \in \mathcal{P}'} \{c^\top x \mid A''x \geq b''\} = \min_{x \in \mathcal{P}' \cap \mathcal{Q}''} \{c^\top x\} \geq z_{LP}$$

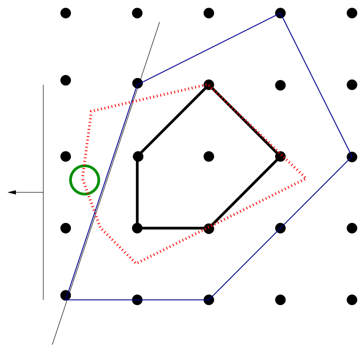
- $\mathcal{P}'$ is represented *implicitly* through solution of a **subproblem**.
- Decomposition Methods
  - Cutting Plane Method (Outer Method)
  - Dantzig-Wolfe Method / Lagrangian Method (Inner Methods)



Example: $z_{LP} = 2.25 < z_D = 2.42 < z_{IP} = 3.0$

Cutting Plane Method

- Cutting Plane Method (CPM) gives an approximation of $\mathcal{P}$ by building an *outer* approximation of $\mathcal{P}'$ intersected with $\mathcal{Q}''$.
- Let $[D, d]$ denote the facets of $\mathcal{P}'$, so that $\mathcal{P}' = \{x \in \mathbb{R}^n \mid Dx \geq d\}$.



1. **Initialize**: Form outer approximation $[D^0, d^0] = [A'', b'']$

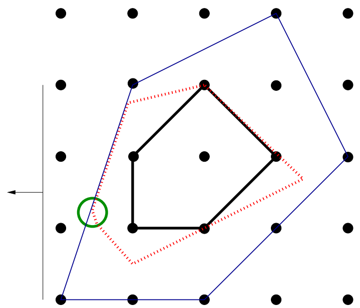
$$\mathcal{P}_O^0 = \{x \in \mathbb{R}^n \mid D^0 x \geq d^0\} \supseteq \mathcal{P}'$$

2. **Master Problem**: Obtain optimal *primal* solution x_{CP}^t

$$z_{CP}^t = \min_{x \in \mathbb{R}^n} \{c^\top x \mid D^t x \geq d^t\} = \min_{x \in \mathcal{P}_O^t} \{c^\top x\}$$

3. **Subproblem**: Call $SEP(x_{CP}^t, \mathcal{P}')$ to generate v.i.s for $\mathcal{P}$, violated by x_{CP}^t.
4. **Update**: If found, form a new outer approximation, and go to Step 2

$$\mathcal{P}_O^{t+1} = \{x \in \mathbb{R}^n \mid D^{t+1} x \leq d^{t+1}\} \supseteq \mathcal{P}'$$

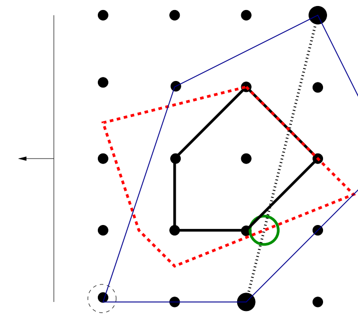


- The method converges to the bound $z_{CP} = c^\top \hat{x}_{CP} = z_D$.

Dantzig-Wolfe Method

- Dantzig-Wolfe Method (DW) gives an approximation of $\mathcal{P}$ by building an *inner* approximation of $\mathcal{P}'$ intersected with $\mathcal{Q}''$.
- Let $\mathcal{E}$ denote the extreme points of $\mathcal{P}'$, so that

$$\mathcal{P}' = \{x \in \mathbb{R}^n \mid x = \sum_{s \in \mathcal{E}} s\lambda_s, \sum_{s \in \mathcal{E}} \lambda_s = 1, \lambda_s \geq 0 \; \forall s \in \mathcal{E}\}.$$

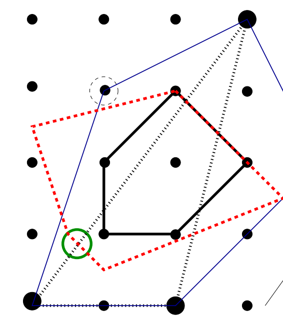


1. **Initialize**: Form inner approximation $\mathcal{E}^0 \subset \mathcal{E}$

$$\mathcal{P}_I^0 = \{x \in \mathbb{R}^n \mid x = \sum_{s \in \mathcal{E}^0} s\lambda_s, \sum_{s \in \mathcal{E}^0} \lambda_s = 1, \lambda_s \geq 0 \; \forall s \in \mathcal{E}^0\} \subseteq \mathcal{P}'$$

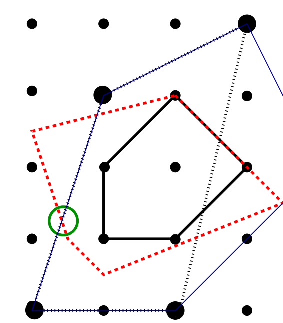
2. **Master Problem**: Obtain optimal *dual* solution $(u^t_{DW}, \alpha^t_{DW})$

$$\bar{z}^t_{DW} = \min_{\lambda \in \mathbb{R}_+^{\mathcal{E}^t}} \{c^\top (\sum_{s \in \mathcal{E}^t} s\lambda_s) \mid A''(\sum_{s \in \mathcal{E}^t} s\lambda_s) \geq b'', \sum_{s \in \mathcal{E}^t} \lambda_s = 1\}$$



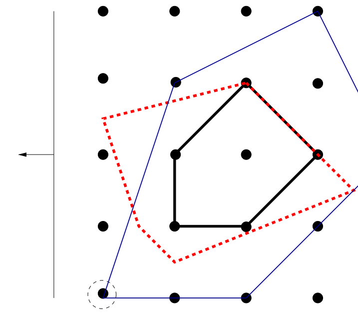
3. **Subproblem**: Call $OPT(c^\top - (u^t_{DW})^\top A'', \mathcal{P}')$, to generate e.p.s with $rc(s) < 0$.
4. **Update**: If found, form a new inner approximation, and go to Step 2

$$\mathcal{P}_I^{t+1} = \{x \in \mathbb{R}^n \mid x = \sum_{s \in \mathcal{E}^{t+1}} s\lambda_s, \sum_{s \in \mathcal{E}^{t+1}} \lambda_s = 1, \lambda_s \geq 0 \; \forall s \in \mathcal{E}^{t+1}\} \subseteq \mathcal{P}'$$



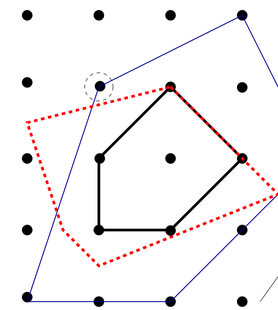
- The method converges to the bound $z_{DW} = c^\top(\sum_{s \in \mathcal{E}} s\hat{\lambda}_s) = c^\top \hat{x}_{DW} = z_D$

Lagrangian Method



- Lagrangian Method (LD) formulates a relaxation as finding the minimal extreme point of $\mathcal{P}'$ with respect to a cost which is penalized if the point lies outside of $\mathcal{Q}''$.
- The Lagrangian Dual is a piecewise-linear concave function

$$z_{LD} = \max_{u \in \mathbb{R}^{m''}_{+}} \{\min_{s \in \mathcal{E}} \{c^\top s + u^\top (b'' - A''s)\}\}$$

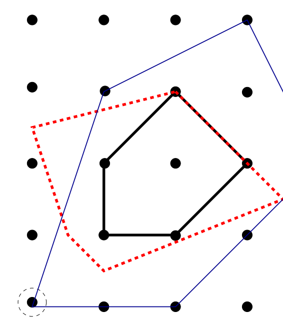


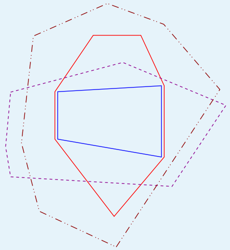
- Rewriting LD as an LP gives the dual of the DW-LP

$$z_{LD} = \max_{\alpha \in \mathbb{R}, u \in \mathbb{R}^{m''}_{+}} \{\alpha + b''^\top u \mid \alpha \leq (c^\top - u^\top A'')s \; \forall s \in \mathcal{E}\}$$

- So, $z_{LD} = z_{DW}$ and Lagrangian Method also achieves the bound z_D.

1. **Initialize**: Define $s^0 \in \mathcal{E}$, initialize dual multipliers u^0_{LD} for $[A'', b'']$.
2. **Master Problem**: Update the dual multipliers using directional information from s^t.
3. **Subproblem**: Call $OPT(c - (u^t_{LD})^\top A'', \mathcal{P}')$, to obtain a new direction $s^{t+1} \in \mathcal{E}$. If the stopping criterion is not met, go to Step 2.





Common Framework

- The continuous approximation of $\mathcal{P}$ is formed as the intersection of two explicitly defined polyhedra (*both with a small description*).

$$z_{LP} = \min_{x \in \mathbb{R}^n} \{c^\top x \mid x \in \mathcal{Q}' \cap \mathcal{Q}''\}$$

- Decomposition Methods form an approximation as the intersection of one explicitly defined polyhedron (*with a small description*) and one implicitly defined polyhedron (*with a large description*).

$$z_D = \min_{x \in \mathbb{R}^n} \{c^\top x \mid x \in \mathcal{P}' \cap \mathcal{Q}''\} \geq z_{LP}$$

- Each of the traditional decomposition methods contain two primary steps
    - **Master Problem:** Update the primal or dual **solution** information.
    - **Subproblem:** Update the **approximation** of $\mathcal{P}$: $SEP(x, \mathcal{P}')$ or $OPT(c, \mathcal{P}')$.
- Integrated Decomposition Methods form an approximation as the intersection of two implicitly defined polyhedra (*both with a large description*).
- So, we improve on the bound z_D by building *both* an inner approximation $\mathcal{P}_I$ of $\mathcal{P}'$ intersected with some outer approximation $\mathcal{P}_O \subset \mathcal{Q}''$.

Integrated Decomposition Methods

Price and Cut

- Price and Cut (PC) gives an approximation of $\mathcal{P}$ by building an *inner* approximation of $\mathcal{P}'$ (as in DW) intersected with an *outer* approximation of $\mathcal{P}$.

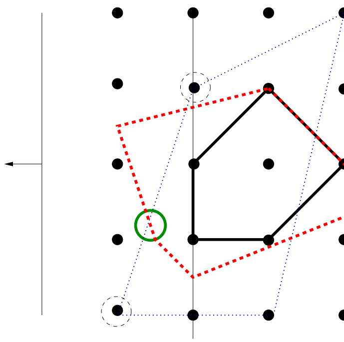
1. **Initialize**: Form inner $\mathcal{E}^0 \subset \mathcal{E}$ and outer $[D^0, d^0] = [A'', b'']$ approximations

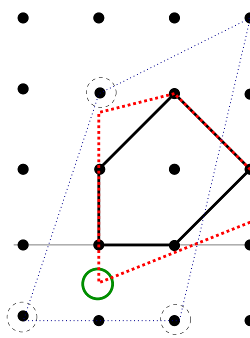
$$\begin{aligned} \mathcal{P}_I^0 &= \{x \in \mathbb{R}^n \mid x = \textstyle\sum_{s \in \mathcal{E}^0} s\lambda_s, \sum_{s \in \mathcal{E}^0} \lambda_s = 1, \lambda_s \geq 0 \; \forall s \in \mathcal{E}^0\} \subseteq \mathcal{P}' \\ \mathcal{P}_O^0 &= \{x \in \mathbb{R}^n \mid D^0 x \geq d^0\} \supseteq \mathcal{P} \end{aligned}$$

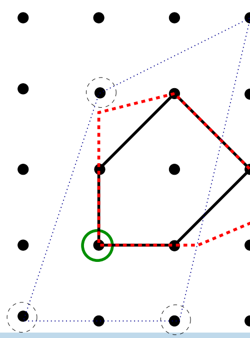
2. **Master Problem**: Solve the DW-LP to obtain optimal *dual* $(u^t_{PC}, \alpha^t_{PC})$, decomposition λ^t_{PC}, and *primal* solution x^t_{PC}.

$$\bar{z}^t_{PC} = \min_{\lambda \in \mathbb{R}_+^{\mathcal{E}^t}} \{c^\top (\sum_{s \in \mathcal{E}^t} s\lambda_s) \mid D^t (\sum_{s \in \mathcal{E}^t} s\lambda_s) \geq d^t, \sum_{s \in \mathcal{E}^t} \lambda_s = 1\}$$

3. Do either (a) or (b).
   (a) **Pricing Subproblem and Update**: Call $OPT(c^\top - (u^t_{PC})^\top D^t, \mathcal{P}')$, to generate e.p.s with $rc(s) < 0$. If found, form a new inner approximation $\mathcal{P}_I^{t+1}$, and go to Step 2.
   (b) **Cutting Subproblem and Update**: Call $SEP(x^t_{PC}, \mathcal{P})$ to generate v.i.s violated by x^t_{PC}. If found, form a new outer approximation $\mathcal{P}_O^{t+1}$, and go to Step 2.



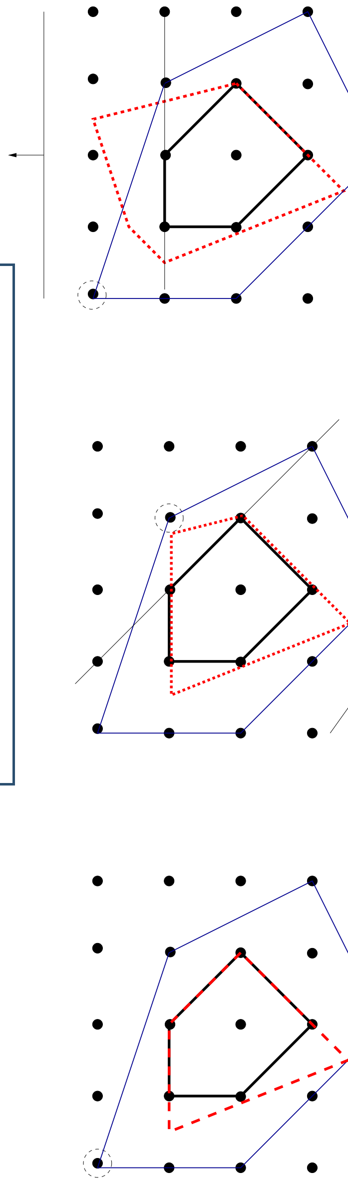




Relax and Cut

- Relax and Cut (RC) improves on the bound z_D using LD and augmenting the multiplier space with valid inequalities that are violated by the solution to the Lagrangian subproblem.

1. **Initialize**: Define $s^0 \in \mathcal{E}$, $[D^0, d^0] = [A'', b'']$, initialize dual multipliers u^0_{LD} for $[D^0, d^0]$.
2. **Master Problem**: Update the dual multipliers using directional information from s^t.
3. Do either (a) or (b).
   (a) **Pricing Subproblem**: Call $OPT(c - (u^t_{LD})^\top D^t, \mathcal{P}')$, to obtain a new direction $s^{t+1} \in \mathcal{E}$. If the stopping criterion is not met, go to Step 2.
   (b) **Cutting Subproblem**: Call $SEP(s^t, \mathcal{P})$ to generate v.i.s violated by s^t. If found, add them to $[D^t, d^t]$ along with new dual multipliers, and go to Step 2.



Structured Separation

- In general, the complexity of $OPT(c, X) = SEP(x, X)$.
- **Observation:** Restrictions on the input or output of these subroutines can change their complexity.
- Template Paradigm, restricts the *output* of $SEP(x, X)$ to valid inequalities (a, β) that conform to a certain structure. This class of inequalities forms a polyhedron $C \supset X$.
- For example, let $\mathcal{P}$ be the convex hull of solutions to the TSP.
  - $SEP(x, \mathcal{P})$ is NP-Complete.
  - $SEP(x, \mathcal{C})$ is polynomially solvable, for $\mathcal{C} \supset \mathcal{P}$ the Subtour Polytope (Min-Cut) or Blossom Polytope (Padberg-Rao).
- Structured Separation, restricts the *input* of $SEP(x, X)$, such that x conforms to some structure. For example, if x is restricted to solutions to a combinatorial problem, then separation often becomes much easier.

Example - TSP

- Traveling Salesman Problem Formulation:

$$
\begin{array}{lcll}
x(\delta(\{u\})) & = & 2 & \forall u \in V \\
x(E(S)) & \leq & |S| - 1 & \forall S \subset V,\ 3 \leq |S| \leq |V| - 1 \\
x_e \in \{0,1\} & & & \forall e \in E
\end{array}
$$

- $\mathcal{P}'$ = 1-Tree Relaxation: $OPT(c, 1-Tree)$ in $O(|E| \log |V|)$

$$
\begin{array}{lcll}
x(\delta(\{0\})) & = & 2 & \\
x(E(V \setminus \{0\})) & = & |V| - 2 & \\
x(E(S)) & \leq & |S| - 1 & \forall S \subset V \setminus \{0\}, 3 \leq |S| \leq |V| - 1 \\
x_e \in \{0,1\} & & & \forall e \in E
\end{array}
$$

- $\mathcal{P}'$ = 2-Matching Relaxation: $OPT(c, 2-Match)$ in polynomial time

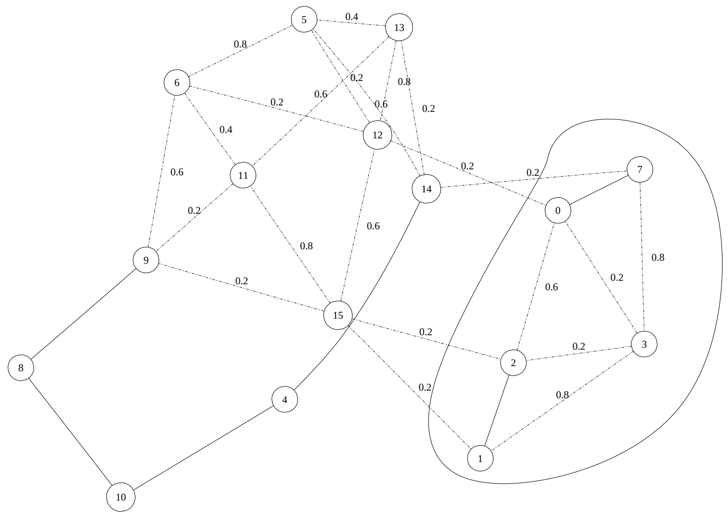
$$
\begin{array}{lcll}
x(\delta(u)) & = & 2 & \forall u \in V \\
x_e \in \{0,1\} & & & \forall e \in E
\end{array}
$$

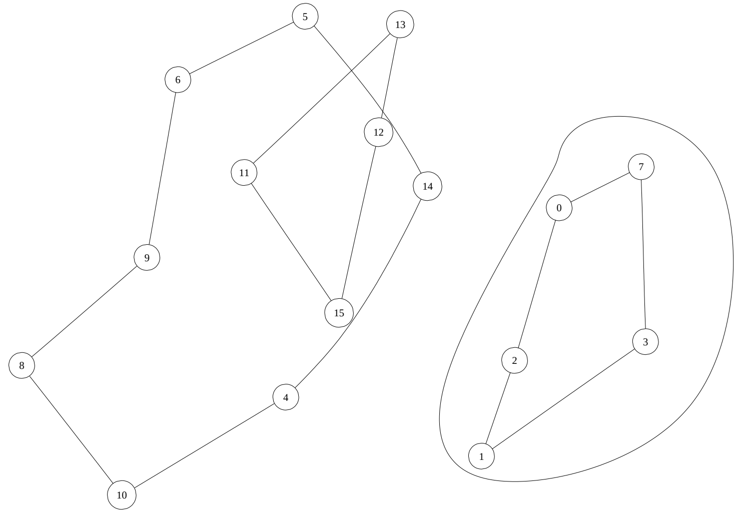
Example - TSP

- Separation of Subtour Inequalities:

$$x(E(S)) \leq |S| - 1$$

- $SEP(x, Subtour)$, for $x \in \mathbb{R}^n$ can be solved in $O(|V|^4)$ (Min-Cut)
- $SEP(s, Subtour)$, for s a 2-matching, can be solved in $O(|V|)$
  - Simply determine the connected components C_i, and set $S = C_i$ for each component (each gives a violation of 1).



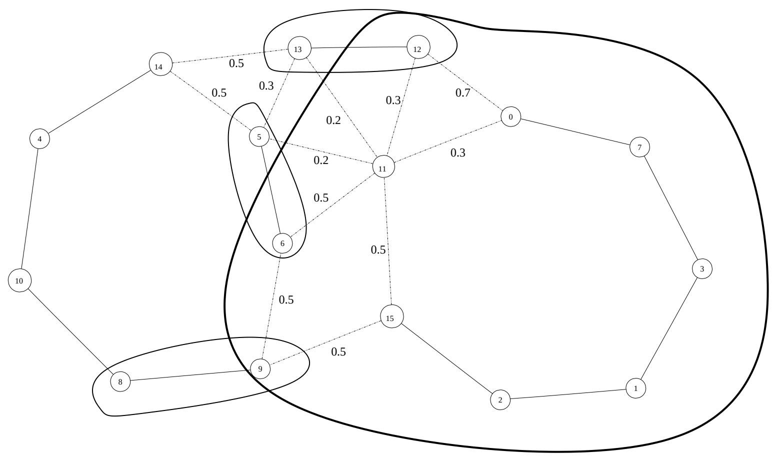


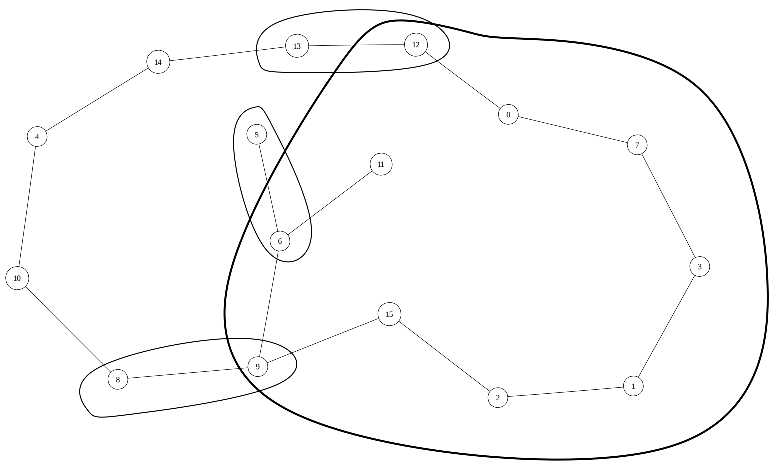
Example - TSP

- Separation of Comb Inequalities:

$$x(E(H)) + \sum_{i=1}^{k} x(E(T_i)) \leq |H| + \sum_{i=1}^{k} (|T_i| - 1) - \lceil k/2 \rceil$$

- $SEP(x, Blossoms)$, for $x \in \mathbb{R}^n$ can be solved in $O(|V|^5)$ (Padberg-Rao)
- $SEP(s, Blossoms)$, for s a 1-Tree, can be solved in $O(|V|^2)$
  - Consider paths in the cycle as candidate handles H and an odd ($>= 3$) set of edges with one endpoint in H and one in $V \setminus H$ as candidate teeth (each gives a violation of $\lceil k/2 \rceil - 1$).
  - This can also be used as a quick heuristic to separate 1-Trees for more general comb structures, for which there is no known polynomial algorithm for separation of arbitrary vectors.





Motivation

- In Relax and Cut, the solutions to the Lagrangian subproblem $s \in \mathcal{E}$ typically have some *nice* combinatorial structure. So, the cutting step in Relax and Cut $SEP(s, \mathcal{P})$, can be relatively easy as opposed to general separation.
- **Question:** Can we take advantage of this in other contexts?
- LP theory tells us that in order to improve the bound, it is *necessary and sufficient* to cut off the entire face of optimal solutions F.
- This condition is difficult to verify, so we typically use the *necessary condition* that the generated inequality be violated by some member of that face, $x \in F$.
  - In the Cutting Plane Method, we search for inequalities that violate $x^t_{CP} \in F^t$, where F^t is optimal face over $\mathcal{P}^t_O \cap \mathcal{Q}''$.
  - In the Price and Cut Method, we search for inequalities that violate $x^t_{PC} \in F^t$, where F^t is optimal face over $\mathcal{P}^t_I \cap \mathcal{P}^t_O$.

Motivation

- Now, consider the following set

$$\mathcal{S}(u, \alpha) = \{s \in \mathcal{E} \mid (c^\top - u^\top A'')s = \alpha\},$$

- Then, $\mathcal{S}(u^t_{DW}, \alpha^t_{DW})$ is the set e.p.s with $rc(s) = 0$ in the DW-LP master or the set of alternative optimal solutions to the Lagrangian subproblem.

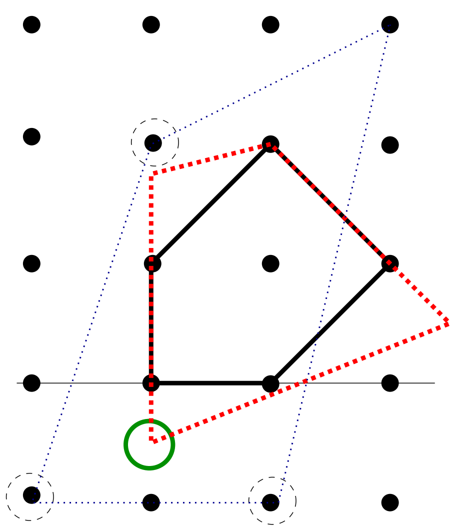
Theorem 1 $F^t \subseteq conv(\mathcal{S}(u^t_{DW}, \alpha^t_{DW}))$

- Therefore, separation of $\mathcal{S}(u^t_{DW}, \alpha^t_{DW})$ gives an alternative *necessary and sufficient* condition for an inequality to be improving.
- By convexity, it is clear that every improving inequality must violate at least one extreme point in the optimal decomposition.

Theorem 2 *If* $(a, \beta) \in \mathbb{R}^{(n+1)}$ *is an improving then there must exist an* $s \in \mathcal{D} = \{s \in \mathcal{E} \mid \lambda^t_s > 0\}$ *such that* $a^\top s < \beta$

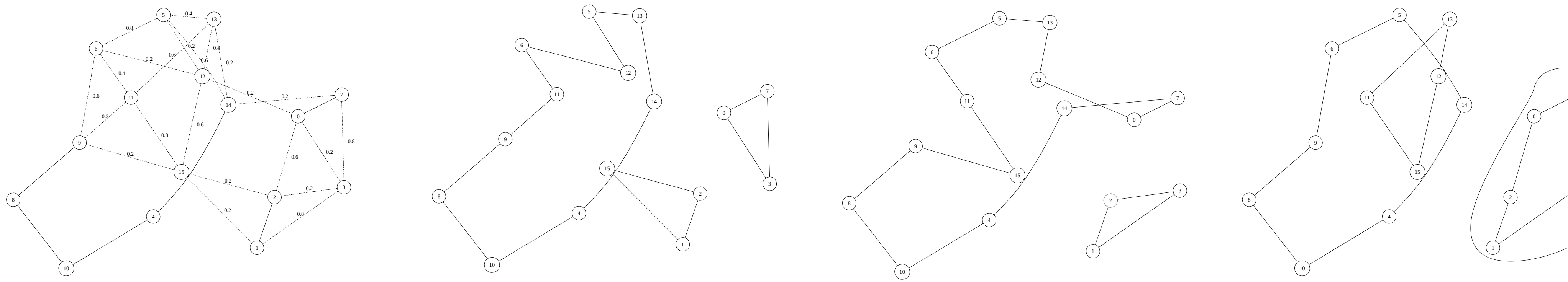
Theorem 3 $\mathcal{D} = \{s \in \mathcal{E} \mid \lambda^t_s > 0\} \subseteq \mathcal{S}(u^t_{PC}, \alpha^t_{PC})$

- Theorems 1-3, along with the observation that structured separation can be relatively easy, motivates the following revised PC method.



Price and Cut (Revisited)

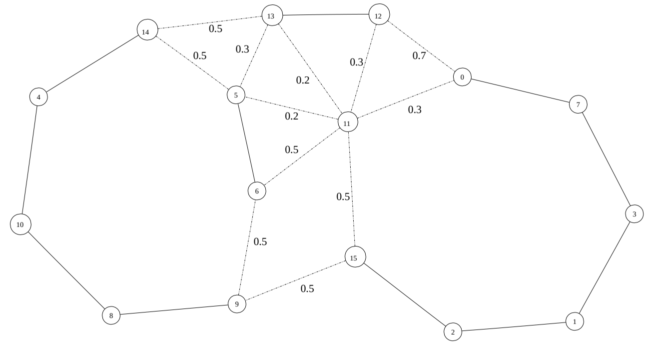
- Theorems 1-3 give us an alternative *necessary condition* for finding improving inequalities. PC gives us the optimal decomposition $D = \{s \in \mathcal{E} \mid \lambda_s > 0\}$.
- **Key Idea:** In the cutting subproblem, rather than (or in addition to) separating x^t_{PC}, separate each $s \in D$.
- The violated subtour found by separating the 2-Matching *also* violates the fractional point, but was found at little cost.



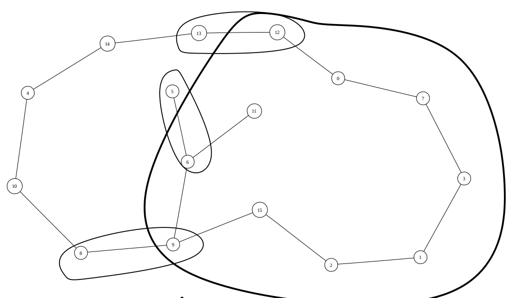
$\hat{x}$ $\quad$ $\hat{\lambda}_0 = 0.2$ $\quad$ $\hat{\lambda}_1 = 0.2$ $\quad$ $\hat{\lambda}_2 = 0.6$

Price and Cut (Revisited)

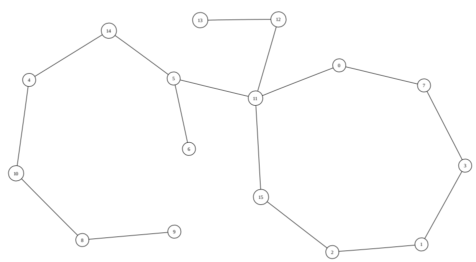
- Similarly, the violated blossom found by separating the 1-Tree *also* violates the fractional point, but was found at little cost.



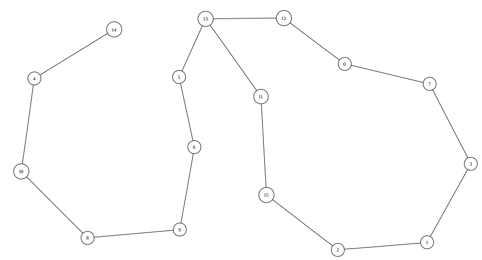
$\hat{x}$



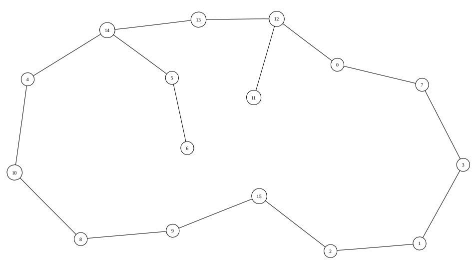
$\hat{\lambda}_0 = 0.3$



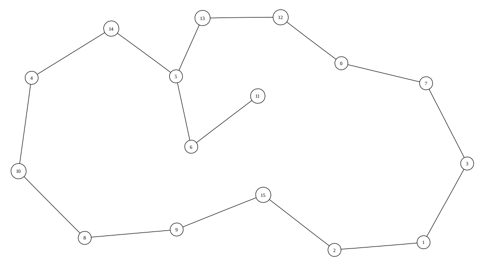
$\hat{\lambda}_1 = 0.2$



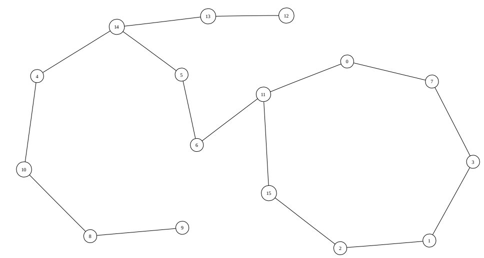
$\hat{\lambda}_2 = 0.2$



$\hat{\lambda}_3 = 0.1$



$\hat{\lambda}_4 = 0.1$



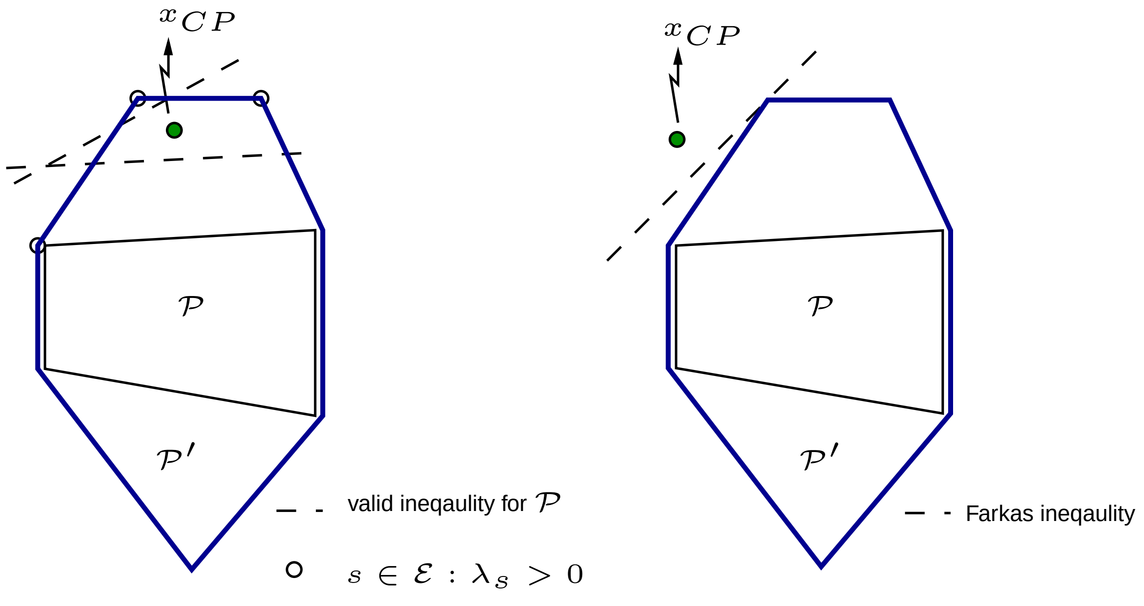
$\hat{\lambda}_5 = 0.1$

Decomp and Cut

- In the context of the traditional CPM, we can construct (*inverse DW*) the decomposition λ from the current fractional solution x_{CP} by solving the following LP

$$\max_{\lambda \in \mathbb{R}_+^{\mathcal{E}}} \{\mathbf{0}^\top \lambda \mid \sum_{s \in \mathcal{E}} s\lambda_s = x_{CP}, \ \sum_{s \in \mathcal{E}} \lambda_s = 1\},$$

- If we find a decomposition $\mathcal{D}$, then we separate each $s \in \mathcal{D}$, as in revised PC.
- If we fail, then the LP proof of infeasibility (**Farkas Cut**) gives us a separating hyperplane which can be used to cut off the current fractional point.



(a) $x_{CP} \in \mathcal{P}'$ (b) $x_{CP} \notin \mathcal{P}'$

DECOMP Framework

DECOMP Framework

- **DECOMP** provides a flexible software framework for testing and extending the theoretical framework presented thus far, with the primary goal of *minimal user responsibility*.
- DECOMP was built around data structures and interfaces provided by COIN-OR: **COmputational INfrastructure for Operations Research.**
- **BCP** provides a framework for parallel implementation of PC in a branch and bound framework with *LP-Based Bounding*.
- A generalization of BCP currently under development:
  - **ALPs**: Abstract Library for Parallel Search (ICS'05 - TA02)
  - **BiCePS**: Branch, Constrain and Price [*Generic Bounding*]
  - **BLIS**: BiCePS Linear Integer Solver = BCP
- DECOMP could provide an implementation of the **BiCePS** layer.

DECOMP - Applications

- The framework, written in C++, is accessed through two user interfaces:
  - Applications Interface: `DecompApp`
  - Algorithms Interface: `DecompAlgo`
- In order to develop an application, the user must derive the following methods/objects. All other methods have appropriate defaults but are `virtual` and may be overridden.
  - `DecompApp::createCore()`. Define $[A'', b'']$.
  - `DecompVar`. Define a variable $s \in \mathcal{F}'$ in terms of x-space.
  - `DecompCut`. Define a cut (a, β) in terms of x-space.
  - `DecompApp::solveRelaxedProblem()`. Provide a subroutine for $OPT(c, \mathcal{P}')$, given a cost vector c, that returns a set of solutions as `DecompVar` objects $\in \mathcal{F}'$.
  - `DecompApp::generateCuts(`s`)`. Provide a subroutine $SEP(s, \mathcal{P})$, given a `DecompVar` $\in \mathcal{F}'$, that returns a set of `DecompCut` objects.
- If the user wishes to do traditional CPM or PC, they must also provide
  - `DecompApp::generateCuts(`x`)`. Provide a subroutine $SEP(x, \mathcal{P})$, given a arbitrary real vector, that returns a set of `DecompCut` objects.

DECOMP Framework

- One important feature of DECOMP is that the user only needs to provide methods for their application in the original space (x-space), rather than in the space of a particular reformulation.
- This allows for users to consider cuts and variables in their most *intuitive* form and greatly simplifies the process of expansion into rows and columns.
- Features:
  - Automatic reformulation - row and column expansion in DW master, dualization and multiplier updates in RC, etc...
  - One interface to all default algorithms: **CPM/DC, DW, LD, PC, RC**.
  - Built on top of the COIN/OSI interface, so easily interchange LP solvers.
  - Active LP compression, variable and cut pool management.
  - Easily switch between relaxations (choice of $\mathcal{P}'$).

DECOMP - Algorithms

- The base class `DecompAlgo` provides the shell (master / subproblem) for integrated decomposition methods.
- Each of the methods described have derived default implementations `DecompAlgoX : public DecompAlgo`.
- New, hybrid or extended methods can be easily derived by overriding the various subroutines which are called from the base class. For example,
  - Alternative methods for solving the master LP in DW, such as **interior point methods** or ACCPM.
  - The user might choose to add a stabilizing factor to the dual updates in LD, as in **bundle methods**.
  - The user might choose the **Volume algorithm** for solving the LD, which provides an approximation primal solution for which cuts can be generated.

Applications Under Development

- Steiner Tree Problem
  - Minimum Spanning Tree : Lifted SECs, Partition - RC* [Lucena 92]
- Traveling Salesman Problem
  - One-Tree: Blossoms, Combs
  - Matching: SECs
- Vehicle Routing Problem
  - k-Traveling Salesman Problem : GSECs - DC [Ralphs, et al. 03]
  - k-Tree : GSECs, Combs, Multistars - RC* [Marthinhon, et al. 01]
- Axial Assignment Problem
  - Assignment Problem : Clique-Facets - RC [Balas, Saltzman 91]
- Knapsack Constrained Circuit Problem
  - Knapsack Problem : Cycle Cover, Maximal-Set Inequalities
  - Circuit Problem: Cycle Cover, Maximal-Set Inequalities
- Edge-Weighted Clique Problem
  - Tree Relaxation : Trees, Cliques - RC [Hunting, et al. 01]
- Subtour Elimination Problem [G. Benoit / S. Boyd]
  - Fractional 2-Factor Problem : SECs - DC / LP Context [Benoit, Boyd 03]

Summary

- Decomposition Methods approximate $\mathcal{P}$ as $\mathcal{P}' \cap \mathcal{Q}''$, where $\mathcal{P}'$ may have a *large* description.
- Integrated Decomposition Methods optimize over $\mathcal{P}_I \cap \mathcal{P}_O$, where $\mathcal{P}_I \subset \mathcal{P}'$ and $\mathcal{P}_O \supset \mathcal{P}$. Both polyhedra may have a *large* description.
- Structured separation can be much easier than general separation.
- We gave some motivation for two new techniques: **revised-PC** and **DC**.
  - The question remains: Empirically, how *good* are the cuts generated by separation of $s \in \mathcal{D}$?
  - However, for some facet classes, it doesn't matter - *we simply don't know how to separate* $x \in \mathbb{R}^n$. These ideas provide a starting point.
- **DECOMP** provides an easy-to-use framework for comparing and developing various decomposition-based methods.
- The code is open-source, currently released under CPL and will eventually be available through the COIN-OR project repository **www.coin-or.org**.
- Related publication at Optimization-Online: T. Ralphs, M. Galati, *Decomposition in Integer Linear Programming*