Decomposition-based Methods for Large-scale Discrete Optimization

Matthew V Galati

Ted K Ralphs

Department of Industrial and Systems Engineering

Lehigh University, Bethlehem, PA, USA

Départment de Mathématiques

École Polytechnique Fédérale de Lausanne, Suisse

May 14, 2002

Setup and Definitions

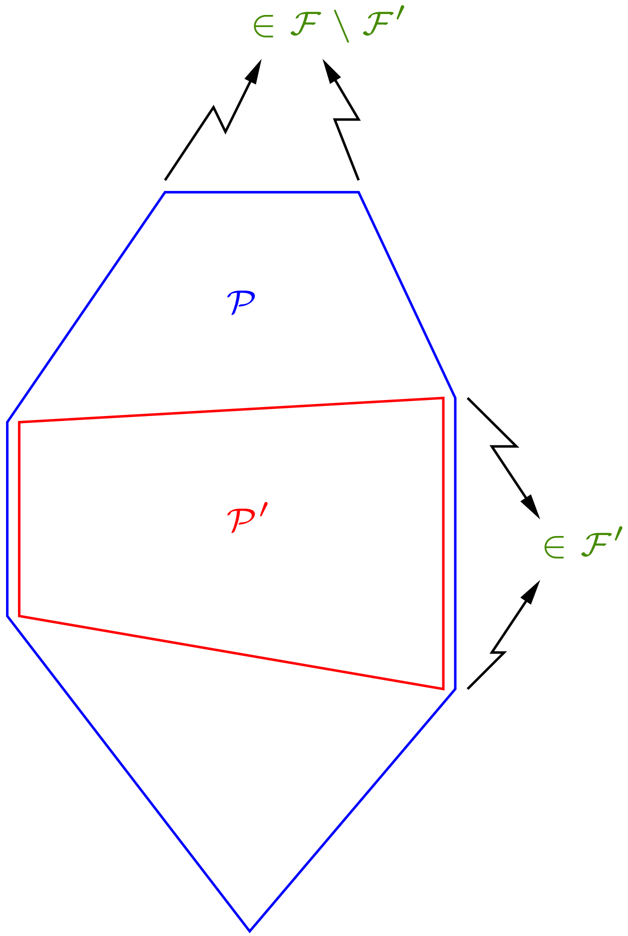
- For simplicity, we will only consider *combinatorial optimization problems* (COPs) defined by
  - Ground Set: E
  - Feasible Set: $\mathcal{F} \subseteq 2^E$
- An *instance* of a given COP is defined by a cost vector $c \in \mathbf{Z}^E$.
- For a given instance, the *cost* of $S \in \mathcal{F}$ is $c(S) = \sum_{e \in S} c_e$.
- We wish to find an element of $\mathcal{F}$ with minimum cost.
- Associated ILP for finding $\min_{S \in \mathcal{F}} c(S)$

$$
\begin{aligned}
\min \sum_{e \in E} c_e x_e & \\
s.t. \qquad\qquad Ax &\geq b \\
x_e &\in \{0,1\}, \forall e \in E
\end{aligned}
$$

- Reinterpret $\mathcal{F} := \{x \in \{0,1\}^E : Ax \geq b\}$

More Setup and Definitions

- Consider a COP $CP = (E, \mathcal{F})$ called the *base problem*.
- A *restriction* of CP is $CP' = (E, \mathcal{F}')$ where $\mathcal{F}' = \{x \in \mathcal{F} : Dx \geq d\}$.
- Suppose CP is "easy" and CP' is "hard".
- This means we can solve instances of CP effectively.
- Notation:
  - $\mathcal{P} = conv(\mathcal{F}) = \{x \in \mathbf{R}^n : A^I x \geq b^I\}$ (*base polytope*).
  - $\mathcal{P}^{'} = conv(\mathcal{F}^{'}) = \{x \in \mathcal{P} : D^I x \geq d^I\}$ (*restricted polytope*).
- Note that CP “easy” $\Rightarrow$ we can separate over $\mathcal{P}$.



The base polytope $\mathcal{P}$ and its restriction $\mathcal{P}'$

Examples

Problem:	Relaxation:
Traveling Salesman	Assignment 1-Tree
Vehicle Routing	Multiple Traveling Salesman k-Tree
Fixed-charge Network Flow	Minimum Cost Network Flow
Generalized Assignment	Assignment

Decomposition Methods

- The goal is to capitalize on our ability to solve instances of CP in order to solve instances of CP'.
- Use branch and bound with a decomposition-based lower bounding scheme.
- Three approaches to generating bounds:
  - Dantzig-Wolfe Decomposition
  - Lagrangian Relaxation
  - Cutting Plane Algorithm

Dantzig-Wolfe Decomposition

- Dantzig-Wolfe Decomposition

$$
\begin{aligned}
\min \quad & \sum_{f \in \mathcal{F}} c x^f \lambda^f \\
s.t. \quad & \sum_{f \in \mathcal{F}} D x^f \lambda^f \geq d \\
& \sum_{f \in \mathcal{F}} \lambda^f = 1 \\
& \lambda \in \mathbf{R}_+^{\mathcal{F}}
\end{aligned}
$$

- Solving the D-W LP yields a lower bound on $OPT(CP')$.
- We can do this using column generation.
- The column generation subproblem is an instance of CP (easy).
- We can recover the relaxed solution in terms of the original variables and branch by restricting column generation.

Lagrangian Relaxation

- Lagrangian Subproblem

$$LR(u) = \min_{x \in \mathcal{F}} \{(c - uD)x + ud\}$$

- Lagrangian Dual

$$\max_{u \geq 0} LR(u)$$

- Again, the LD provides a lower bound on $OPT(CP')$.
- We can solve this relaxation using subgradient optimization.
- The Lagrangian subproblem is an instance of CP.
- Branching is not straightforward in this case.

Lagrangian Relaxation using LP

- An alternative to subgradient optimization is to write the Lagrangian dual as a linear program.

$$\max_{u \geq 0} \min_{x \in \mathcal{F}} \{(c - uD)x + ud\} = \max_{u_0, u \geq 0} \{u_0 : u_0 \leq (c - uD)x^f + ud, \forall f \in \mathcal{F}\}$$

- This LP has a large number of constraints, but the separation problem is again an instance of CP.
- We can solve this LP effectively.
- In fact, this is just the dual of the D-W LP.
- We can again recover the primal solution and branching can be done by restricting cut generation.

Cutting Plane Algorithm

- Because of the equivalence of optimization and separation, separating over $\mathcal{P}$ is also "easy."
- Thus, a third approach to bounding is to solve the LP relaxation

$$\min\{cx : A^I x \geq b^I, Dx \geq d\}$$

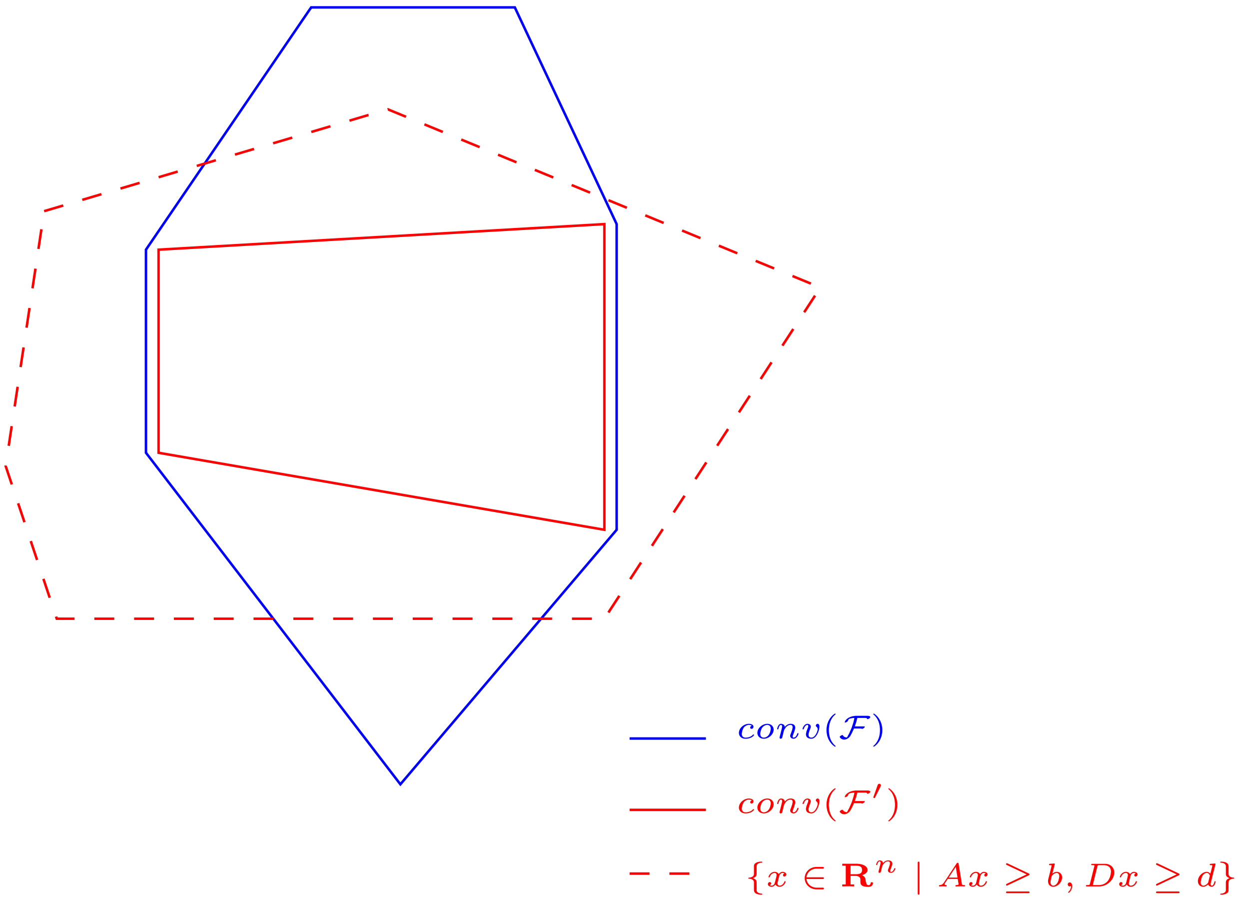
- This bounding scheme can also be embedded into branch and bound.
- In this case, we could branch on fractional variables or any other disjunction.

Comparing the Bounds

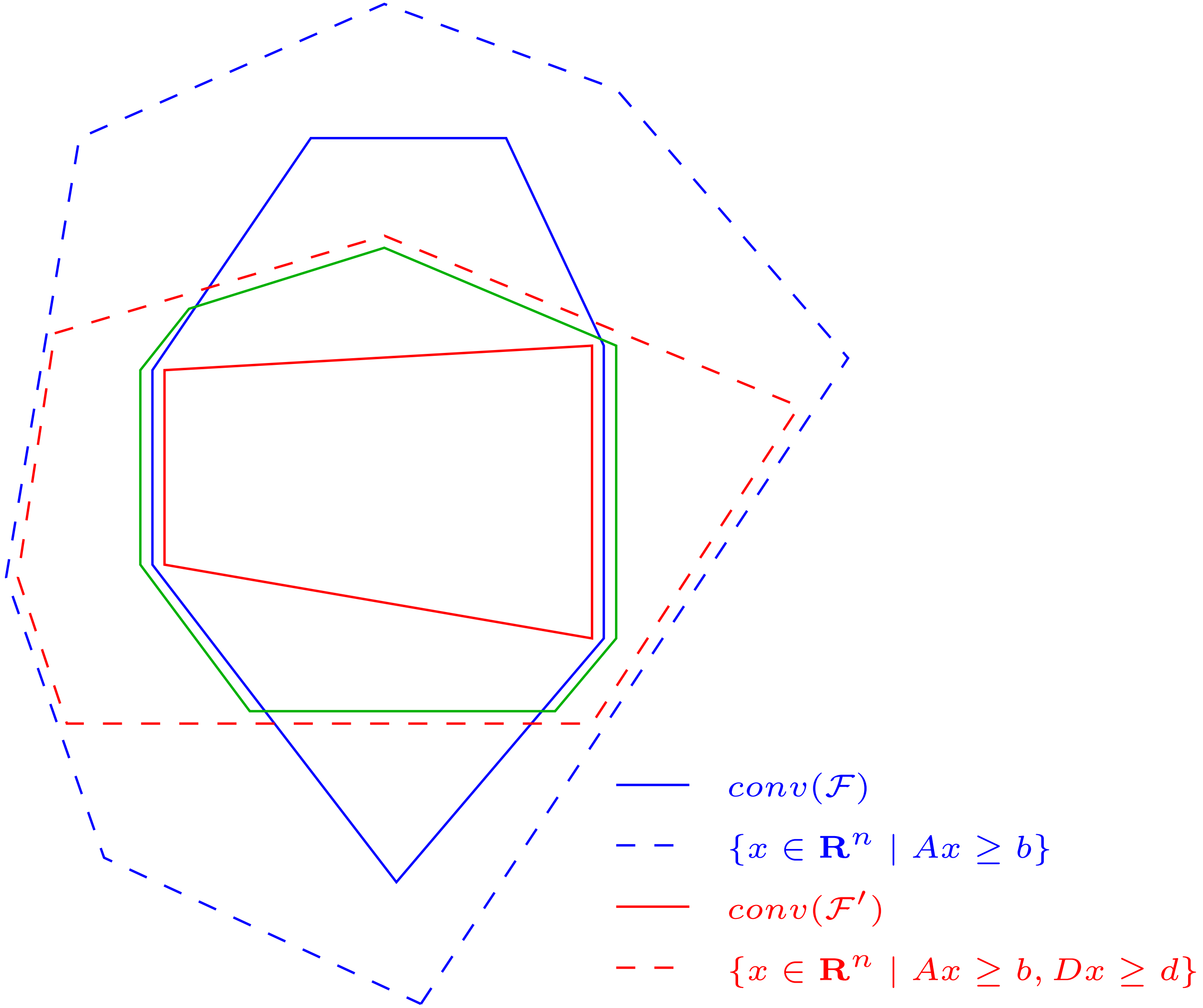
- We have three methods of computing bounds based on our ability to solve a relaxation.
- How do the bounds compare?

$$
\begin{aligned}
&\max_{u \geq 0} \min_{x \in \mathcal{F}} \{(c - uD)x + ud\} && (LR)\\
&= \max_{u_0, u \geq 0} \{u_0 : u_0 \leq (c - uD)x^f + ud, f \in \mathcal{F}\} &&\\
&= \min_{\lambda^f \geq 0} \{\sum_{f \in \mathcal{F}} cx^f \lambda^f : \sum_{f \in \mathcal{F}} Dx^f \lambda^f \geq d, \sum_{f \in \mathcal{F}} \lambda^f = 1\} && (DW)\\
&= \min\{cx \;:\; Dx \geq d, x \in \mathcal{P}\} &&\\
&= \min\{cx : A^I x \geq b^I, Dx \geq d\} && (CP)
\end{aligned}
$$

- All three bounds are the same.



$conv(\mathcal{F})$
$conv(\mathcal{F}')$
$\{x \in \mathbf{R}^n \mid Ax \geq b, Dx \geq d\}$


$conv(\mathcal{F})$
$\{x \in \mathbf{R}^n \mid Ax \geq b\}$
$conv(\mathcal{F}')$
$\{x \in \mathbf{R}^n \mid Ax \geq b, Dx \geq d\}$
$DW/LDBound$

Improving the Bounds

- We have the bound $\min\{cx : A^I x \geq b^I, Dx \geq d\}$.
- This bound may not be good enough to solve difficult instances of CP'.
- One way to improve the bound is to add inequalities from $D^I x \geq d^I$.
- Problem: Separation over $\mathcal{P}'$ is "hard".
- How do we do this?
  - The template approach is to separate these inequalities into classes and derive separation algorithms for individual classes.
  - This is still difficult for most interesting classes.
  - Suppose that separating members of $\mathcal{F}$ from $\mathcal{P}'$ is "easy".
  - Does this situation occur in practice? Yes.

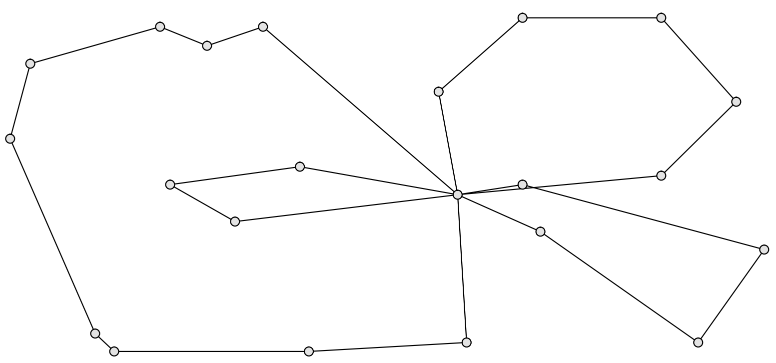
The Vehicle Routing Problem

The VRP is a combinatorial problem whose *ground set* is the edges of a graph $G(V, E)$. Notation:

- V is the set of customers and the depot (0)
- d is a vector of the customer demands
- k is the number of routes
- C is the capacity of a truck

A feasible solution is composed of:

- a partition $\{R_1, \ldots, R_k\}$ of V such that $\sum_{j \in R_i} d_j \leq C,\ 1 \leq i \leq k;$
- a permutation σ_i of $R_i \cup \{0\}$ specifying the order of the customers on route i.



Classical Formulation for the VRP

IP Formulation:

$$
\begin{array}{rcll}
\sum_{j=1}^{n} x_{0j} & = & 2k & (1)\\
\sum_{j=1}^{n} x_{ij} & = & 2 \qquad \forall i \in V \setminus \{0\} & (2)\\
\sum_{\substack{i \in S \\ j \notin S}} x_{ij} & \geq & 2b(S) \quad \forall S \subset V \setminus \{0\},\ |S| > 1. & (3)
\end{array}
$$

$$
\begin{array}{rcl}
b(S) & = & \text{lower bound on the number of trucks required to service } S\\
 & = & \left\lceil \left(\sum_{i \in S} d_i\right)/C \right\rceil \text{ (normally)}
\end{array}
$$

- Relaxations:
  - Multiple Traveling Salesman Problem: Set $C = \sum_{i \in S} d_i$.
  - k-Tree: Relax constraints (2) but leave $\sum_{(i,j) \in E} x_{ij} = n + k$.
- Separation of the capacity inequalities (3) in this formulation is NP-hard.
- Given the incidence vector of an MTSP or a k-Tree, we can easily determine whether it satisfies all of these inequalities.

Dynamic Cut Generation in a Standard Branch and Cut Framework

- Branch and cut is designed specifically to easily incorporate additional strong cutting planes.
- This is one reason why it has been so effective for such a wide range of problems.
- However, it is generally difficult to solve the separation problem for an arbitrary fractional solution.
- We can use decomposition to take advantage of our ability to separate members of $\mathcal{F}$ from $\mathcal{P}'$.

Dynamic Cut Generation in Lagrangian Relaxation

- Consider branch and bound based on Lagrangian relaxation.
- The solution to the Lagrangian dual is a member of $\mathcal{F}$.
- If it is a member of $\mathcal{F}'$, then it is optimal.
- Otherwise, we can attempt to separate it from $\mathcal{P}'$.
- We then "dualize" the newly generated inequality on the fly by adding them to the matrix D of side constraints.
- This (not so well-known) technique has been called *relax and cut*.

Dynamic Cut Generation in Dantzig-Wolfe Decomposition

- Now consider branch and bound based on D-W decomposition.
- Consider the fractional point $\hat{x} = \sum_{f \in \mathcal{F}} \lambda^f x^f$.
- Key Observation: If an inequality is violated by $\hat{x}$, then it must be violated by some x^f such that $\lambda^f > 0$.
- Idea: Generate inequalities violated by some x^f such that $\lambda^f > 0$.
- Add only those that are also violated by $\hat{x}$.
- Adding such inequalities should result in improvement of the bound.

Decomposition-Based Separation Algorithm

<u>**Input:**</u> $\hat{x} \in \mathbf{R}^E$

<u>**Output:**</u> A valid inequality for $\mathcal{P}'$ which is violated by $\hat{x}$, if one is found.

- **Step 0.** Apply separation algorithms and heuristics for $\mathcal{P}$ and $\mathcal{P}'$. If one of these returns a violated inequality, then STOP and output the violated inequality.
- **Step 1.** Otherwise, attempt to decompose $\hat{x}$ into a convex combination of members of $\mathcal{F}$ by solving the LP

$$\min\{\mathbf{0}^T\lambda \mid \sum_{f\in\mathcal{F}} \lambda^f x^f = \hat{x}, \sum_{f\in\mathcal{F}} \lambda^f = 1, \lambda^f \geq 0\}.$$

- **Step 2a.** If a decomposition $\hat{\lambda}$ exists, separate each extreme point x^f such that $\hat{\lambda}_f > 0$ from $\mathcal{P}'$. Return any generated inequalities that are also violated by $\hat{x}$.
- **Step 2b.** If a decomposition does not exist, return a Farkas cut (a, β) for $\mathcal{P}$ that is violated by $\hat{x}$.

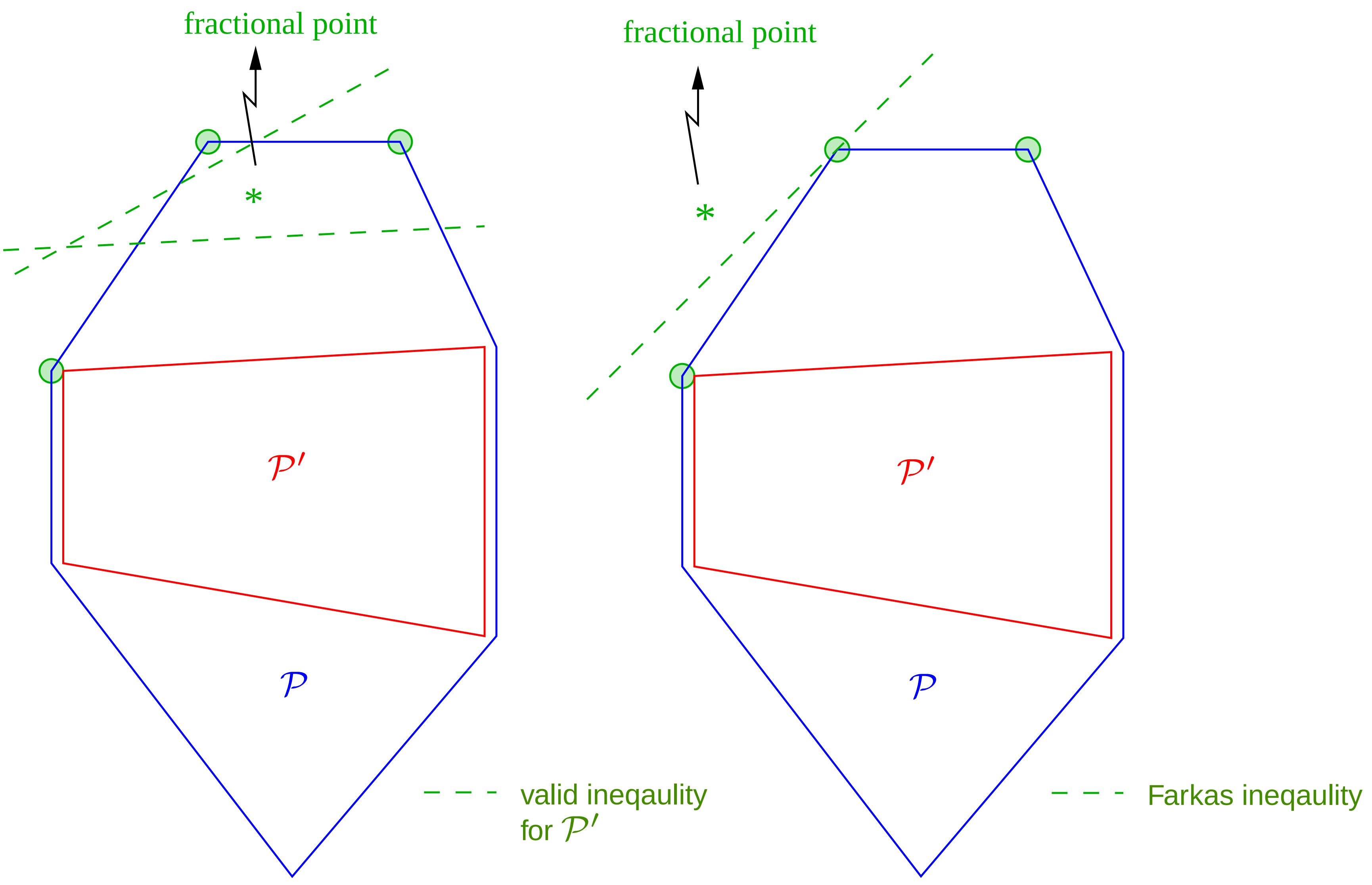
Note: If we separate over $\mathcal{P}$ *exactly*, we can always find a decomposition.

Column Generation Algorithm

Input: $\hat{x} \in \mathbf{R}^E$

Output: Either (1) a valid inequality for $\mathcal{P}$ violated by $\hat{x}$;
or (2) a decomposition of $\hat{x}$ into members of $\mathcal{F}$.

- **Step 1.0.** Generate a matrix T' containing a small subset of promising columns.
- **Step 1.1.** Solve the LP using the dual simplex algorithm over T'. If this LP is feasible, then STOP.
- **Step 1.2.** Otherwise, let r be the row in which the dual unboundedness condition was discovered, and let $(a, -\beta)$ be the r^{th} row of B^{-1}. Solve CP with cost vector a. Let t be the incidence vector of the result. If $at < \beta$, then t is a column eligible to enter the basis. Add t to T' and go to 1.1.
- **Setp 1.3.** Otherwise, the LP is infeasible and $\hat{x}$ does not lie in the convex hull of the (implicitly defined) column set, usually $\mathcal{P}$. In this case, we can still separate $\hat{x}$ from $\mathcal{P}$ by imposing the appropriate Farkas inequality.


fractional point
fractional point
$\mathcal{P}'$
$\mathcal{P}$
$\mathcal{P}'$
$\mathcal{P}$
valid ineqaulity
for $\mathcal{P}'$
Farkas ineqaulity

Projection and Lifting

- Note that we may be able increase efficiency by restricting column generation or projection.
- Only generate columns that "conform" to $\hat{x}$.
  - If $\hat{x}_i = 1$ and $\hat{x} = \sum_{f \in \mathcal{F}} \lambda^f x^f$, then $\hat{x}_i^f = 1, \forall f \in \mathcal{F}$ s.t. $\lambda^f > 0.$
  - If $\hat{x}_i = 0$ and $\hat{x} = \sum_{f \in \mathcal{F}} \lambda^f x^f$, then $\hat{x}_i^f = 0, \forall f \in \mathcal{F}$ s.t. $\lambda^f > 0.$
- Difficulty: Farkas inequalities must be lifted in this case. Can be as difficult as as the original column generation subproblem (an instance of "easy" CP).

Lifting the Farkas Inequalities

- Using a big-M method we define the vector a' to be

$$a_e{}' = \begin{cases} M & if\ e \in E_0 = \{e : \hat{x}_e = 0\}; \\ -M & if\ e \in E_1 = \{e : \hat{x}_e = 1\}; \\ a_e & otherwise \end{cases}$$

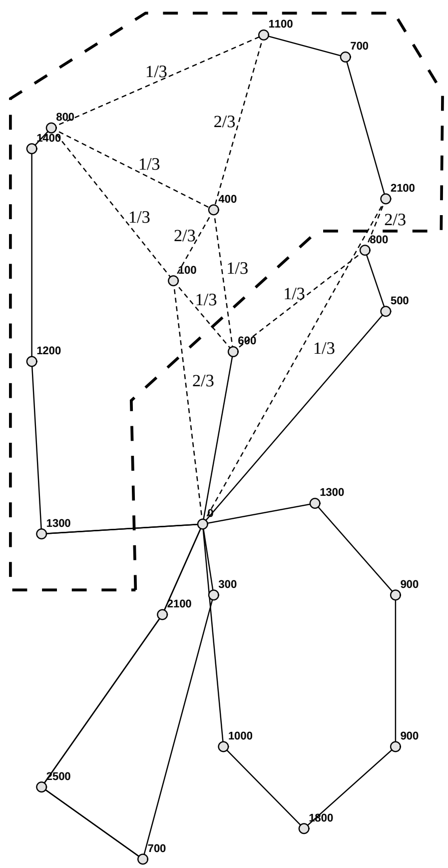
  where (a, β) is the original inequality.
- Then the inequality $a'x \geq \beta - M|E_1|$ is valid for $\mathcal{P}'$ and is violated by $\hat{x}$ as long as $M \geq max\{\beta - ax : x \in \mathcal{P}'\}$. Any lower bound for CP' obtained with cost vector a can hence be used to derive a suitable constant.
- To get stronger coefficients, one can use a sequential lifting procedure based on the same idea (can be expensive).

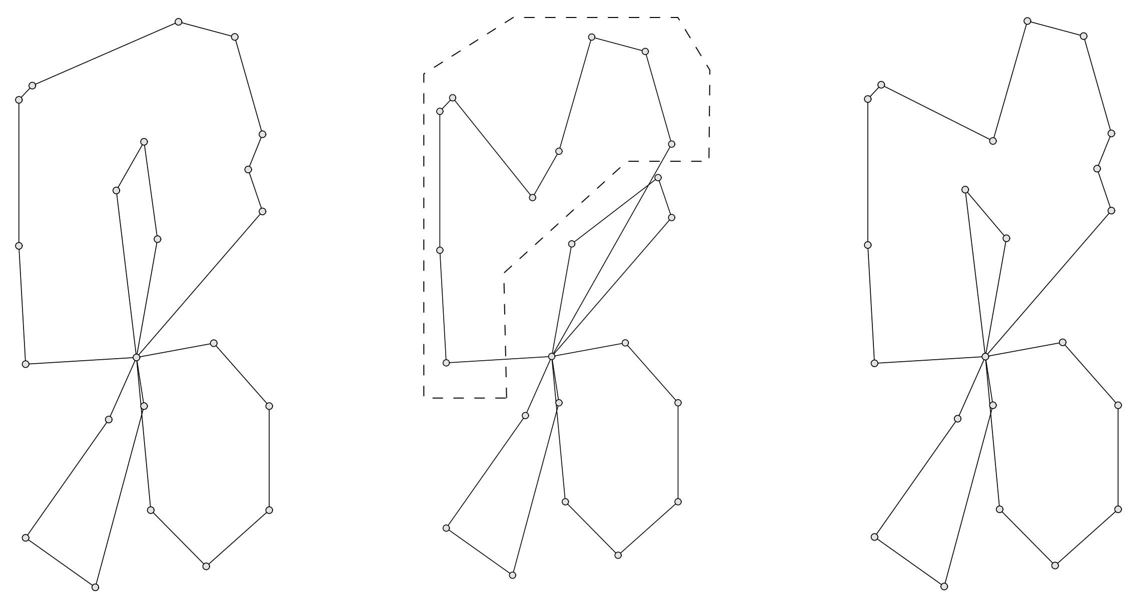
Extensions

- We can also restrict column generation in other ways, for instance we can limit to only columns whose cost does not exceed the current upper bound.
- By restricting column generation, we reduce our chances of finding a decomposition, but strengthen the resulting Farkas inequality.
- Idea: Intentionally restrict column generation in order to generate strong Farkas inequalities.
- For instance, we might try only generating columns corresponding to extreme points of $\mathcal{P}'$!
- This makes it impossible to get a decomposition, but the Farkas inequalities are much stronger.
- However, the column generation subproblem is then an instance of CP'.
- Applegate, Bixby, Chvátal and Cook ’01, *TSP Cuts Which Do Not Conform to the Template Paradigm*

Solving the VRP using MTSP

- We can use this separation algorithm for the capacity constraints with the Multiple Traveling Salesman Problem as the relaxation.
- The MTSP is not polynomial, but can be solved “effectively.”





Solving the VRP using k-Tree

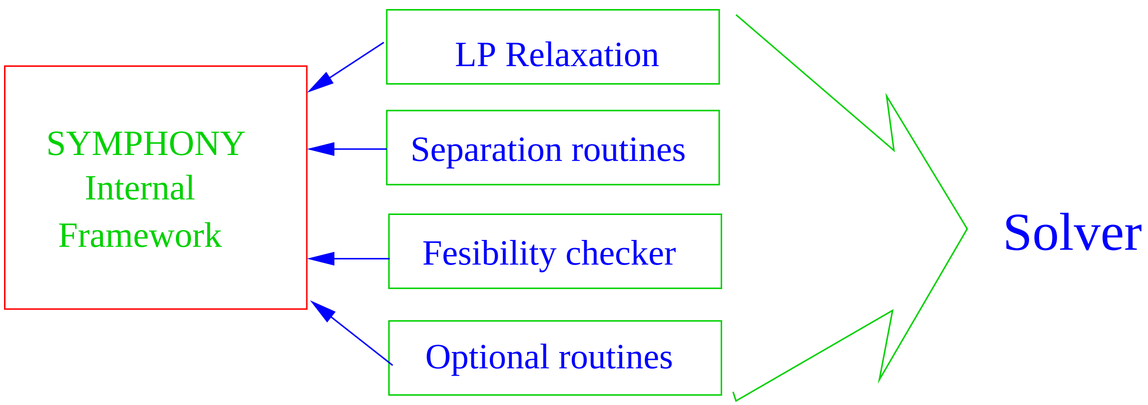
- Another possibility is to use the k-Tree relaxation.
- The k-Tree problem is polynomially solvable (Fisher '94).
- We can separate extreme points of the k-Tree polytope using both capacity constraints and combs.
- Combs can be separated heuristically in polynomial time (Martinhon, Lucena and Maculan - unpublished).

Computational Experiments

- We implemented this algorithm and used it to solve instances of the VRP using the MTSP relaxation.
- The algorithm was implemented using the SYMPHONY branch, cut, and price library.
- Column Generation
  - If the fractional graph was sparse, we tried to generate all conforming columns by brute force.
  - If this failed, we switched over to column generation.
  - We also put an overall time limit on the algorithm.
- Several experiments on different branching rules (on cuts and variables).
- Various methods of projection and lifting of the Farkas inequalities.

SYMPHONY and COIN/BCP

- SYMPHONY (C) and COIN/BCP (C++) are parallel frameworks for branch, cut, and price developed by Ted Ralphs and Laci Ladanyi.
- The user supplies:
  - the initial LP relaxation,
  - separation subroutines,
  - feasibility checker, and
  - other optional subroutines.
- The framework handles everything else.



Some Computational Results

- Ralphs, Kopman, Pulleyblank and Trotter ’02, *On the Capacitated Vehicle Routing Problem*

- The decompositions were found in a small number of iterations and usually involved a small number of columns.
- We observe a significant reduction in the number of search tree nodes examined.
- However, the column generation subroutine and lifting the Farkas inequalities added to the running times.
- In several instances the fractional solution is always outside the MTSP polytope. We might apply MTSP separation directly.
- The MTSP relaxation is too difficult in some cases (k-Tree).

Conclusions

- Overall, these methods seem to retain the advantages of classical decomposition-based methods, but can produce tighter bounds than LR and DW.
- Theoretically, these methods are generally applicable to a wide-range of difficult COPs.
- Computationally, we know little about the performance of these methods. However, it has shown great potential in some settings (VRP/mTSP).

Future Work

- We are currently designing an abstract branch and bound framework for implementing these algorithms.
- The user supplies
  - The solver for the relaxation.
  - The separation routine for members of $\mathcal{F}$ (optional).
- The framework will make it easy to switch between any of these decomposition-based algorithms.
- This will enable more extensive computational studies and allow us to gain important knowledge about these methods.
- All this will be available an open source through the COIN-OR project repository (www.coin-or.org).

Common Optimization INterface for Operations Research

- An initiative to spur the development of open source software for the operations research community.
- Current projects in the COIN repository www.coin-or.org:
  - BCP: a parallel branch-cut-price framework
  - CGL: a cut generation library
  - DFO: a package for solving general nonlinear optimization problems when derivatives are unavailable
  - VOL: the volume algorithm
  - OSI: an open solver interface layer
  - OTS: an open framework for tabu search
  - IPOPT: an interior point algorithm for general large-scale nonlinear optimization