Decomposition-based Methods for Large-scale Discrete Optimization

Ted Ralphs

Department of Industrial and Systems Engineering
Lehigh University, Bethlehem, PA

http://www.lehigh.edu/~tkr2

Outline

- Review of decomposition-based methods
  - Lagrangian Relaxation
  - Dantzig-Wolfe Decomposition
  - Cutting Plane Algorithms
- Incorporating dynamic cut generation
- A decomposition-based cutting plane algorithm
- Extensions
- Computational results
- Conclusions

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 $\min_{S \in \mathcal{F}} c(S)$

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

- 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 $\mathcal{P}$.

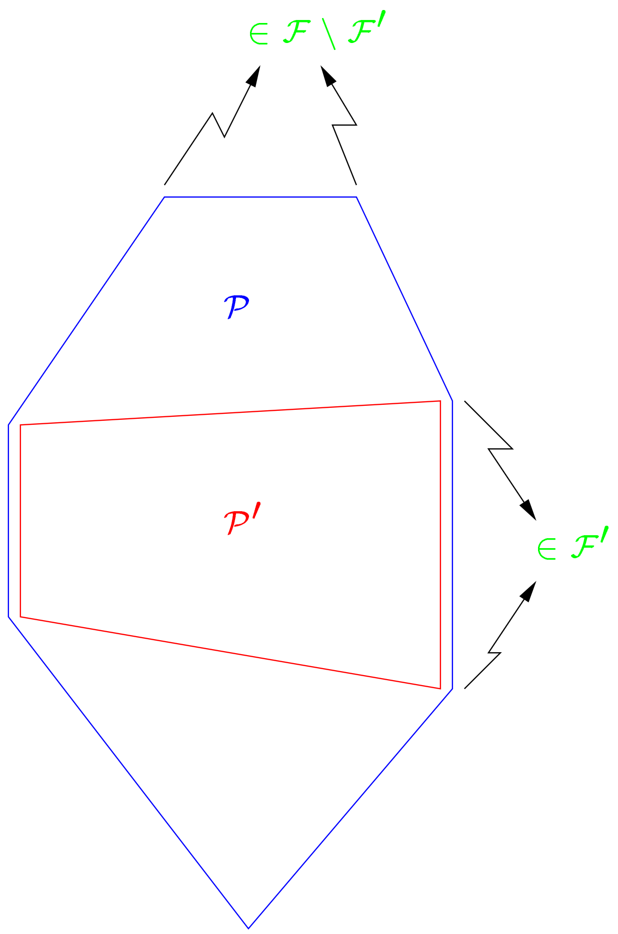


Figure 1: The base polytope $\mathcal{P}$ and its restriction $\mathcal{P}'$

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.

Solving the Lagrangian Dual By Linear Programming

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

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

- 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

$$\min\{x \in \mathbf{R}^n \mid A^I x \geq b^I, Dx \geq d\}$$

.

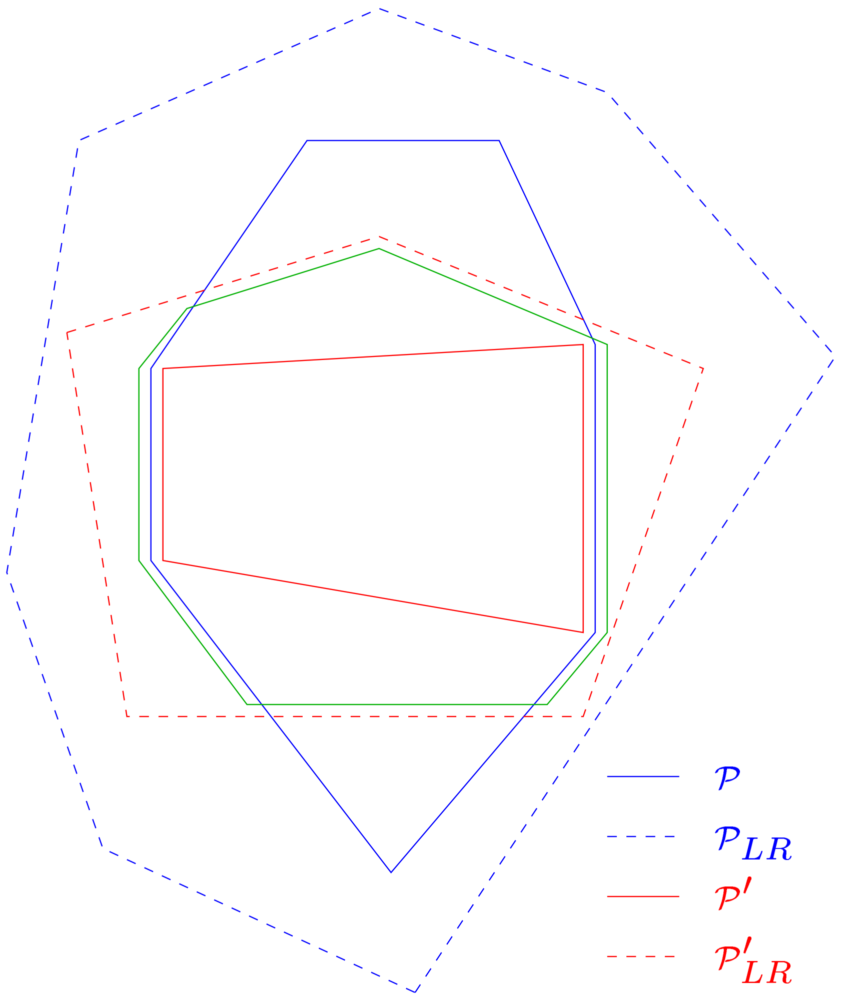
- This bounding scheme can also be embedded into branch and bound.
- In this case, we could branch on fractional variables or any other disjunction.

Comparing the Bounds

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

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

- All three bounds are the same.


$\mathcal{P}$
$\mathcal{P}_{LR}$
$\mathcal{P}'$
$\mathcal{P}'_{LR}$
LR/DW

Improving the Bounds

- We have the bound $\min\{cx : Dx \geq d, A^I x \geq b^I\}$.
- 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:

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

$b(S)$ = lower bound on the number of trucks required to service S (normally $\lceil (\sum_{i \in S} d_i)/C \rceil$).

- If $C = \sum_{i \in S} d_i$, then we have an instance of the Multiple Traveling Salesman Problem.
- Separation for the inequalities in this formulation is NP-hard.
- Given the incidence vector of an MTSP, we can easily determine whether it satisfies all of these inequalities.

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 technique is known as *relax and cut*.
- Problem: The bound may not change (we'll see why).

Dynamic Cut Generation in Dantzig-Wolfe Decomposition

- With D-W decomposition, we can also add dynamically generated inequalities.
- Again, add them to the matrix D of side constraints as they are generated.
- Here, we can do something a little smarter to generate valid inequalities.
- Consider the fractional point $\hat{x} = \sum_{f \in \mathcal{F}} \lambda^f x^f$.
- 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.

Dynamic Cut Generatiomn in a Standard Branch and Cut Framework

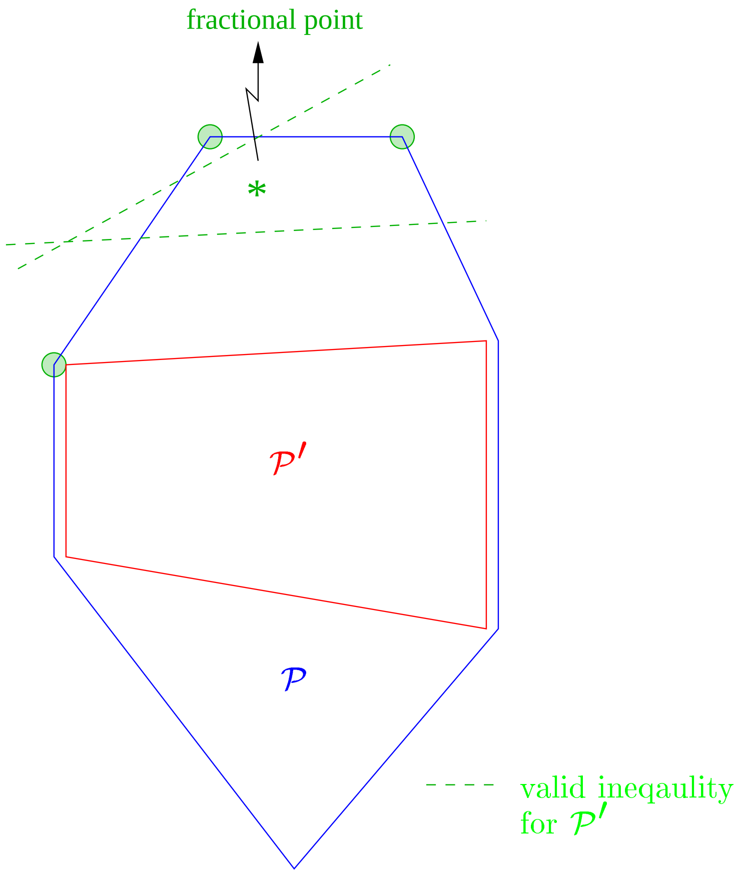
- Branch and cut is designed specifically to easily incorporate additional strong cutting planes.
- This is one reason why it has been so effective for such a wide range of problems.
- However, it is generally difficult to solve the separation problem for an arbitrary fractional solution.
- We can again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

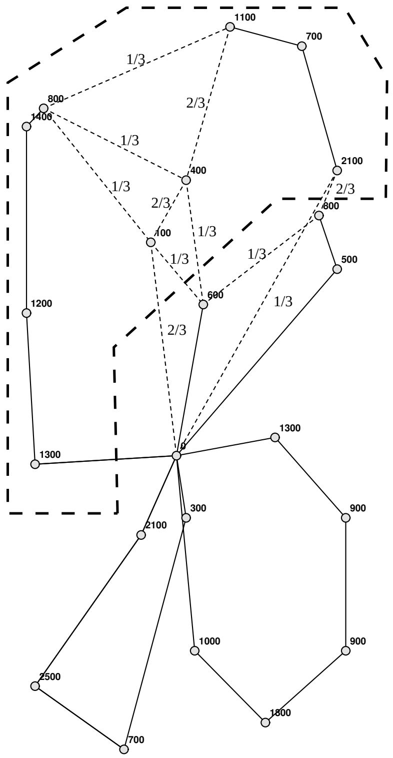
Using Decomposition to Perform Dynamic Cut Generation

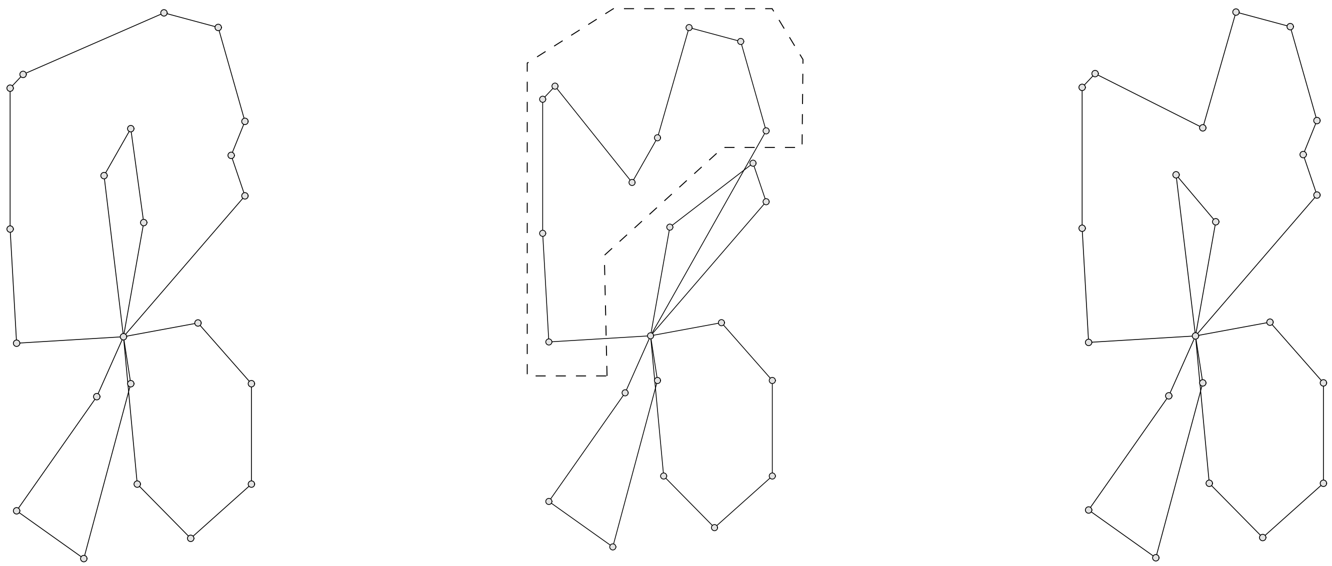
- Instead of computing a Dantzig-Wolfe decomposition at each iteration, only compute it *when needed*.

  1. Use a standard LP-based branch and cut framework.
  2. Try to separate each fractional point using standard procedures.
  3. If the usual procedures fail, try to decompose the current fractional point into a convex combination of members of $\mathcal{F}$.
  4. If successful, separate over members of the decomposition.

- This allows us to take advantage of our ability to separate members of $\mathcal{F}$ from $\mathcal{P}'$, but is more efficient and easier to implement than D-W.


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


1100
700
1/3
800
1400
2/3
1/3
400
2100
1/3
2/3
2/3
1/3
1/3
1/3
500
1/3
1200
2/3
1300
1300
300
2100
900
1000
900
2500
700



A Decomposition-based Separation Algorithm

- We now state the basic algorithm more formally and review some of the implementational details.

  1. Attempt to decompose $\hat{x}$ into a convex combination of members of $\mathcal{F}$ by solving the LP

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

  2. If the LP has a feasible solution $\hat{\lambda}$, separate $f \in \mathcal{F}$ such that $\hat{\lambda}_f > 0$ from $\mathcal{P}'$ to obtain a list of inequalities potentially violated by $\hat{x}$.
  3. Return any generated inequalities that are violated by $\hat{x}$.

- Note that if the fractional solution is inside $\mathcal{P}$, then the LP in Step 1 always has a feasible solution.

- How do we solve the LP in Step 1?

Solving the Decomposition LP

- We can solve this LP by column generation, just as we do with D-W.
- The column generation subproblem is an instance of CP.

1. Start with a small subset of columns (members of $\mathcal{F}$).
2. If decomposition fails, optimize over $\mathcal{P}$ using resulting Farkas inequality (row of B^{-1}).
3. Add new column (member of $\mathcal{F}$), repeat.

Restricting Column Generation

- Note that we may be able increase efficiency by restricting column generation.
- 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. $\lambda^f > 0$.
- We can also restrict column generation in other ways, but we may reduce our chances of finding a decomposition.
- For instance, we can limit to only columns whose cost does not exceed the current upper bound.
- We could also use a "stronger" relaxation if desired.
- Restricting column generation can change the complexity of the column generation subproblem.

No-columns Cuts

- In some instances, *no conforming columns are found*.
- In these cases, the following valid inequality can be imposed.

$$\sum_{e \in E_1} x_e - \sum_{e \in E_0} x_e \leq |E_1| - 1$$

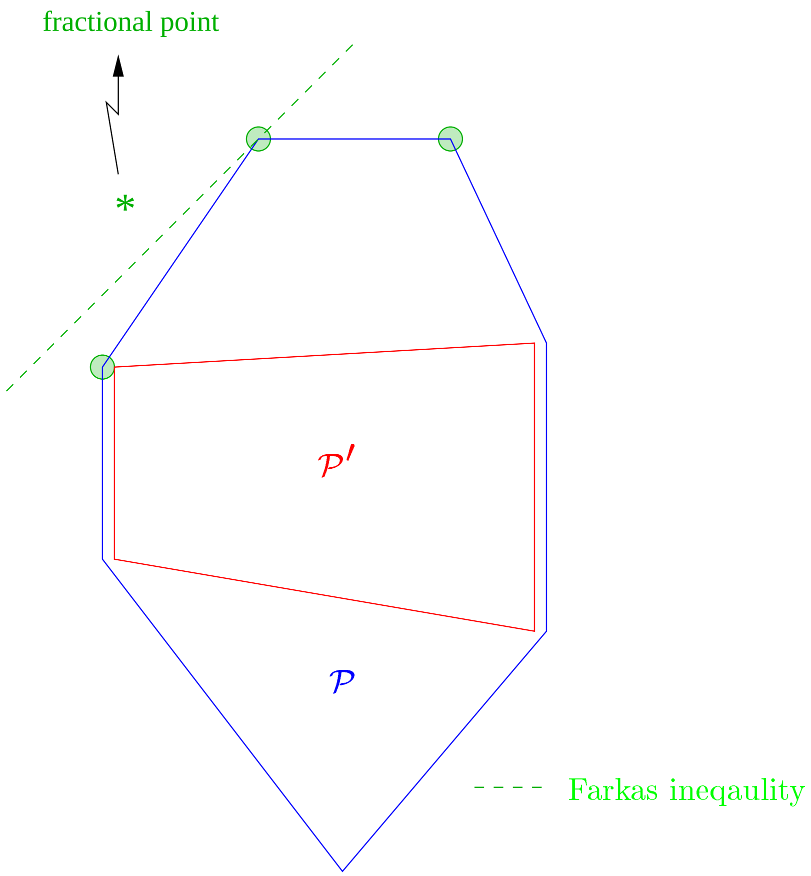
where

$$E_0 = \{e \mid \hat{x}_e = 0\},$$
$$E_1 = \{e : \hat{x}_e = 1\}.$$

- These cuts are similar to the hypo-tour inequalities known for the Vehicle Routing Problem.

Infeasibility of the Decomposition LP

- If the decomposition LP is infeasible, this means that $\hat{x}$ 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 using the Farkas inequality produced by the simplex method.



Strengthening the Farkas Inequalities

- 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'.
- If we restrict column generation, then the resulting Farkas inequality is only valid for the restriction and *may have to be lifted*.
- Lifting can be very expensive.

Solving the VRP

- We can use this separation algorithm for the capacity constraints with MTSP as the relaxation.
- The MTSP is not polynomial, but can be solved "effectively."
- Another possibility is to use the k-tree relaxation.
- The k-tree problem is polynomially solvable.
- We can separate extreme points of the k-tree polytope using both capacity constraints and combs.

Computational Experiements

- 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.
- A few of the details.
  - 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.

Some Computational Results

Conclusions

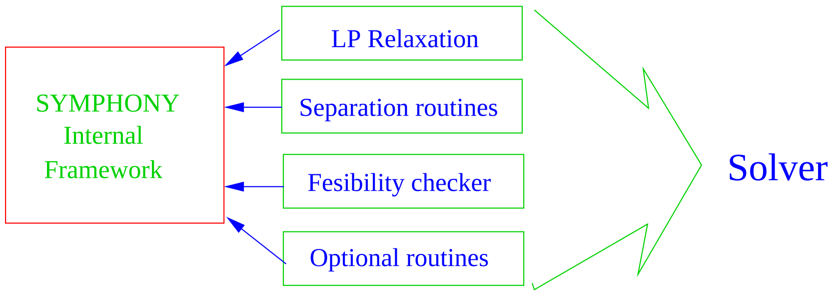
- Conlusions for VRP implementation.
  - The algorithm proved quite effective for separating the capacity constraints.
  - The decompositions were found in a small number of iterations and usually involved a small number of columns.
  - However, we have since developed much better heuristics.
  - The MTSP relaxation is too difficult in some cases.
- Overall, these methods retain the advantages of classical decomposition-based methods, but can produce tighter bounds.
- Generally speaking, we know little about computation with these methods.
- We need a unifying theoretical and computational framework for applying these methods.

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).

Shameless Plug for SYMPHONY and COIN/BCP

- SYMPHONY and COIN/BCP are parallel frameworks for branch, cut, and price.
- The user supplies:
  - the initial LP relaxation,
  - separation subroutines,
  - feasibility checker, and
  - other optional subroutines.
- SYMPHONY handles everything else.
- The source code and documentation are available from www.BranchAndCut.org



Shameless Plug for COIN-OR

- COIN-OR stands for Common Optimization Interface for Operations Research.
- The COIN-OR project is a loose consortium of researchers from both academia and industry.
- We promote and assist in the development of interoperable, open source software for optimization.
- We are also developers ourselves.
- Currently, the repository includes a wide range of software for optimization that is all freely downloadable.
- We are also developing interface standards that will allow various third-party software packages to interoperate.
- Check it out at www.coin-or.org.