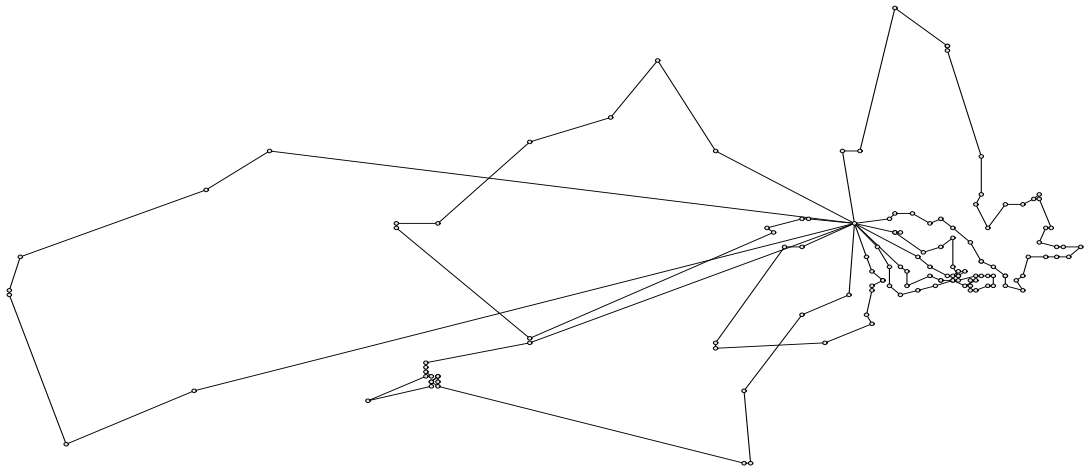
A Generic Separation Algorithm and Its Application to the Vehicle Routing Problem



Presented by:

Ted Ralphs

Joint work with:

Leo Kopman

Les Trotter

Bill Pulleyblank

Outline of Talk

- Introduction
- Description of the Vehicle Routing Problem
- Description of the Decomposition Algorithm
- Implementation
- Computational Results
- Conclusions
- Future Work

Draft paper, source code, data sets, etc. available at
http://BranchAndCut.org.

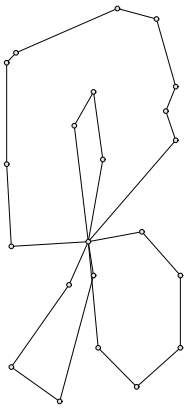
Combinatorial Optimization

A *combinatorial optimization problem* $CP = (E, \mathcal{F})$ consists of

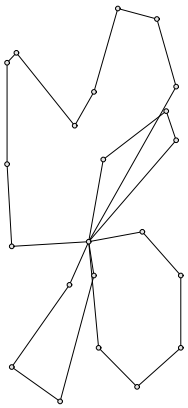
- A *ground set* E, and
- A set $\mathcal{F} \subseteq 2^E$ of *feasible solutions* to CP.

Given a cost function $c \in \mathbf{Z^E}$, we define the cost of $S \in \mathcal{F}$ to be $c(S) = \sum_{e \in S} c_e$. A *subproblem* is defined by $\mathcal{S} \subseteq \mathcal{F}$.

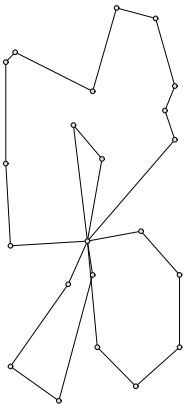
Problem: Find a least cost member of $\mathcal{F}$.



Cost 1100



Cost 1105



Cost 1107

The Vehicle Routing Problem

The VRP is a combinatorial problem $(E, \mathcal{F})$ whose ground set is the edges of a graph $G(E, N)$. Notation:

- d is a vector of the demands.
- N is the set of customers plus the depot (node 0).
- $N^- = N \setminus \{0\}$.
- k is the number of routes.
- C is the capacity of a truck.

A feasible solution is composed of:

- a partition $\{R_1, \ldots, R_k\} \subseteq 2^{N^-}$ 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 in which the customers on route i are to be serviced, $1 \leq i \leq k$.

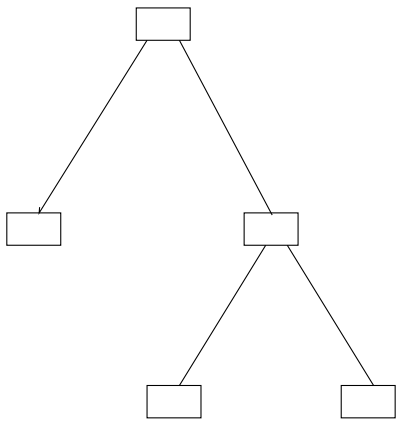
Feasible solutions are those incidence vectors satisfying:

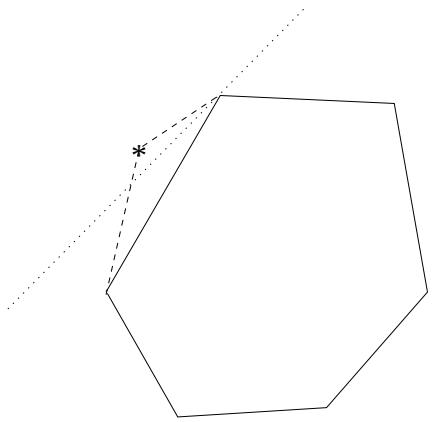
$$\begin{array}{lcll} \sum_{j=1}^{n} x_{0j} & = & 2k & \\ \sum_{j=1}^{n} x_{ij} & = & 2 & \forall i \in N^- \\ \sum_{\substack{i \in S \\ j \notin S}} x_{ij} & \geq & 2b(S) & \forall S \subset N^-,\ |S| > 1. \end{array}$$

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

Solving Hard Combinatorial Models

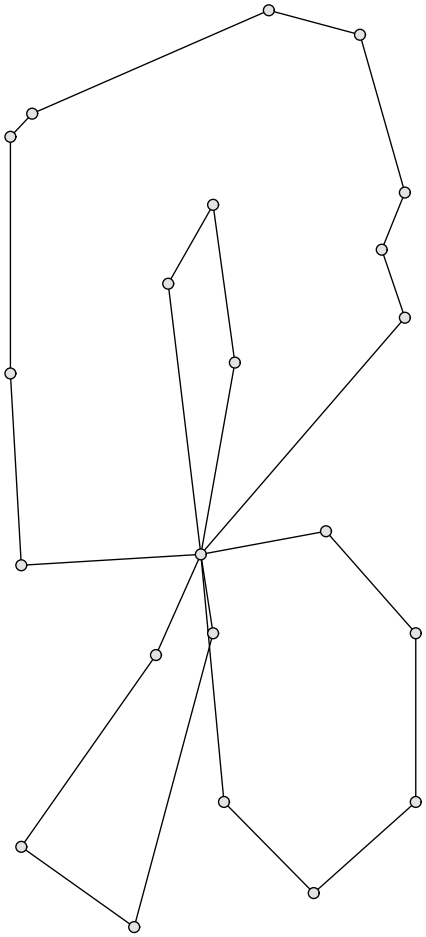
- Branch and cut is the method of choice.
- Vanilla branch and cut algorithm
  - Relax the integrality constraints.
  - Solve the LP relaxation. If infeasible $\Rightarrow$ STOP.
  - If the solution $\hat{x}$ is integral $\Rightarrow$ STOP.
  - Otherwise try to separate $\hat{x}$ from $\mathcal{P}$, the convex hull of integral solutions.
  - If no cutting planes are found, branch to produce subproblems.
- The key to this method is good separation algorithms.
- The remainder of the talk will focus on a new generic approach to this problem.



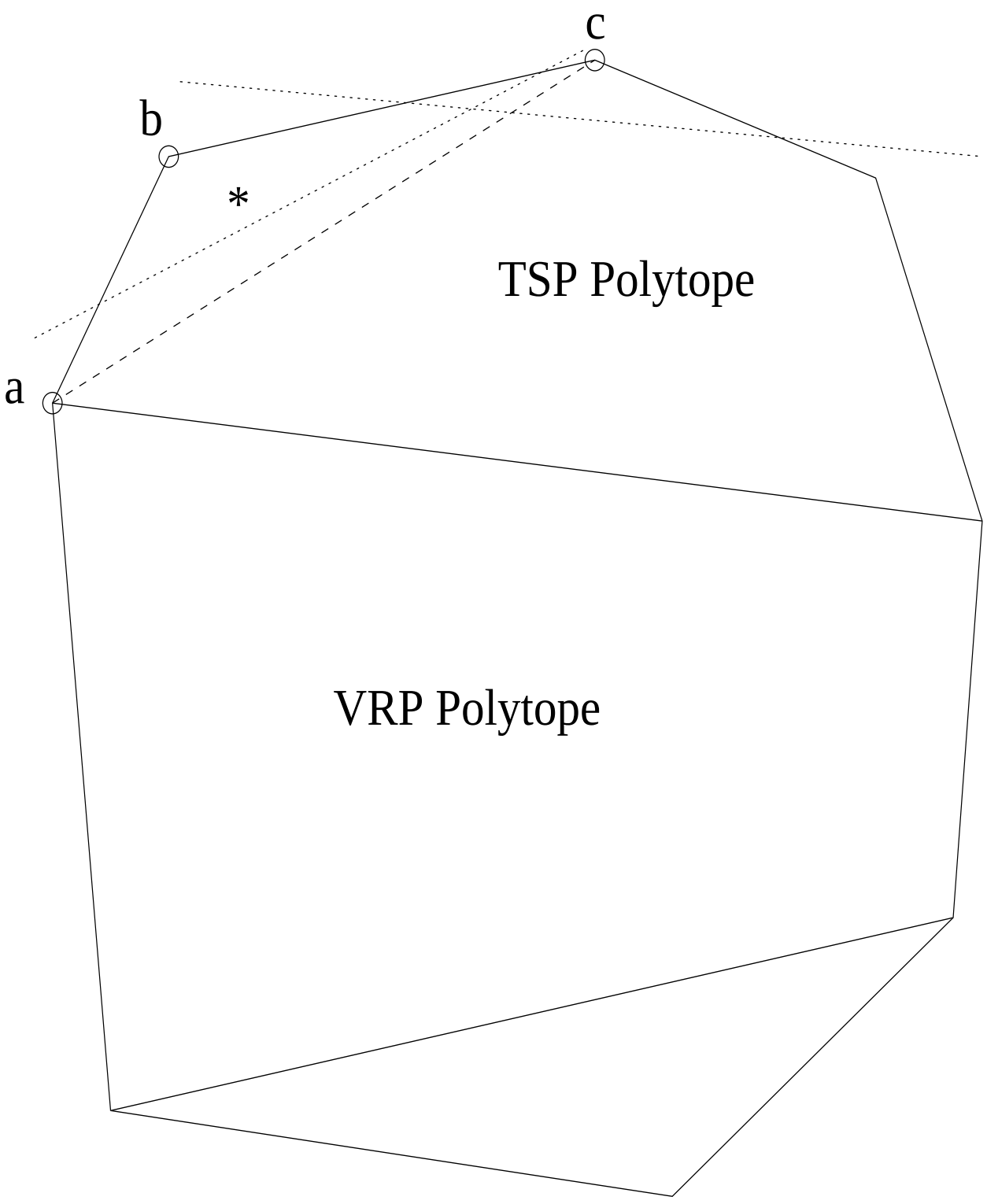


Motivation: Solving Combinatorial Problem with Side Constraints

- The VRP can be thought of as a side-constrained M-TSP.



- Key observation: We can determine in $O(n)$ time whether a particular TSP tour satisfies all the capacity constraints.



valid inequalities

The Decomposition Algorithm

- In general, suppose we have two combinatorial problems

$$\begin{aligned} CP &= (E, \mathcal{F}) \\ CP' &= (E, \mathcal{H}) \end{aligned}$$

  such that $\mathcal{H} \subseteq \mathcal{F}$.

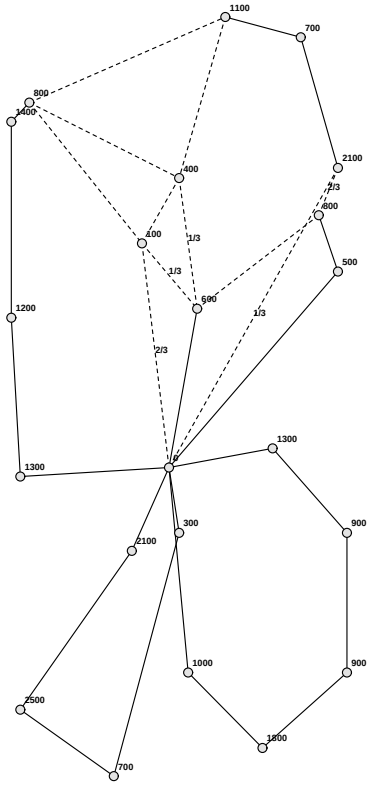
- Attempt to decompose a given fractional point $\hat{x}$ into a convex combination of extreme points of $\mathcal{P} = conv\big(\{x^S : S \in \mathcal{F}\}\big)$ by solving

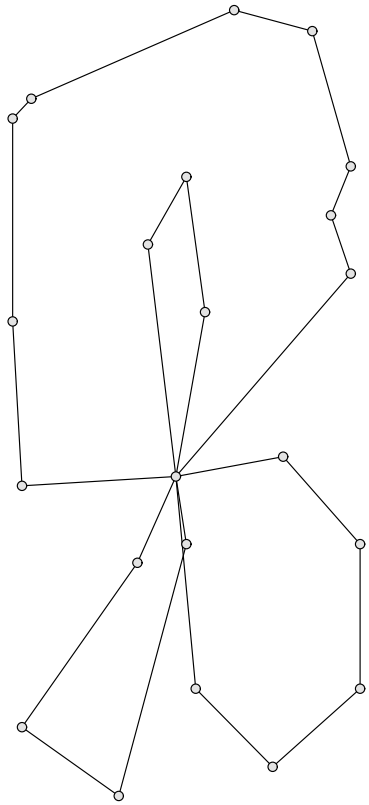
$$\max\{\mathbf{0}^\top \lambda : T\lambda = \hat{x},\ \mathbf{1}^\top \lambda = 1,\ \lambda \geq 0\}$$

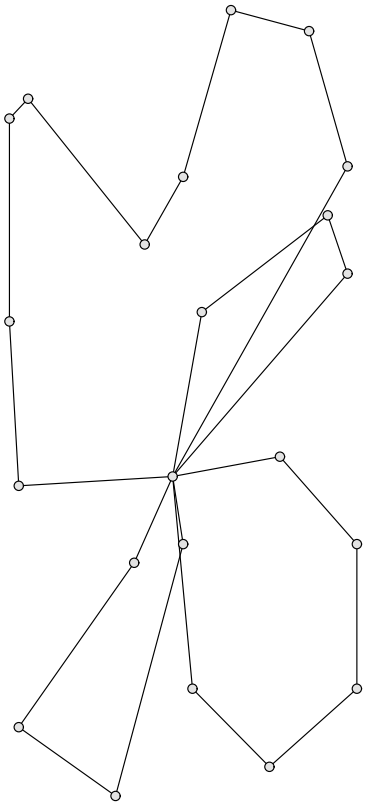
  where T is a matrix whose columns are the extreme points of the polytope $\mathcal{P}$.

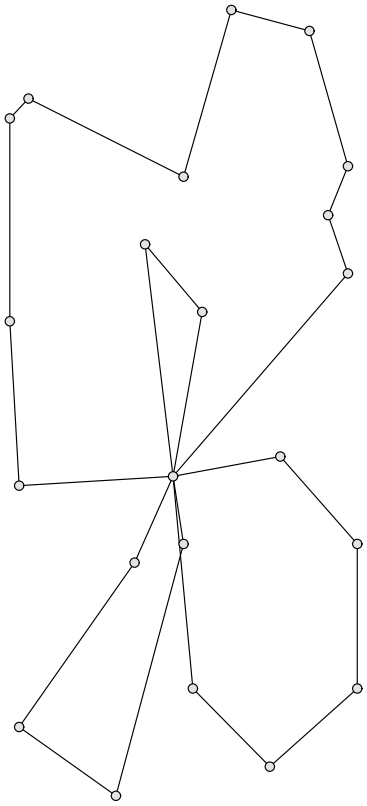
- If successful, then examine the extreme points in the linear combination to obtain a set of possibly violated inequalities.

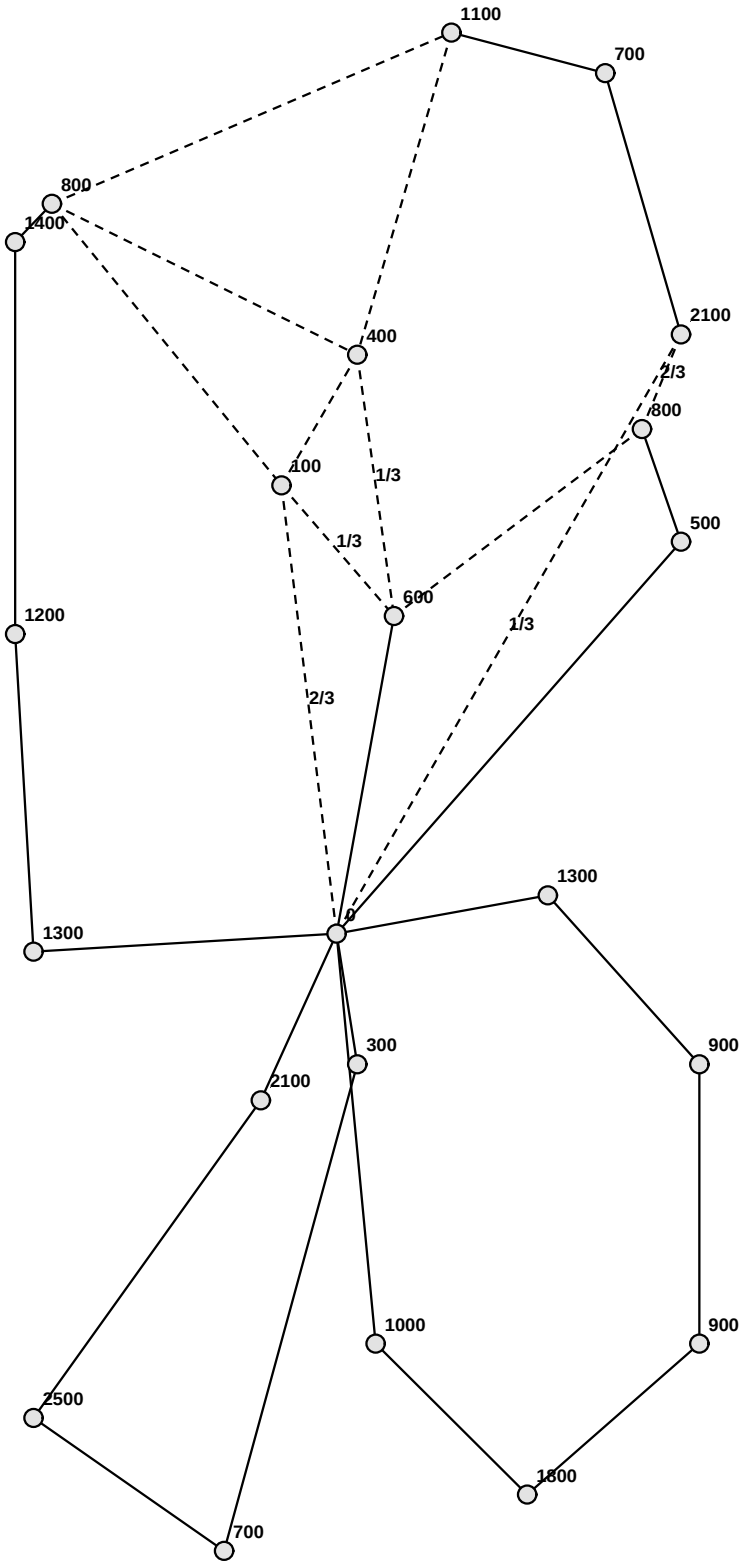
- Otherwise, derive a Farkas inequality that separates $\hat{x}$ from $\mathcal{P}$.


1100
700
800
1400
2100
400
2/3
800
100
1/3
500
1/3
600
1200
1/3
2/3
1300
1300
300
900
2100
1000
900
2500
1800
700








1100
700
800
1400
400
2100
2/3
800
100
1/3
500
1/3
600
1200
1/3
2/3
1300
0
1300
300
900
2100
1000
900
2500
1800
700

The Key Results

Theorem 1 *Let $\hat{x} = T\hat{\lambda}$ be the solution of some decomposition LP. Then if every column t of T with $\hat{\lambda}_t > 0$ satisfies the constraints in some set $\mathcal{S}$, then so must $\hat{x}$.*

Corollary 1 *Let*
$\mathcal{V} = \{(a, \beta) \in \mathcal{S} : a\hat{t} < \beta \text{ for some } \hat{t} \text{ such that } \hat{\lambda}_{\hat{t}} > 0\}$.
Then $\{(a, \beta) \in \mathcal{S} : a\hat{x} < \beta\} \subseteq \mathcal{V}$.

Theorem 2 *Given $\hat{x} \in \mathbf{R}^n$, we have the following mutually exclusive alternatives:*

(I) *There exists* $p \in \mathbf{Z}^+,\ 2 \leq p \leq |E|;\ \lambda_1, \ldots, \lambda_p,\ \sum_{i=1}^{p} \lambda_i = 1,\ \lambda_i > 0 \text{ for } i = 1, \ldots, p;$ *and* $S_1, \ldots, S_p,\ S_i \in \mathcal{F} \text{ for } i = 1, \ldots, p$ *such that* $\sum_{i=1}^{p} \lambda_i x^{S_i} = \hat{x}$.

(II) *There exists* $y \in \mathbf{R}^E$ *and* $\gamma \in \mathbf{R}$ *such that* (y, γ) *is a valid inequality for* $\mathcal{P}$ *but* $y\hat{x} < \gamma$.

Implementing the Algorithm

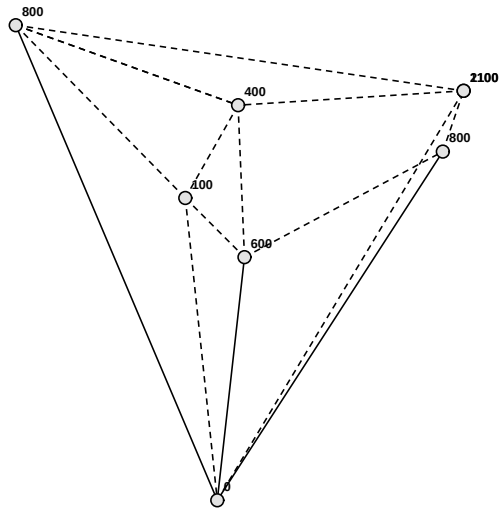
- **Problem:** The matrix T can be large and is difficult to generate.

- **Idea 1:** Use dynamic column generation.
- **Disadvantage:** Column generation subproblem can still be difficult.

- **Idea 2:** Project the problem into a lower dimensional space.
- **Disadvantage:** Resulting Farkas inequalities must be lifted.

Matrix Generation

- **Projection:** Fix all all zero- and one-edges.
- **Motivation:** Only columns "conforming" to $\hat{x}$ can be involved in a decomposition.
- **Method:**
  - Generate "seed" columns using a depth-first search of the support graph, following all one-edges.
  - After generating "seeds", switch to column generation.



Column Generation

Step 1 Generate a matrix T' containing a small subset of columns from T.

Step 2 Solve the Decomposition LP with T' using the dual simplex algorithm. If this LP is feasible, STOP.

Step 3 Otherwise, let r be the row in which the dual unboundedness condition was discovered, and let $\mu \in \mathbf{R}^{n+1}$ be the r^{th} row of B^{-1}.

Step 4 Solve CP with cost vector a, consisting of the first n components of μ. Let t be the incidence vector of the result. If $ct < -\mu_{n+1}$, then t is a column eligible to enter the basis. Add t to T' and go to Step 2.

Step 5 Otherwise, $(c, -\mu_{n+1})$ is a valid inequality for $\mathcal{P}$ which is violated by $\hat{x}$.

No-columns Cuts

If no columns are found, then the following cut can be imposed:

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

where

$$E_0 = \{e : \hat{x}_e = 0\}, \tag{2}$$

$$E_1 = \{e : \hat{x}_e = 1\}. \tag{3}$$

Observation:

If we can generate all conforming, feasible (low cost) columns, we can ALWAYS generate a no-columns cut! We may also get a new upper bound.

Lifting the Farkas Inequalities

Define the vector a' to be

$$a_e{}' = \begin{cases} M & \text{if } e \in E_0; \\ -M & \text{if } e \in E_1; \\ a_e & \text{otherwise} \end{cases} \tag{4}$$

where (a, α) is the original inequality.

The inequality

$$a'x \geq \alpha - M|E_1| \tag{5}$$

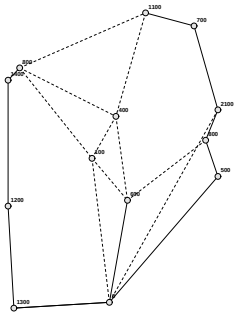
is valid for $\mathcal{P}'$ and is violated by $\hat{x}$ as long as

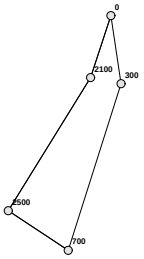
$$M \geq max\{\alpha - ax : x \in \mathcal{P}'\}. \tag{6}$$

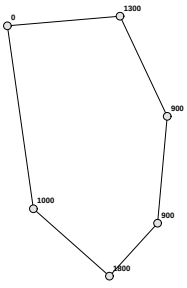
A sequential lifting procedure will generate stronger coefficients, but is expensive.

Overall Cut Generation Strategy

- The degree constraints are always enforced.
- Separation is done only for the capacity constraints from the IP formulation.
- We use the following separation strategy:
  - If the solution is integral, check for connectedness and capacity violations.
  - Otherwise, find the (2-edge) connected components of $G(\hat{x}) \setminus \{0\}$.
  - Iteratively shrink edges in each component with weight greater than or equal to one.
  - Failing that, apply heuristics to each component.
  - Finally, apply the *Decomposition Algorithm*, if appropriate.







Implementation

- We used the SYMPHONY parallel branch, cut, and price framework of Ralphs and Ladanyi to implement a branch and cut algorithm.
- Various branching strategies for cuts and variables and separation strategies were tested.
- We implemented decomposition as follows:
  - A time limit was placed on the column generation phase.
  - It was only re-run within a search node if the gap had narrowed by at least a specified amount since the last call.
  - Column generation and lifting was done using the CONCORDE TSP solver.
  - The bigM method was used to compute lifting coefficients.

Observations and Conclusions

- Finding a decomposition or a set of (projected) Farkas inequalities is not very expensive.
- Effectively utilizing that information once it is obtained is the hard part.
- The algorithm is better at locating violated capacity constraints than previous heuristics *as long as we are inside the TSP polytope*.
- When used in combination with other heuristics, it effectively locates additional inequalities and reduces the size of the tree, but at a price.
- The Farkas inequalities are expensive to lift "well" and weak otherwise.
  - Low dimensional subspaces produce cuts that are "localized".
  - Sequential lifting is too expensive.
  - Generating T without projection (in order to avoid lifting) is also expensive.

Computational Results

	Without Decomposition		With Decomposition		
problem	Tree Size	Wallclock	Tree Size	Wallclock	TreeSize
E-n13-k4	3	0.20	3	0.28	3
E-n22-k4	1	0.15	1	0.19	1
E-n23-k5	1	0.10	1	0.17	1
E-n30-k3	100	6.03	102	12.56	100
E-n31-k7	7	1.06	5	2.45	5
E-n33-k4	1	0.44	1	0.60	1
bayg-n29-k4	4	0.55	4	1.00	4
bays-n29-k5	5	0.96	5	1.24	5
ulysses-n16-k3	1	0.12	1	0.15	1
ulysses-n22-k4	1	0.14	1	0.20	1
gr-n17-k3	1	0.10	1	0.15	1
gr-n21-k3	1	0.29	1	0.34	1
gr-n24-k4	6	1.23	6	5.37	6
fri-n26-k3	6	0.44	6	0.77	6
swiss-n42-k5	17	4.74	17	15.11	17
A-n32-k5	10	1.43	10	5.91	7
A-n33-k5	5	2.73	7	9.80	9
A-n33-k6	21	5.38	13	17.13	34
A-n34-k5	21	5.61	25	29.34	12
A-n36-k5	125	28.31	78	97.92	66
A-n37-k5	18	7.39	18	42.01	8
B-n31-k5	1	0.30	1	0.43	2
B-n35-k5	1	0.41	1	0.57	1
B-n38-k6	35	7.78	20	26.58	22
B-n39-k5	1	0.69	1	0.88	1
B-n41-k6	37	7.53	45	58.90	45
B-n44-k7	1	1.26	1	1.52	1
B-n45-k5	35	14.67	59	83.43	28
B-n50-k7	2	0.71	2	1.33	4
B-n51-k7	184	41.08	181	182.47	164
B-n52-k7	6	1.84	6	11.21	8
B-n56-k7	30	9.74	29	42.73	38
B-n64-k9	20	18.09	17	50.56	14
Total	708	171.49	669	703.28	617

Results with the sequential version of SYMPHONY on a Pentium II running Solaris with CPLEX.

Future Work

- Use the decompositions to find more and better cuts.
- Figure out how to better handle the Farkas inequalities (lifting, etc.)
- Use TSP cutting planes to push the solution into the VRP polytope when decomposition fails.
- Try a different relaxation, such as the Assignment Problem.
- Use a different projection for the matrix T.
- Implement a column pool, analogous to the cut pool, to supplement column generation.
- Find new classes of problems to which the algorithm can be applied.