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

Presented by:

Ted Ralphs

Joint work with:

Leo Kopman

Les Trotter

Outline of Talk

- Introduction
- Description of the Vehicle Routing Problem
- Description of the Decomposition Algorithm
- Extensions to the Basic Algorithm
- Computational Results
- Conclusions

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 $c \in \mathbf{Z}^{\mathbf{E}}$, 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 $\mathcal{F}$.

Solution: Apply branch and cut techniques.

Generic Branch and Cut Algorithm for Combinatorial Optimization

Input: $(E, \mathcal{F})$, $c \in \mathbf{Z}^E$, $\alpha \in \mathbf{Z}$ and $\overline{s} \in \mathcal{F}$ such that $c(\overline{s}) = \alpha$.
Output: A least cost member s^* of $\mathcal{F}$.

1. Create an LP relaxation R^0 consisting of inequalities valid for the polytope $\mathcal{P} = conv(\mathcal{F})$.

2. Set the candidate list $\mathcal{C} = \{\mathcal{R}^0\}$.

3. REPEAT UNTIL $\mathcal{C} = \emptyset$

- Select a subproblem $\mathcal{S}^i$ defined by incidence vectors in $\mathcal{F}$ that are feasible solutions to the corresponding LP relaxation $\mathcal{R}^i$ from $\mathcal{C}$. Set $\mathcal{C} \to \mathcal{C} \setminus \mathcal{R}^i$.
- Iteratively solve and augment $\mathcal{R}^i$ with additional violated inequalities valid for $\mathcal{P}$ until no more can be found.
- If $\mathcal{R}^i$ becomes infeasible or its optimal value exceeds $\alpha - 1$, then *prune* the subproblem.
- Otherwise, if the optimal solution vector $\hat{x}$ is integral check it for membership in $\mathcal{F}$. Update α and $\bar{s}$ if $\hat{x} \in \mathcal{F}$ and $c(\hat{x}) < \alpha$.
- If $\hat{x}$ is infeasible, branch by partitioning $conv(\mathcal{S}^i)$ using 1 or more hyperplanes and add the new subproblems to $\mathcal{C}$.

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:

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

$b(S)$ = 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 the VRP Using COMPSys

The Master Process:

- Upper bounding consists of multiple heuristics run in parallel in several phases:
  - clustering,
  - routing, and
  - exchanging.
- Construction of the root node consists of:
  - selection of initial edge set, and
  - preprocessing.

The Tree Manager: Completely generic.

The Cut Pool: Generic except for cut storage data structures.

The LP Solver:

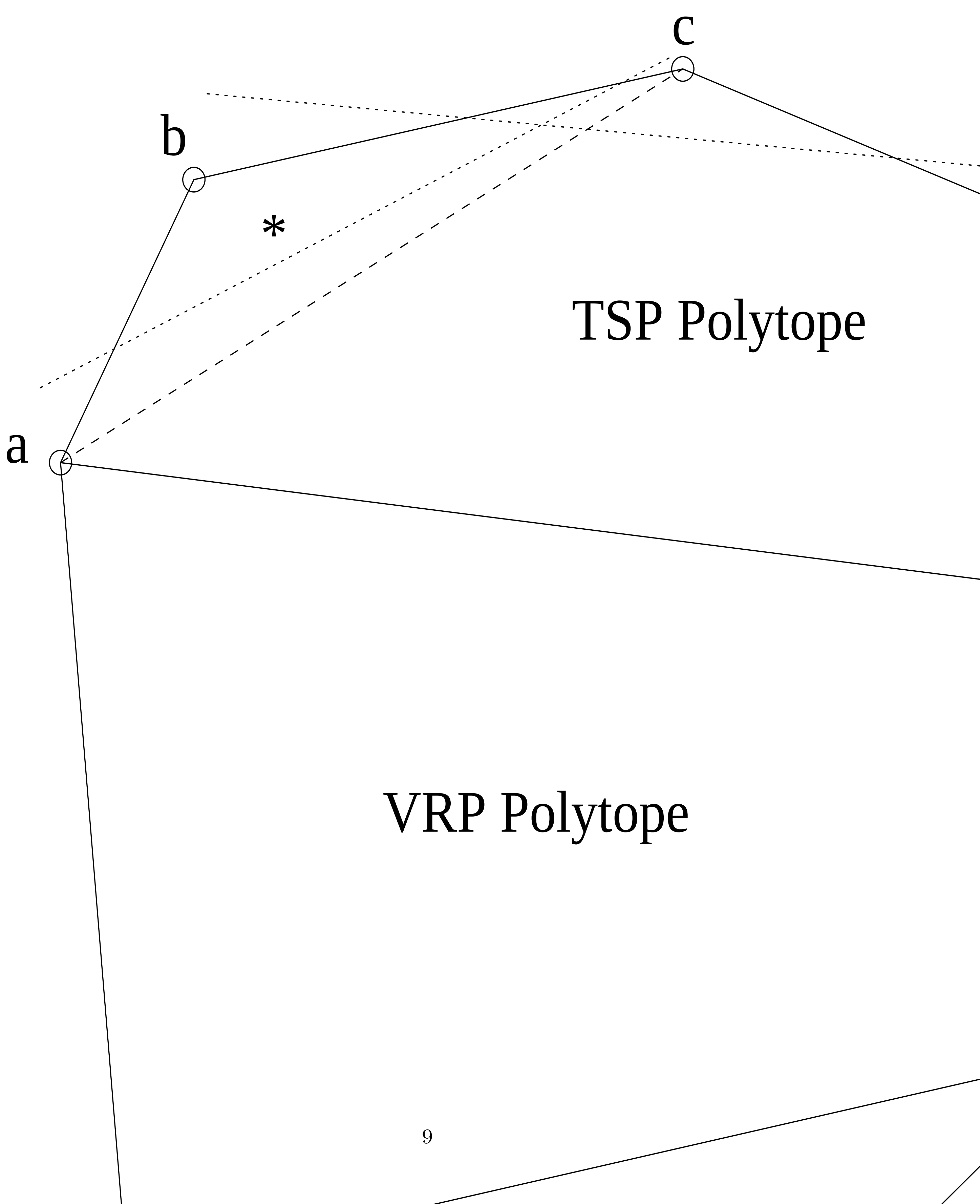
- Logical fixing, and
- Column generation.

Cut Generation for the VRP

- 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 the support graph without the depot.
  - Shrink all edges in the support graph with weight greater than one and look for violated constraints.
  - Failing that, apply the greedy shrinking algorithm of Augerat, et al.
  - Finally, apply the *Decomposition Algorithm.*

Combinatorial Problems with Side Constraints (Why is the VRP harder than the TSP?)

- Feasible solutions to the VRP, called *routings*, can be thought of as TSP tours on a slightly expanded graph obtained by duplicating the depot $k-1$ times.
- The set of TSP tours on this graph that also obey the capacity constraints corresponds to routings for the corresponding VRP.
- Hence, the TSP can be thought of as a relaxation of the VRP.
- In general, suppose we have two combinatorial problems, $CP = (E, \mathcal{F})$ and $CP' = (E, \mathcal{H})$ such that $\mathcal{H} \subseteq \mathcal{F}$. CP is then a relaxation of CP'.
- We call $ax \geq \beta$ a *side constraint* if it is a valid inequality for $CPP' = conv\Big(\{x^S : S \in \mathcal{H}\}\Big)$ but not for $CPP = conv\Big(\{x^S : S \in \mathcal{F}\}\Big)$.
- The capacity constraints constitute a family of side constraints for the TSP with respect to the VRP.
- Key observation: We can determine in $O(n)$ time whether a particular TSP tour satisfies all the capacity constraints.


c
b
*
a
TSP Polytope
VRP Polytope

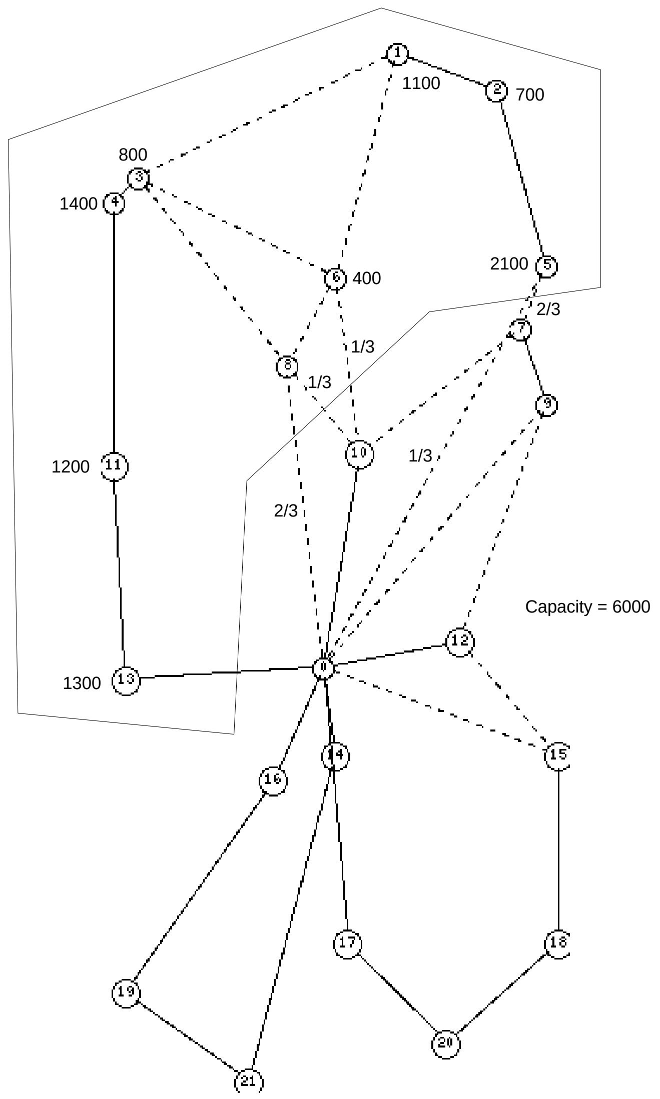
The Decomposition Algorithm

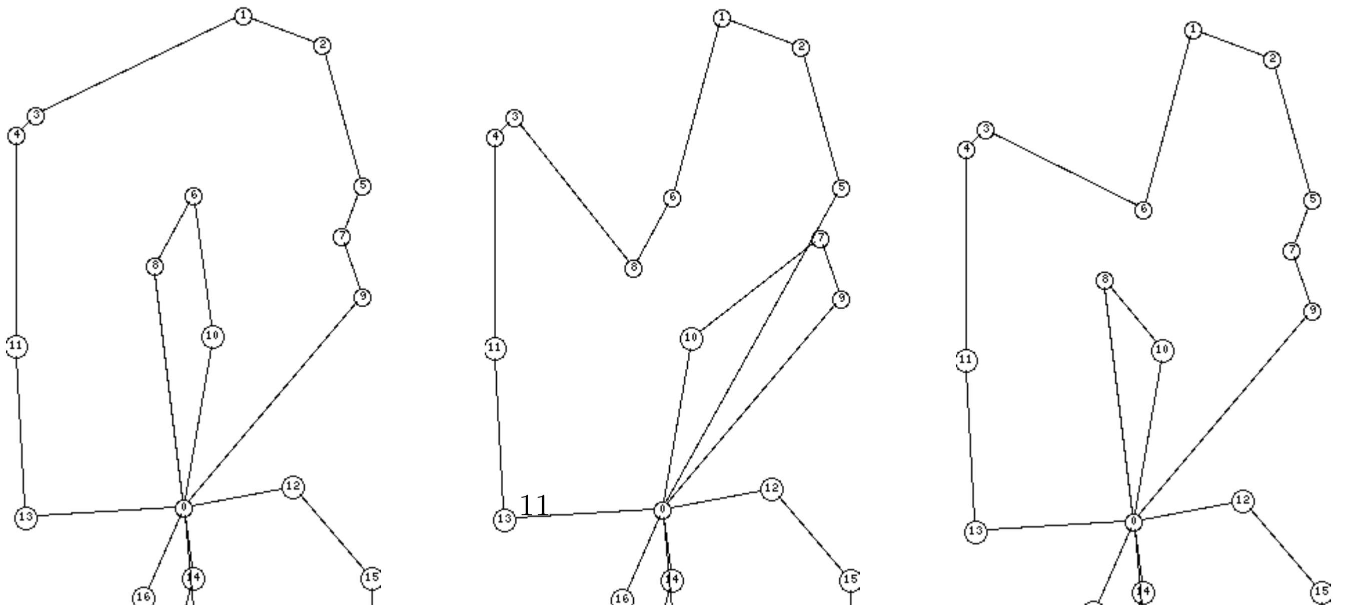
- Attempt to decompose the fractional solution $\hat{x}$ into a convex combination of extreme points of CPP by solving the *Decomposition LP* defined by

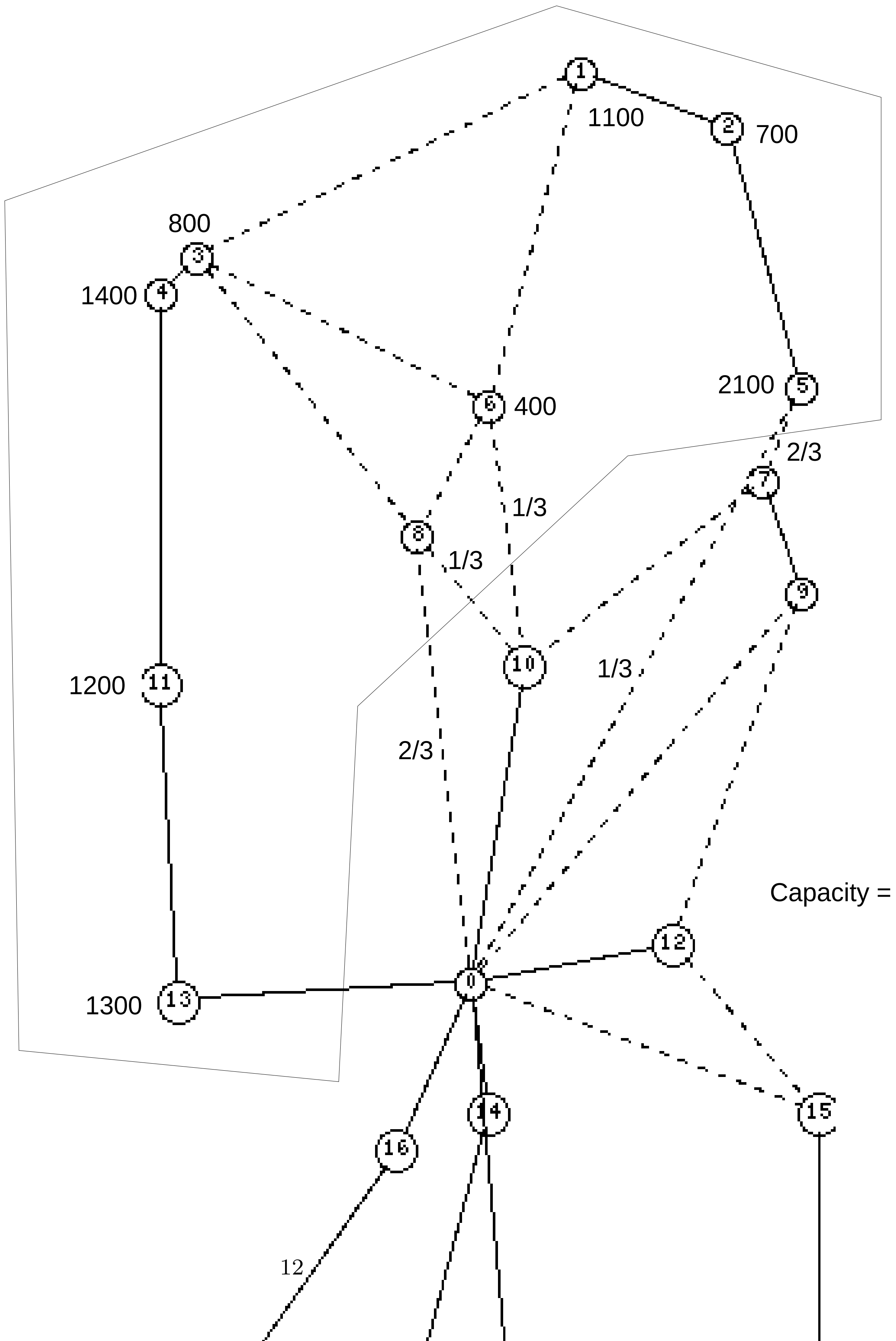
$$\max\{\mathbf{0}^\top \lambda : T\lambda = \hat{x},\ \mathbf{1}^\top \lambda = 1,\ \lambda \geq 0\}$$

  where T is a matrix whose columns are the extreme points of the polytope $CPP.$ This is related to Dantzig-Wolfe decomposition.

- If successful, then separate over the extreme points in the linear combination to obtain a set of possibly violated inequalities. Otherwise, derive an inequality which separates $\hat{x}$ from CPP.


1100
700
800
1400
400
2100
2/3
1/3
1/3
1/3
1200
2/3
Capacity = 6000
1300




1
2
1100
700
800
3
1400
4
5
2100
6
400
2/3
7
1/3
8
1/3
9
10
1/3
1200
11
2/3
Capacity =
12
0
1300
13
14
15
16

Solving the Decomposition LP

In theory, we can solve the decomposition LP by 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, then we have a decomposition.

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 c, 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 1.

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

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 $\mathcal{S}$, 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* CPP *but $y\hat{x} < \gamma$.*

Extensions of the Basic Algorithm

- The main drawback of the basic algorithm is the difficulty of generating the matrix T.
- Possible solution: If the fractional graph is sparse, generate columns using a recursive search.
- The enumeration can optionally be limited to columns that
    1. are feasible for the constrained problem,
    2. have cost less than the current upper bound, and
    3. "conform" to the current fractional solution.
- Advantages of incomplete enumeration
    - Many fewer columns – quicker enumeration.
    - (Hopefully) Stronger Farkas cuts.
    - If no columns are found, then we can impose a *No-columns Cut.*

No-columns Cuts

If no feasible columns are found that conform with the current fractional solution, then the following cut can be imposed:

$$\sum_{e \in R} x_e - \sum_{e \in E \setminus E(\hat{x})} x_e \leq |R| - 1 \tag{1}$$

where $R = \{e : \hat{x}_e = 1\}$ and $E(\hat{x})$ is the edge set of the support graph of $\hat{x}$ of the current fractional solution.

Short proof:
$\sum_{e \in R} x_e \leq |R|$ and $\sum_{e : \hat{x}_e = 0} x_e \geq 0$ are true for any vector $x \in \{0,1\}^E$. Matrix T being vacuous implies that $\sum_{e \in R} x_e = |R|$ and $\sum_{e : \hat{x}_e = 0} x_e = 0$ are never simultaneously true for an integral vector belonging to the polytope containing the optimal solution. Hence at least one of $\sum_{e \in R} x^*_e \leq |R| - 1$, $\sum_{e : \hat{x}_e = 0} x^*_e \geq 1$ must be true.

VRP Implementatiom

- We used the COMPSys library to implement a branch and cut algorithm for the VRP.
- Various branching strategies for cuts and variables were tested.
- For computational efficiency, we implemented decomposition as follows:
  - It was only called when all other algorithms had failed.
  - It was run independently on each connected component of $G(\hat{x}) \setminus \{0\}$.
  - It was only run on components whose density is below a specified threshold.
  - All one-paths were contracted prior to running the algorithm.
  - It was only re-run within a search node if the gap had narrowed by at least a specified amount since the last call.

Observations

- When calling decomp on sparse graphs, we can almost always generate a no-columns cut.
- Both the no-columns cuts and the Farkas cuts are relatively weak in their current form and only appear to induce small local changes in the solution.
- If allowed to run on denser graphs, adding these cuts does reduce the size of the tree significantly, but running times are slow.
- Strong branching is extremely effective for the VRP

Computational Results for the VRP

Problem	Tree Size	Tree Depth	CPU sec	Wallclock
eil13	15	4	0.16	1.23
eil22	3	1	0.09	0.73
eil23	5	2	0.05	0.65
eil30	7993	31	169.63	240.34
eil31	277	38	5.79	9.32
eil33	443	21	15.01	21.50
bayg29	137	10	3.62	5.53
bays29	215	31	4.39	6.91
ulysses16	59	9	0.60	1.46
ulysses22	735	16	12.15	19.52
gr17	1	0	0.02	0.30
gr21	29	9	0.87	1.89
gr24	271	17	5.26	8.69
fri26	499	16	6.65	11.56
swiss42	973	18	46.79	61.75
att48	3635	26	126.53	162.24
gr48	7581	30	345.38	420.89
hk48	5771	36	253.17	312.33
eil51	3141	19	186.49	224.84
Total (1 LP)	31783		1182.40	1509.88
Total (2 LPs)	31011		1103.61	748.21
Total (4 LPs)	28777		1093.44	380.41
Total (8 LPs)	26947		1018.00	214.49

Running times are for the IBM Scalable POWERparallel System II consisting of a network of IBM RISC/System 6000s computers connected by a high speed switch located at Cornell University's Theory Center. The LP engine used is CPLEX.

Conclusions

- COMPSys is an easy and effective tool for implementing parallel branch and cut.
- The Decomposition Algorithm is very effective at finding capacity constraints not discovered by other heuristics.
  - The algorithm was successful at finding a decomposition approximately 65% of the time.
  - Additional constraints were imposed 25% of the time.
- The current implementation is much too slow to be useful.
- Better cut generation over more classes of cuts is needed to solve bigger instances of the VRP.
- Investigation of new branching strategies, including branching on cuts, could improve running times.
- The two-phase algorithm, along with repricing and tree trimming, reduces running times significantly over standard approaches.
- Strong Branching is also very effective.

Future Work

- Vehicle Routing Problem
  - Improvements to the Decomposition Algorithm
  - More separation!
  - Branching on cuts
- Applications Under Development
  - Márta Esö, Airline Crew Scheduling Problem
  - Laci Ladányi, Traveling Salesman Problem
  - Leonid Kopman, Vehicle Routing Problem
- Improvements to COMPSys
  - Support for multiple search trees
  - Better load balancing – dynamic allocation of cut generators
  - Upgrade communication protocol
  - Serial and shared-memory versions (using OpenMP)