COMPSys:

Combinatorial
Optimization
Multi-
Processing
SyStem

Presented by:

Ted Ralphs

Joint work with:

Márta Esö

Laci Ladányi

Les Trotter

Outline of Talk

- Introduction to Parallel Branch and Cut
- Description of COMPSys
- Introduction to the Vehicle Routing Problem
- Solving the VRP Using COMPSys
- 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 $c \in \mathbf{Z^E}$, 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 $\mathcal{C} \rightarrow \mathcal{C} \setminus \mathcal{R}^i$.
   - 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 $\overline{s}$ 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 $\mathcal{C}$.

Parallelizing Branch and Cut

There are several obvious ways to parallelize branch and cut:

- Process multiple subproblems in parallel.

  Advantage: Faster enumeration.

  Disadvantage: Can enlarge the search tree.

- Within a single subproblem, solve LP relaxations and generate cuts in parallel.

  Advantage: LP reoptimized sooner and more often.

  Disadvantage: Cut generation can "lag behind."

- A further possibility is to process multiple search trees in parallel.

  Advantage: Trees share upper bounds, cuts, and can use different branching rules, etc.

  Disadvantage: Wasted computation.

COMPSys

COMPSys is a **generic** framework for implementing parallel branch and cut algorithms.

Main Features

- The shell is implemented as a "black box" that completely separates problem-specific subroutines from the rest of the algorithm.
- Only a few user-supplied subroutines are required to develop a state-of-the-art branch and cut algorithm for any problem.
- User supplies:
  - Initial upper bounding subroutines,
  - Separation subroutines,
  - The initial LP relaxation, and
  - Other optional subroutines.
- COMPSys takes care of:
  - Bookkeeping, managing search tree, cut storage, etc.,
  - Interfacing with LP solver, and
  - Process communication.

Implementation

In COMPSys, there are six module types that work together to perform the algorithm:

Master Maintains problem instance data and keeps track of the best solution found so far.

Tree Manager Controls overall execution by tracking growth of the tree and dispatching subproblems to the LP solvers.

LP Solvers Perform processing and branching operations on subproblems.

Cut Generators Take LP solutions and generate valid inequalities.

Cut Pools Act as auxiliary cut generators by maintaining a list of the "most effective" inequalities found so far.

GUI Allows graphical display of fractional and integer solutions as well as real-time addition of cuts by user.

The Processes of Parallel Branch and

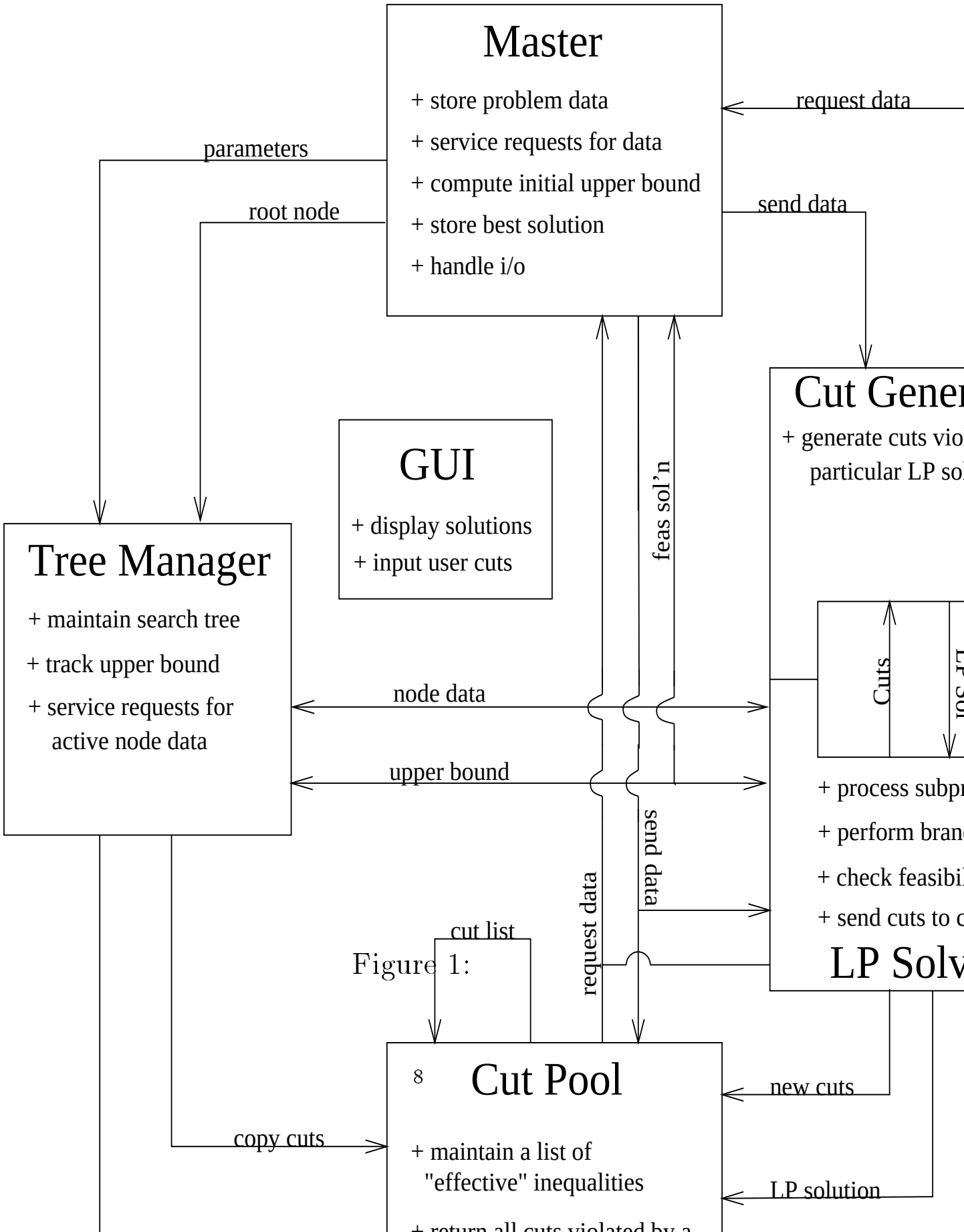


Figure 1:

Current Ports

- Platforms
  - Pentium PC running Linux
  - Sun Sparc running Solaris or SunOS
  - IBM RS6000 running AIX
- Applications
  - Vehicle Routing Problem
  - Traveling Salesman Problem
  - Airline Crew Scheduling Problem (Set Partitioning)
- LP Solvers
  - XMP
  - CPLEX

Process communication is currently accomplished using the Parallel Virtual Machine message-passing protocol.

COMPSys Features: Handling of Cuts

- Currently, each LP has its own dedicated cut generator.
- Cuts are initially received by the LP solver
  - Each LP solver maintains a small "local cut pool."
  - A limited number of cuts are added to the LP in each iteration. This prevents "saturation."
  - Cuts are only sent to the global cut pool if they prove effective locally.
- One or more cut pools maintain a list of the most "effective" cuts found so far.
  - Each pool services a subtree – pools are dynamically allocated.
  - The use of multiple pools allows locally valid cuts to be generated if desired.
  - With multiple cut pools, pools are smaller and contain cuts that were generated "closer" in the tree $\Rightarrow$ more likely to be violated.
  - The size of each pool is controlled through the purging of "ineffective" cuts.

COMPSys Features: Handling of Variables

- Reduced cost fixing is standard and logical fixing is available with a user-supplied subroutine.
- Column generation is supported.
  - The user supplies the base set of variables and a column generation subroutine.
  - Column generation can be done before branching or before pruning.
- A unique two-phase algorithm is also available.
  - The algorithm is run to completion using the base set of variables before generating additional columns.
  - Using the upper bound and cuts from the first phase, all variables are priced out in the root node and are then propogated down into the leaves as required.
  - The tree is also trimmed by aggregating children back into their parent as appropriate.
- General upper and lower bounds are supported.

COMPSys Features: Branching

- Can branch on cuts or variables.
- Multi-way branching is supported.
  - Any number of children is allowed.
  - Branch on several left hand values for a constraint.
- Strong branching is also available.
  - Select several branching candidates.
  - "Presolve" each candidate.
  - Choose the "best" for branching.

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 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:

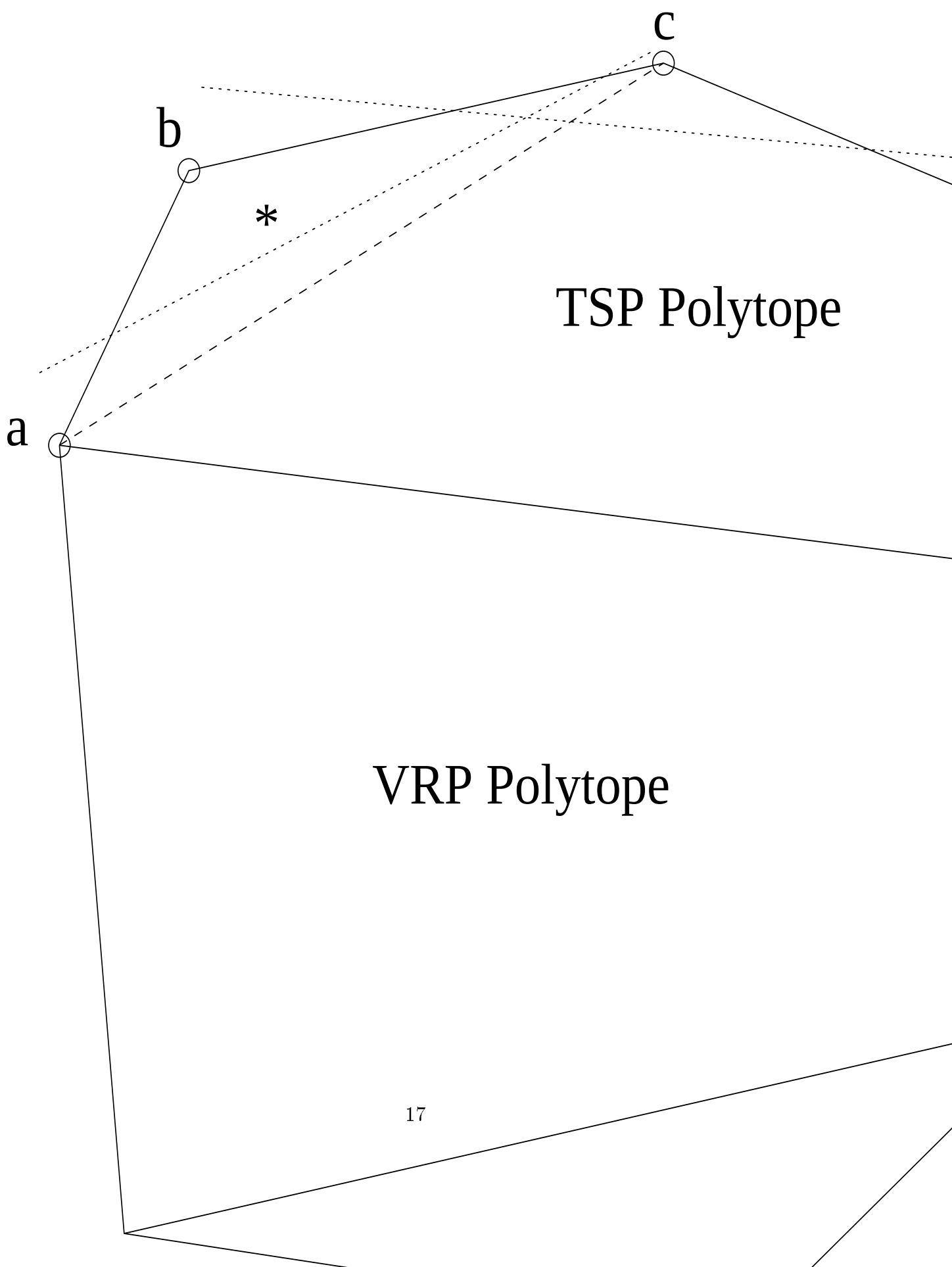
- Logical fixing, and
- Column generation.

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 *Extended Shrinking Heuristic* based on the N-I min cut algorithm.
  - Finally, apply the *Decomposition Algorithm.*

Combinatorial Problems with Side Constraints

- 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.
- 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 called a *subproblem* 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
*
TSP Polytope
a
VRP Polytope

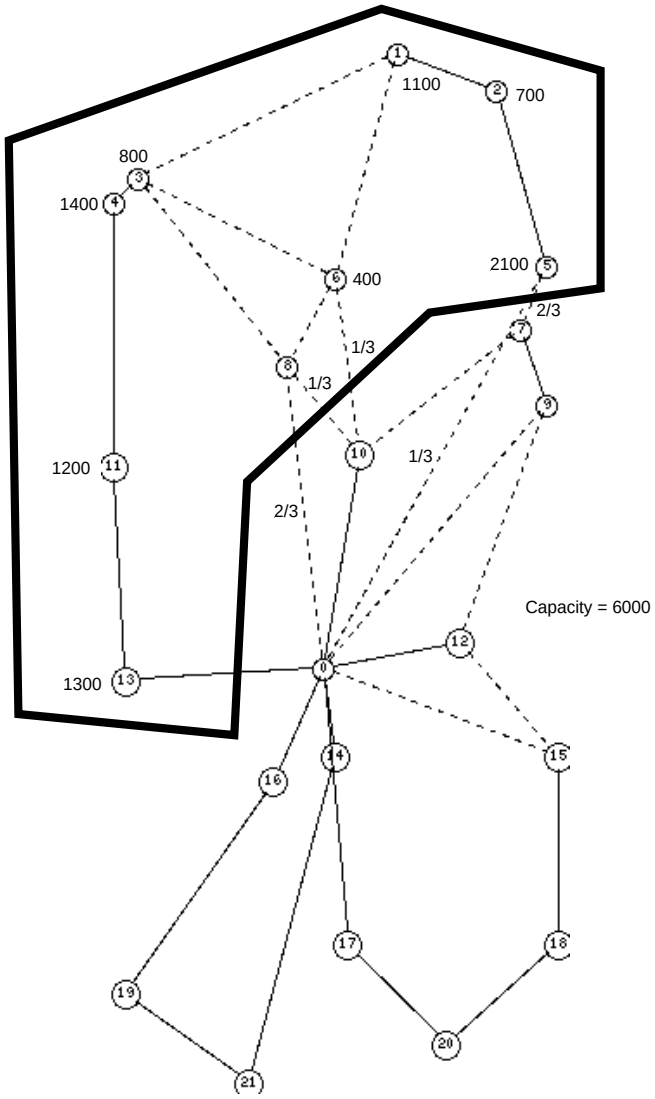
The Decomposition Algorithm

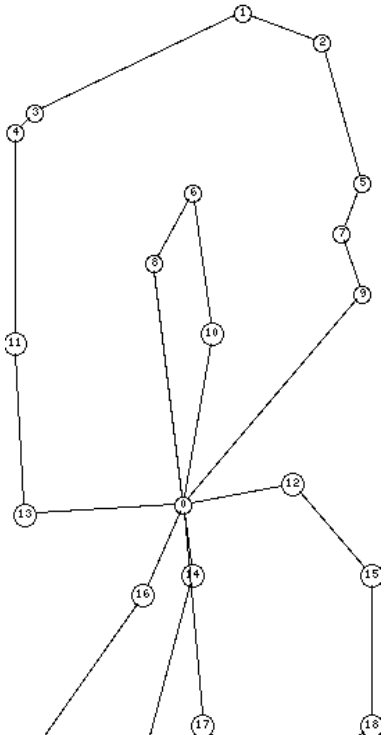
- Attempt to decompose the fractional solution $\hat{x}$ into a convex combination of TSP tours on a slightly expanded graph by solving the *Decomposition LP* defined by

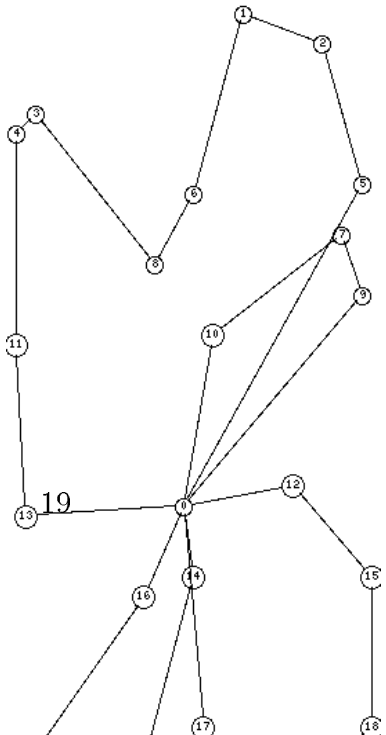
$$\max\{\mathbf{0}^\top \lambda : T\lambda = \hat{x},\ \mathbf{1}^\top \lambda = 1,\ \lambda \geq 0\}$$

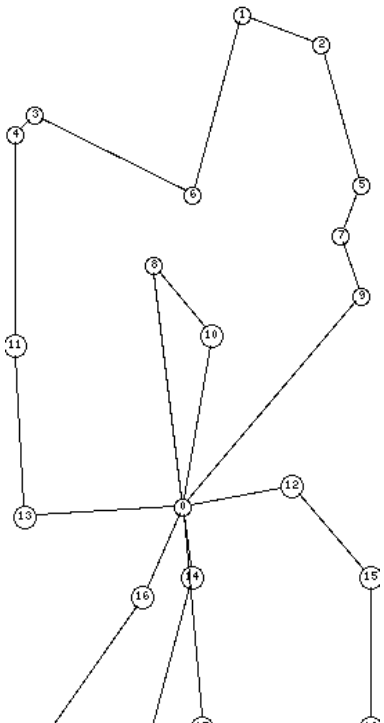
  where T is a matrix whose columns are the extreme points of the TSP polytope. This is related to Dantzig-Wolfe decomposition.

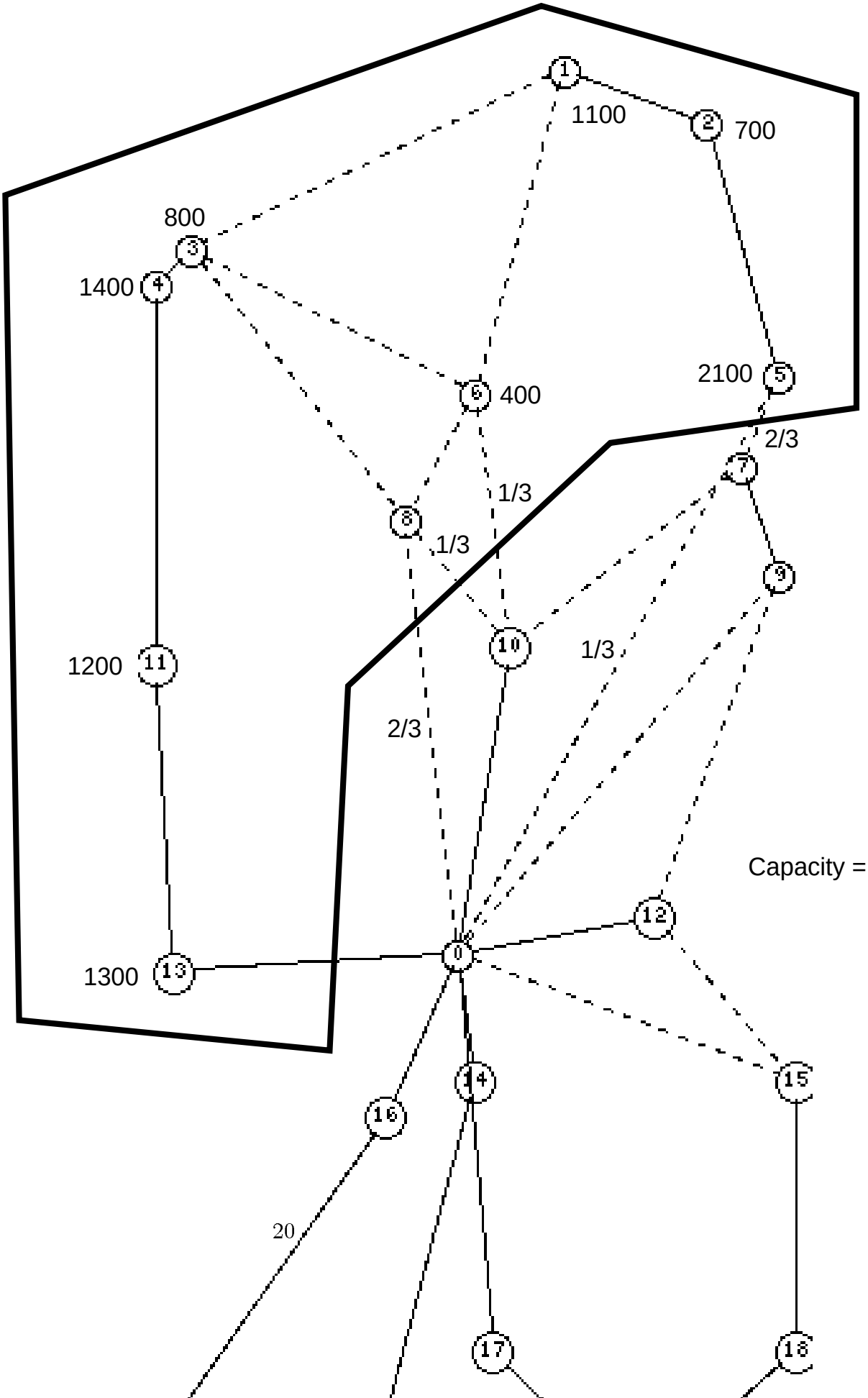
- If successful, then separate over the tours in the linear combination to obtain a set of possibly violated inequalities. Otherwise, we obtain an inequality which separates $\hat{x}$ from the TSP polytope.


1
1100
2
700
800
3
1400
4
2100
5
6
400
2/3
7
1/3
8
1/3
9
10
1/3
1200
11
2/3
Capacity = 6000
12
1300
13
0
14
15
16
17
18
19
20
21








1
1100
2
700
800
3
1400
4
2100
5
6
400
2/3
7
1/3
8
1/3
9
10
1/3
1200
11
2/3
Capacity =
12
1300
13
0
14
15
16
17
18

The Key Results

Theorem 1 *Let $\hat{x} = T\hat{\lambda}$ be the optimal solution following an iteration of the revised Dantzig-Wolfe algorithm. 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 $p \in \mathbf{Z}_+,\ 2 \leq p \leq |E|;\ \lambda_1, \ldots, \lambda_p,\ \sum_{i=1}^{p} \lambda_i = 1,\ \lambda_i > 0$ for $i = 1, \ldots, p$; and $S_1, \ldots, S_p,\ S_i \in \mathcal{F}$ 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 $y\hat{x} < \gamma$.*

Solving the Decomposition LP

We solve the decomposition LP by column generation. We call this method the *Revised Dantzig-Wolfe Algorithm.*

Step 1 Generate a matrix T' containing a small subset of "promising" 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}. Solve CP with cost vector c defined by

$$c_i = \begin{cases} M & \text{if } \hat{x}_i = 0; \\ -M & \text{if } \hat{x}_i = 1; \\ \mu_i & \text{otherwise} \end{cases} \tag{1}$$

for $1 \leq i \leq n$ where M is sufficiently large. Let t be the incidence vector of the result. If $\mu t < -\mu_{n+1}$, then t is a column eligible to enter the basis. Add t to T' and go to 1.1. Otherwise, $(\mu, -\mu_{n+1})$ is a valid inequality for CPP which is violated by $\hat{x}$.

Current Implementation of the Decomposition Algorithm

- Decomposition is also implemented as a Black Box.
- The user supplies
  - the column generation subroutine, and
  - the subroutines for generating side constraints from solution vectors.
- A separate pool of potential columns is maintained in exactly the same way as the cut pool for the LP.
- Currently, for the VRP, columns are generated by "brute force" enumeration (very inefficient).

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
  - New separation techniques
  - 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
  - Dynamic allocation of cut generators
  - Support for multiple search trees
  - Upgrade communication protocol to MPI
  - Support for OSL