# Capacitated Vehicle Routing and Some Related Problems

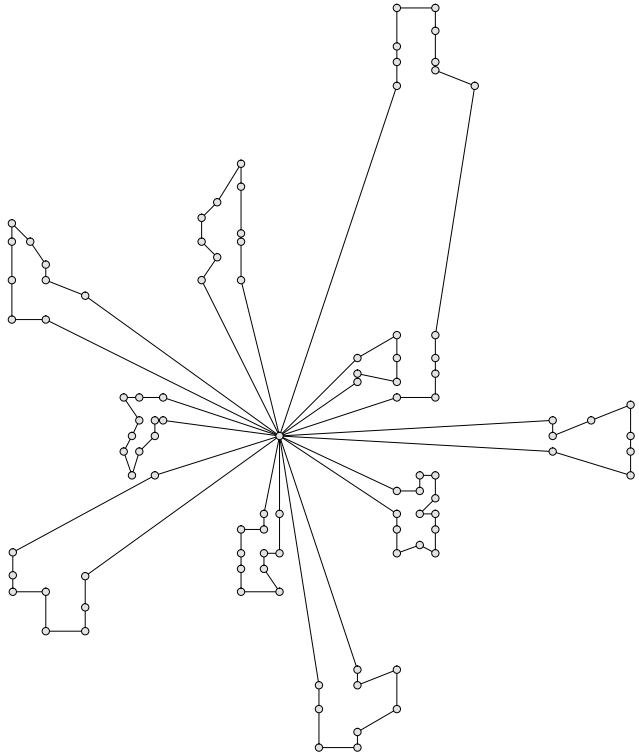
Ted Ralphs

Industrial and Systems Engineering
Lehigh University

Computer Science and Engineering, March 24, 2005

# Outline of Talk

- Introduction to Integer Programming
  - Modeling
  - Solutions Techniques
- Introduction to Vehicle Routing
- Modeling Issues
- Related Problems
- Complexity
- Valid Inequalities
- Implementation
- Computational Issues and Results
- Future Directions



# Mathematical Programming Models

- What does *mathematical programming* mean?
- Programming here means "planning."
- Literally, these are "mathematical models for planning."
- Also called *optimization models*.
- Essential elements
  - Decision variables
  - Constraints
  - Objective Function
  - Parameters and Data

# Forming a Mathematical Programming Model

- The general form of a *mathematical programming model* is:

$$\begin{array}{lll} min & f(x) & \\ s.t. & g_i(x) \left\{ \begin{array}{l} \leq \\ = \\ \geq \end{array} \right\} & b_i \\ & x \quad \in \quad X & \end{array}$$

  $X \subseteq \mathbb{R}^n$ is an (implicitly defined) set that may be discrete.

- A *mathematical programming problem* is a problem that can be expressed using a mathematical programming model (called the *formulation*).

- A single mathematical programming problem can be represented using many different formulations (important).

# Types of Mathematical Programming Models

- The type of mathematical programming model is determined mainly by
  - The form of the objective and the constraints.
  - The form of the set $X$.
- In this talk, we consider linear models.
  - The objective function is linear.
  - The constraints are linear.
  - Linear models are specified by cost vector $c \in \mathbb{R}^n$, constraint matrix $A \in \mathbb{R}^{m \times n}$, and right-hand side vector $b \in \mathbb{R}^m$ and have the form

$$\begin{aligned} min\, & c^T x \\ s.t. \quad & Ax \geq b \\ & x \in X \end{aligned}$$

# Linear Models

- Generally speaking, linear models are easier to solve than more general types of models.
- If $X = \mathbb{R}^n$, the model is called a *linear program* (LP).
- Linear programming models can be solved effectively.
- If some of the variables in the model are required to take on integer values, the model is called a *mixed integer linear programs* (MILPs).
- MILPs can be extremely difficult to solve in practice.

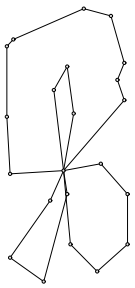
# Modeling with Integer Variables

- Why do we need integer variables?
- If a variable represents the quantity of a physical resource that only comes in discrete units, then it must be assigned an integer value.
- We can use *0-1 (binary) variables* for a variety of other purposes.
  - Modeling yes/no decisions.
  - Enforcing disjunctions.
  - Enforcing logical conditions.
  - Modeling fixed costs.
  - Modeling piecewise linear functions.
- The simplest form of ILP is a *combinatorial optimization problem* (COP), where all variables are binary.

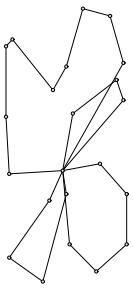
# Combinatorial Optimization

- A *combinatorial optimization problem* $CP = (E, \mathcal{F})$ consists of
  - A *ground set* E,
  - A set $\mathcal{F} \subseteq 2^E$ of *feasible solutions*, and
  - A *cost function* $c \in \mathbb{Z}^E$ (optional).
- The *cost* of $S \in \mathcal{F}$ is $c(S) = \sum_{e \in S} c_e$.
- A *subproblem* is defined by $\mathcal{S} \subseteq \mathcal{F}$.

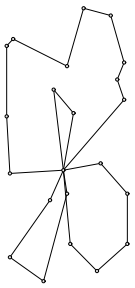
- Problem: Find a least cost member of $\mathcal{F}$.



Cost 1100



Cost 1105



Cost 1107

# Example: Perfect Matching Problem

- We are given a set $N$ of $n$ people that need to paired in teams of two.
- Let $c_{ij}$ represent the "cost" of the team formed by persons $i$ and $j$.
- We wish to minimize the overall cost of the pairings.
- The nodes represent the people and the edges represent pairings.
- We have $x_{ij} = 1$ if $i$ and $j$ are matched, $x_{ij} = 0$ otherwise.
- To simplify the presentation, we assume that $x_{ij} = 0$ if $i \geq j$.

$$min \sum_{\{i,j\} \in N \times N} c_{ij} x_{ij}$$

$$s.t. \quad \sum_{j \in N} x_{ij} = 1, \qquad \forall i \in N$$

$$x_{ij} \in \{0,1\}, \ \forall \{i,j\} \in N \times N, i < j.$$

# Fixed-charge Problems

- In many instances, there is a fixed cost and a variable cost associated with a particular decision.
- Example: Fixed-charge Network Flow Problem (FCNFP)
  - We are given a directed graph $G = (N, A)$ and a demand/supply at each node.
  - There is a fixed cost $c_{ij}$ associated with building arc $(i, j)$.
  - There is also a variable cost $d_{ij}$ for each unit of flow along arc $(i, j)$.
  - We want to minimize the sum of these two costs while meeting demand.
- Example: Cable-Trench Problem (CTP)
  - A FCNFP with only one supply node (the communications hub).
  - All other nodes must be connected to the hub by a cable.
  - The fixed cost is the cost of digging the trenches.
  - The variable cost is the cost of laying the cable.

# How Do We Solve These Problems?

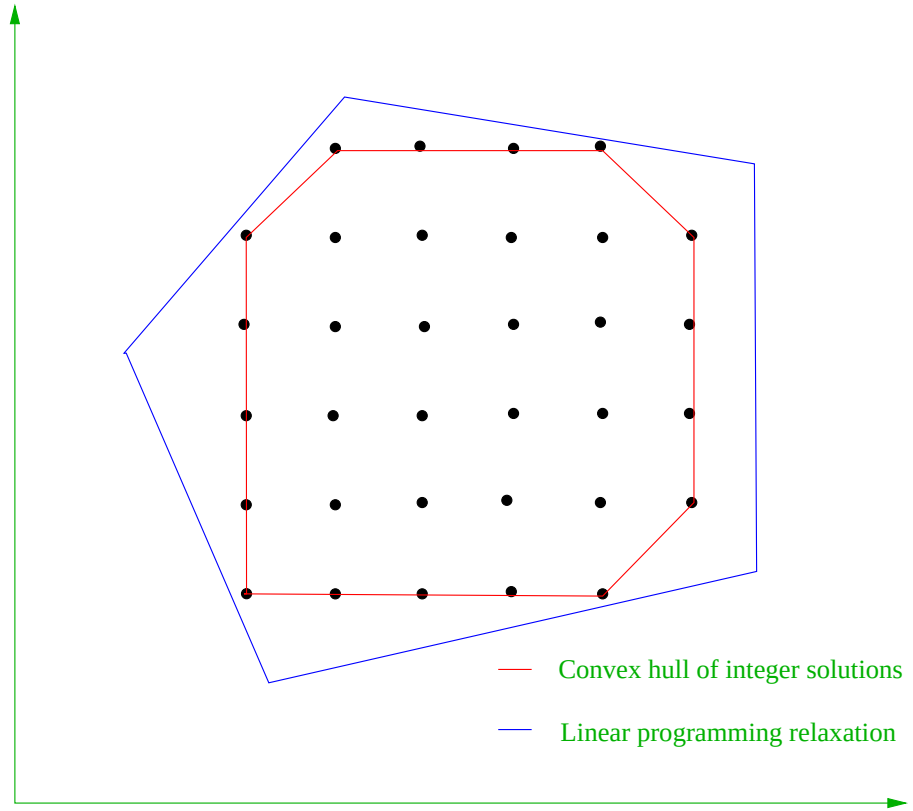
- *Implicit enumeration* techniques try to enumerate the solution space in an intelligent way.
- The most common algorithm of this type is *branch and bound*.
- Suppose $F$ is the set of feasible solutions for some MILP and we wish to solve $\min_{x \in F} c^T x$.
- Consider a partition of $F$ into subsets $F_1, \ldots F_k$. Then

$$\min_{x \in F} c^T x = \min_{1 \leq i \leq k} \{ \min_{x \in F_i} c^T x \}$$

.

- Idea: If we can't solve the original problem directly, we might be able to solve the smaller *subproblems* recursively.
- Dividing the original problem into subproblems is called *branching*.
- Taken to the extreme, this scheme is equivalent to complete enumeration.
- We avoid complete enumeration primarily by deriving bounds on the value of an optimal solution to each subproblem.

# The Geometry of Integer Programming



# Bounding

- A *relaxation* of an ILP is an auxiliary mathematical program for which
  - the feasible region contains the feasible region for the original ILP, and
  - the objective function value of each solution to the original ILP is not increased.
- Types of Relaxations
  - Continuous relaxations
    - Most common continuous relaxation is the *LP relaxation*.
    - Obtained by dropping some or all of the integrality constraints.
    - Easy to solve.
    - Initial bounds weak, but can be strengthened with valid inequalities.
    - Other relaxations are possible using semi-definite programming, for instance.
  - Combinatorial relaxations
    - Obtained by dropping some of the linear constraints.
    - Violation of these constraints can then penalized in the objective function (*Lagrangian relaxation*)
    - Bound strength depends on what constraints are dropped.

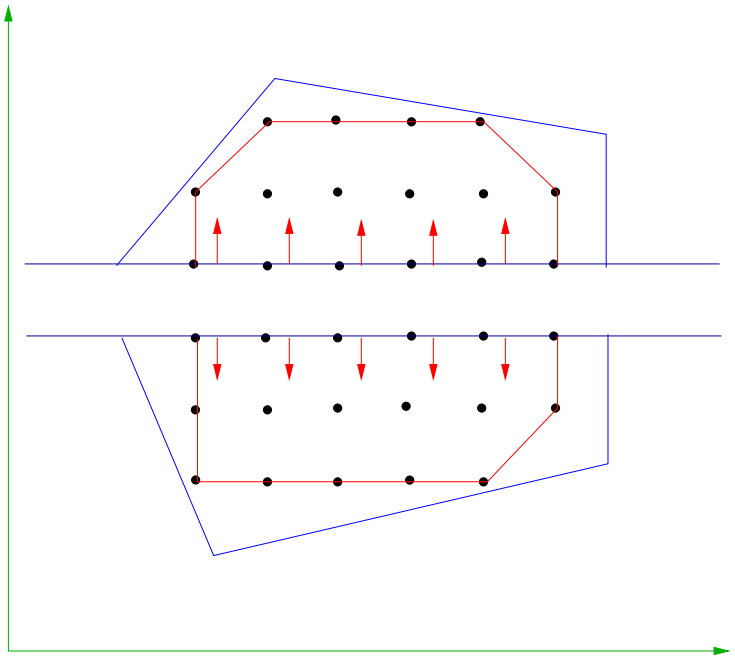
# Branch and Bound Algorithm

- We maintain a queue of *active* subproblems initially containing just the *root subproblem*.
- We choose a subproblem from the queue and solve a relaxation of it to obtain a *bound*.
- The result is one of the following:
  1. The relaxation is infeasible $\Rightarrow$ subproblem is infeasible.
  2. We obtain a feasible solution for the MILP $\Rightarrow$ subproblem solved (new upper bound??).
  3. We obtain an optimal solution to the relaxation that is not feasible for the MILP $\Rightarrow$ lower bound.
- In the first two cases, we are finished.
- In the third case, we compare the lower bound to the global upper bound.
  - If it exceeds the upper bound, we discard the subproblem.
  - If not, we branch and add the resulting subproblems to the queue.

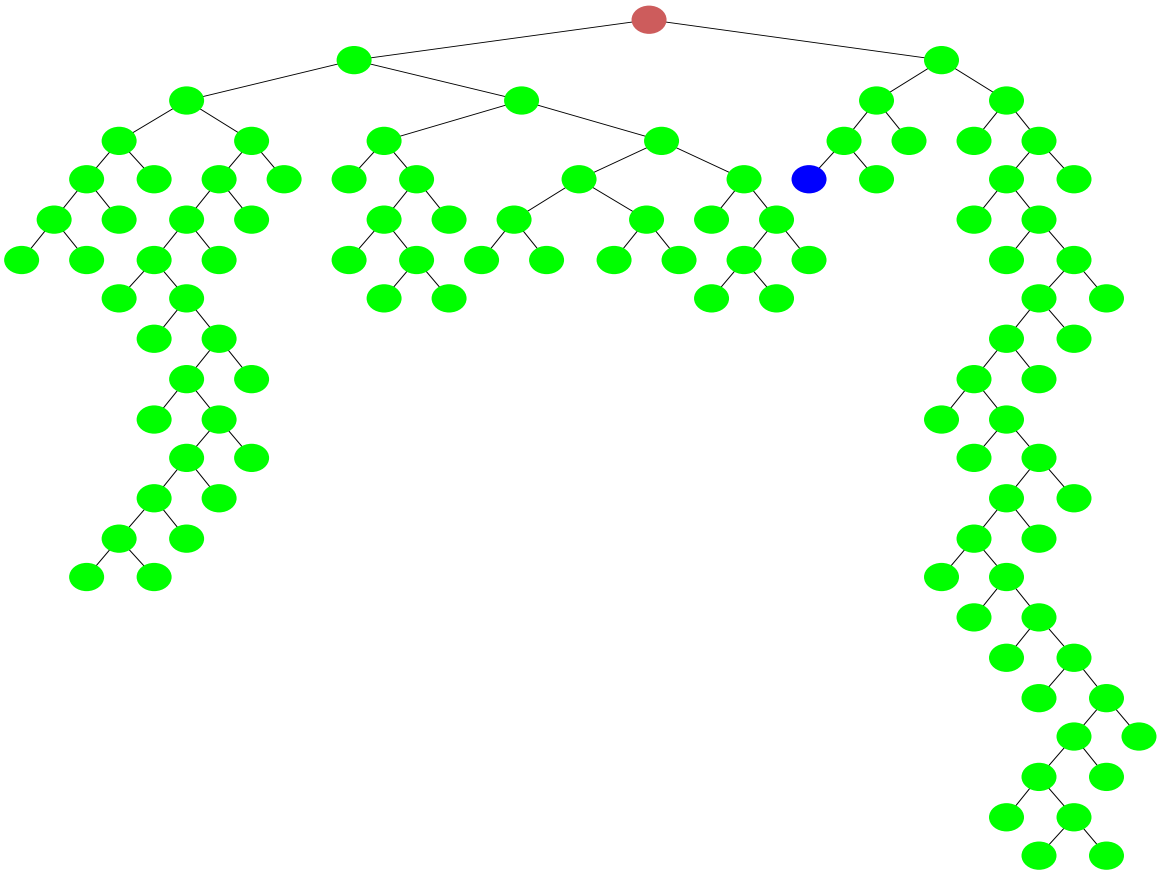
# Branching

Branching involves partitioning the feasible region with hyperplanes such that:

- All optimal solutions are in one of the members of the partition.
- The solution to the current relaxation is not in any of the members of the partition.



# Branch and Bound Tree



# 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 $\sigma_i$ of $R_i \cup \{0\}$ specifying the order of the customers on route $i$.

# Classical Formulation for the VRP

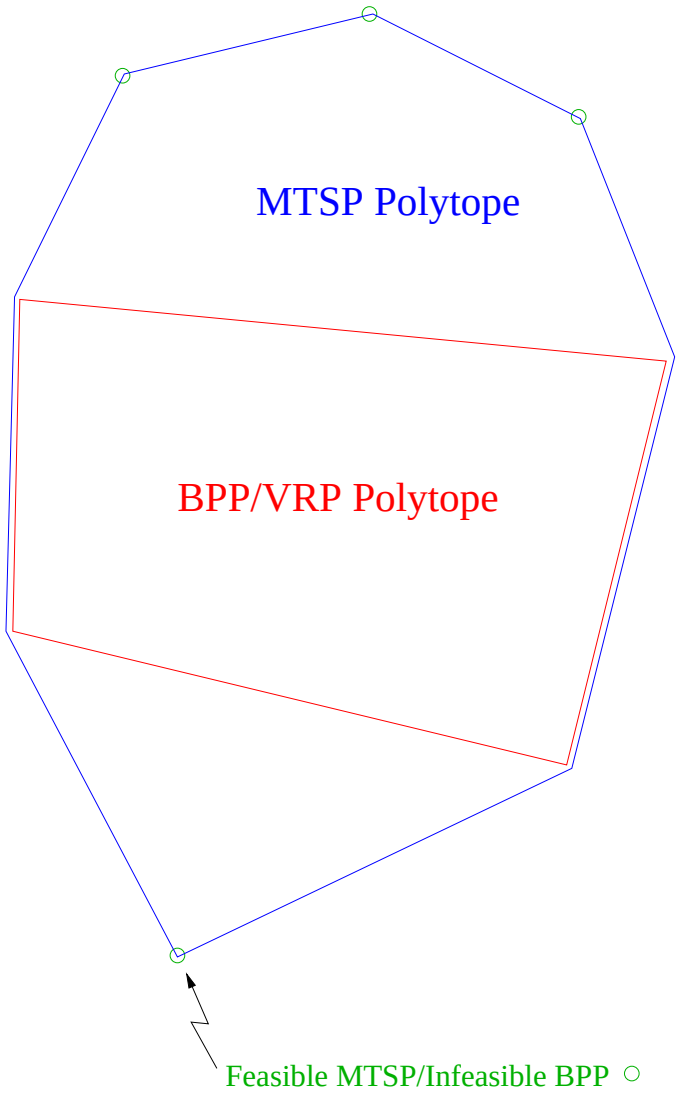
IP Formulation:

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

$b(S) =$ 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 the Multiple Traveling Salesman Problem.

Alternatively, if the edge costs are all zero, then we have the Bin Packing Problem.


MTSP Polytope
BPP/VRP Polytope
Feasible MTSP/Infeasible BPP

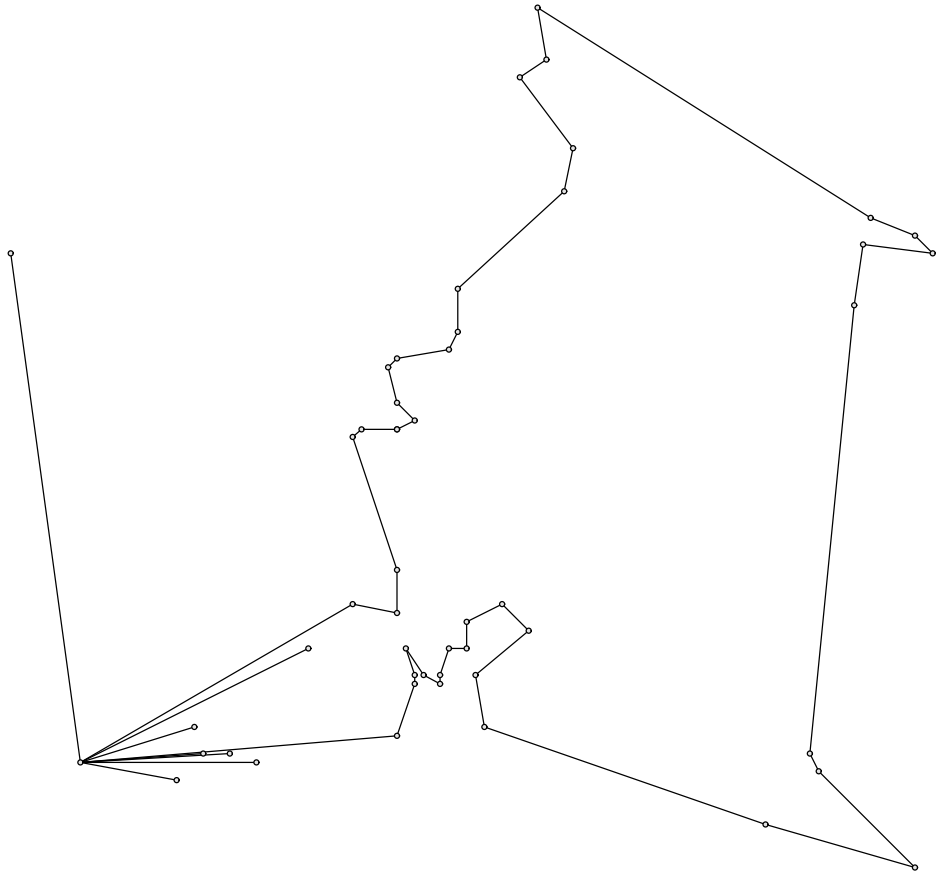
# How Hard is the VRP?

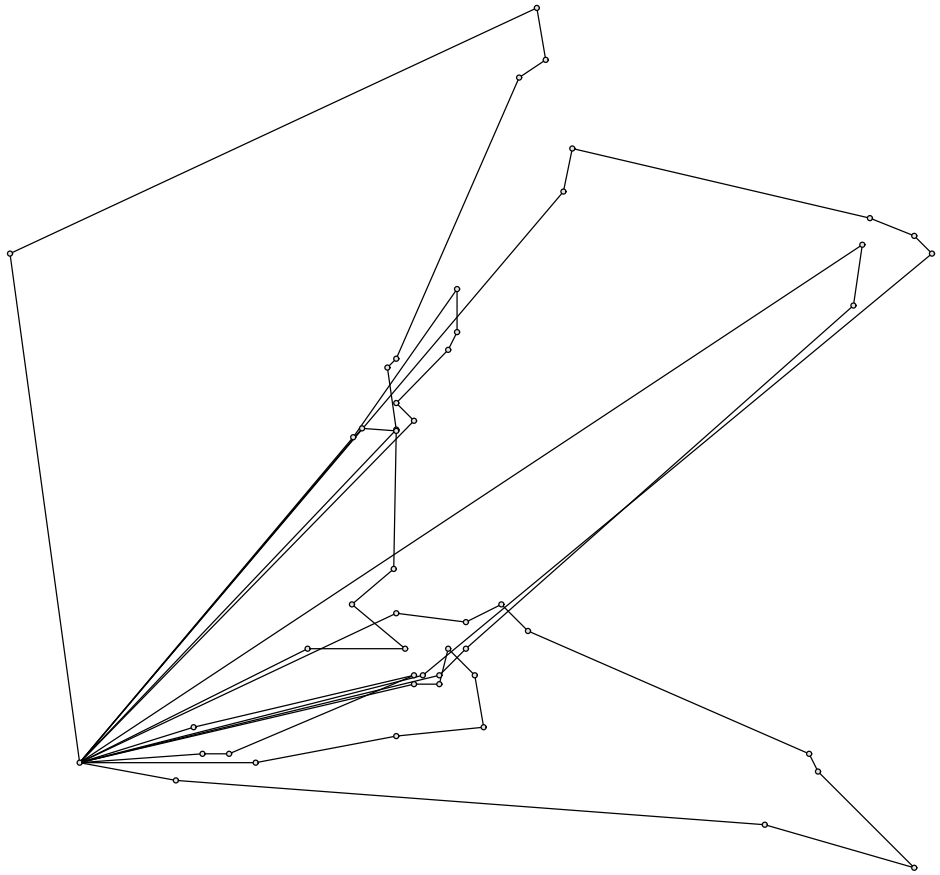
- Test Set
  - TSPLIB/VRPLIB
  - Augerat's repository
  - Available at BranchAndCut.org/VRP
- Largest VRP instance solved: F-n135-k7
- Largest TSP instance solved: d15112
- Smallest VRP instance unsolved (as of '01): B-n50-k8
- Time to solve B-n50-k8 as an MTSP: .1 sec
- Why the gap?

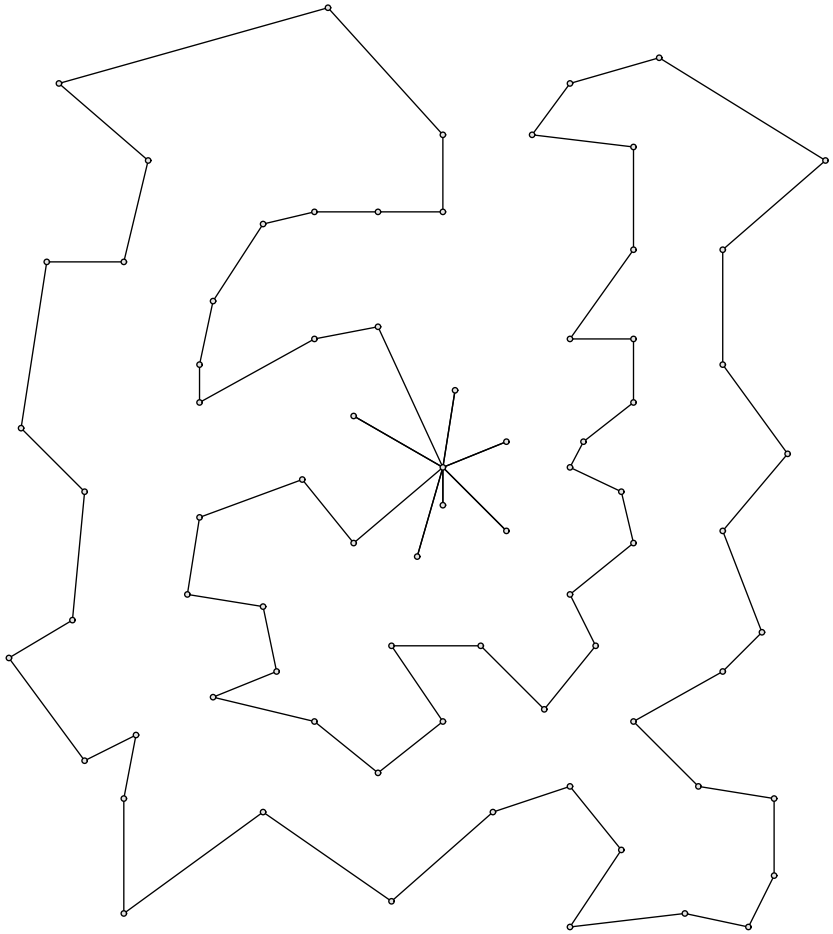
# Standard Approach

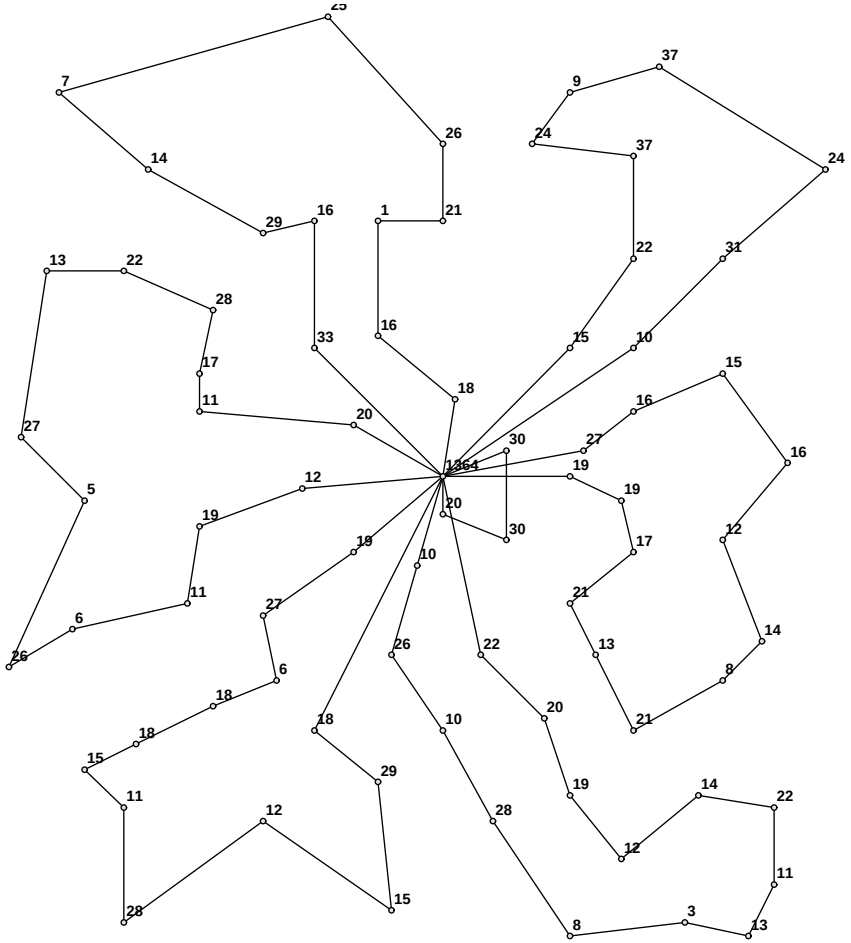
- Standard approaches treat the VRP in much the same way as the TSP.
  - Most known valid inequalities are generalizations from the TSP.
  - Branching rules are also generalizations from the TSP.
- However, the TSP does not seem to be the right template.
- It is the packing, not the routing that makes the problem difficult.


12
21
17
22
14
16
18
20
14
16
23
15
19
4
26
10
8
13
10
18
3
10
12
5
25
23
15
3
2
3
12
21
24
26
69
25
2
9
12



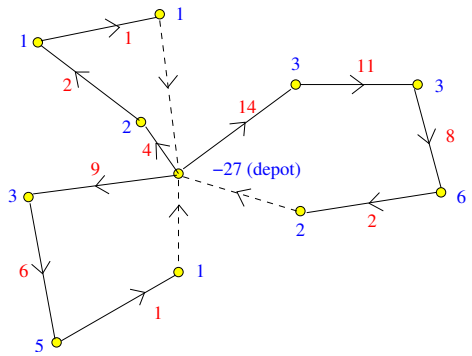





25
7
26
14
29
16
1
21
37
9
24
37
24
22
31
13
22
28
16
33
15
10
17
18
15
11
16
20
27
30
27
16
5
12
19
19
20
19
30
17
12
10
11
21
27
6
14
26
22
13
26
6
8
18
20
21
18
18
10
15
29
19
14
22
11
12
28
12
11
15
28
8
3
13

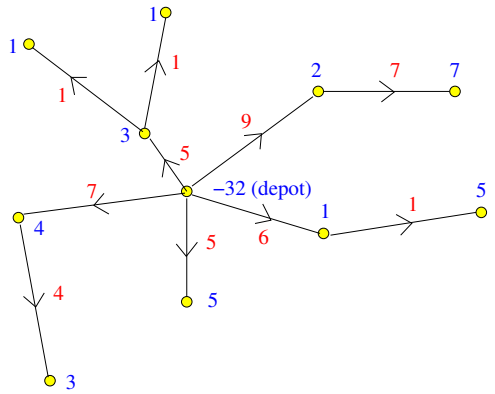
# What Makes the VRP Difficult?

- It is the intersection of two difficult problems.
  - Traveling Salesman Problem (Routing)
  - Bin Packing Problem (Packing)
- We don't have an effective, polynomially sized relaxation.
- Current approaches treat it as a routing problem.
- We know very little about the packing aspect.
- We need a different template.
- Idea: Consider flow-based formulations.



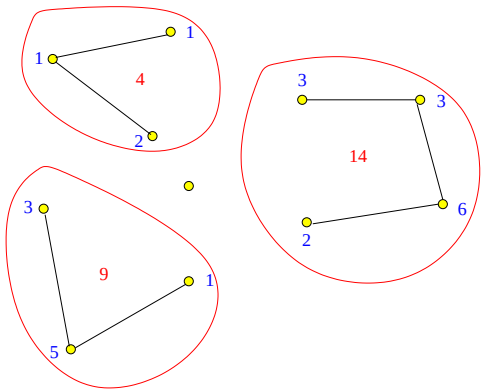
# Node Routing

- We are given an undirected graph $G = (V, E)$.
  - The nodes represent supply/demand points.
- We consider problems with one supply point (the *depot*).
- A *node routing* is a directed subgraph $G'$ of $G$ satisfying the following properties:
  - $G'$ is (weakly) connected.
  - The in-degree of each non-depot node is 1.



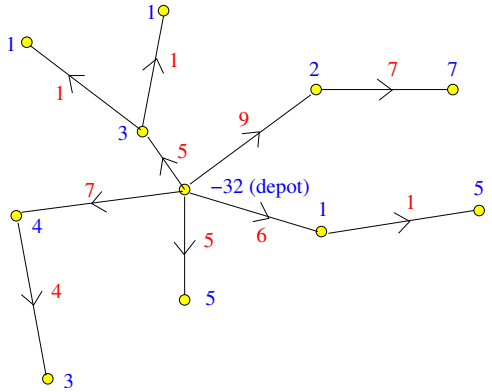
# Capacitated Node Routing

- A *capacitated node routing* is one in which the demand in each component of $G' \setminus \{0\}$ is $\leq C$.
- Feasible solutions are bin packings.
- This restriction is easily modeled using a flow-based formulation.
- With capacities, we can model the VRP and the Capacitated Spanning Tree Problem (CSTP).



# Optimal Node Routing

- Properties of a node routing.
  - It is a spanning arborescence plus (possibly) some edges returning to the depot.
  - There is a unique path from the depot to each demand point.
- We wish to construct a least cost routing.
  - Cost Measures
    - Lengths of all edges in $G'$.
    - Length of all paths from the depot.
    - Linear combination of these two.



# IP Formulation

IP formulation for this routing problem:

$$
\begin{aligned}
Min \quad & \sum_{(i,j)\in A} \gamma\, c_{ij} x_{ij} + \tau\, c_{ij} f_{ij} \\
s.t. \quad & x(\delta(V \setminus \{i\})) = 1 && \forall i \in V \setminus \{0\} \\
& f(\delta(V \setminus \{i\})) - f(\delta(\{i\})) = d_i && \forall i \in V \setminus \{0\} \\
& 0 \le f_{ij} \le C x_{ij} && \forall (i,j) \in A \\
& x_{ij} \in \{0,1\} && \forall (i,j) \in A
\end{aligned}
$$

where:

- $x_{ij}, x_{ji}$ (*fixed-charge variables*) indicate whether $\{i,j\}$ is in the routing *and its orientation*.
- $f_{ij}$ (*flow variable*) represents demand flow from $i$ to $j$.

# Complexity

- This node routing problem is NP-hard even in the uncapacitated case (fixed-charge network flow problem).
- Polynomially solvable special cases.
  - $\tau = 0 \Rightarrow$ Minimum Spanning Tree Problem.
  - $\gamma = 0 \Rightarrow$ Shortest Paths Tree Problem.
  - Note that demands are irrelevant.
- Other special cases.
  - $\tau = 0 \Rightarrow$ Capacitated Spanning Tree Problem.
  - $\tau, \gamma > 0 \Rightarrow$ Cable-Trench Problem.
  - $\tau = 0$ and $x(\delta(\{i\})) = 1 \Rightarrow$ Traveling Salesman Problem.
  - $\tau > 0$ and $x(\delta(\{i\})) = 1 \Rightarrow$ Variable Cost TSP.
  - $x(\delta(V \setminus \{0\})) = x(\delta(\{0\})) = k \Rightarrow$VRP.

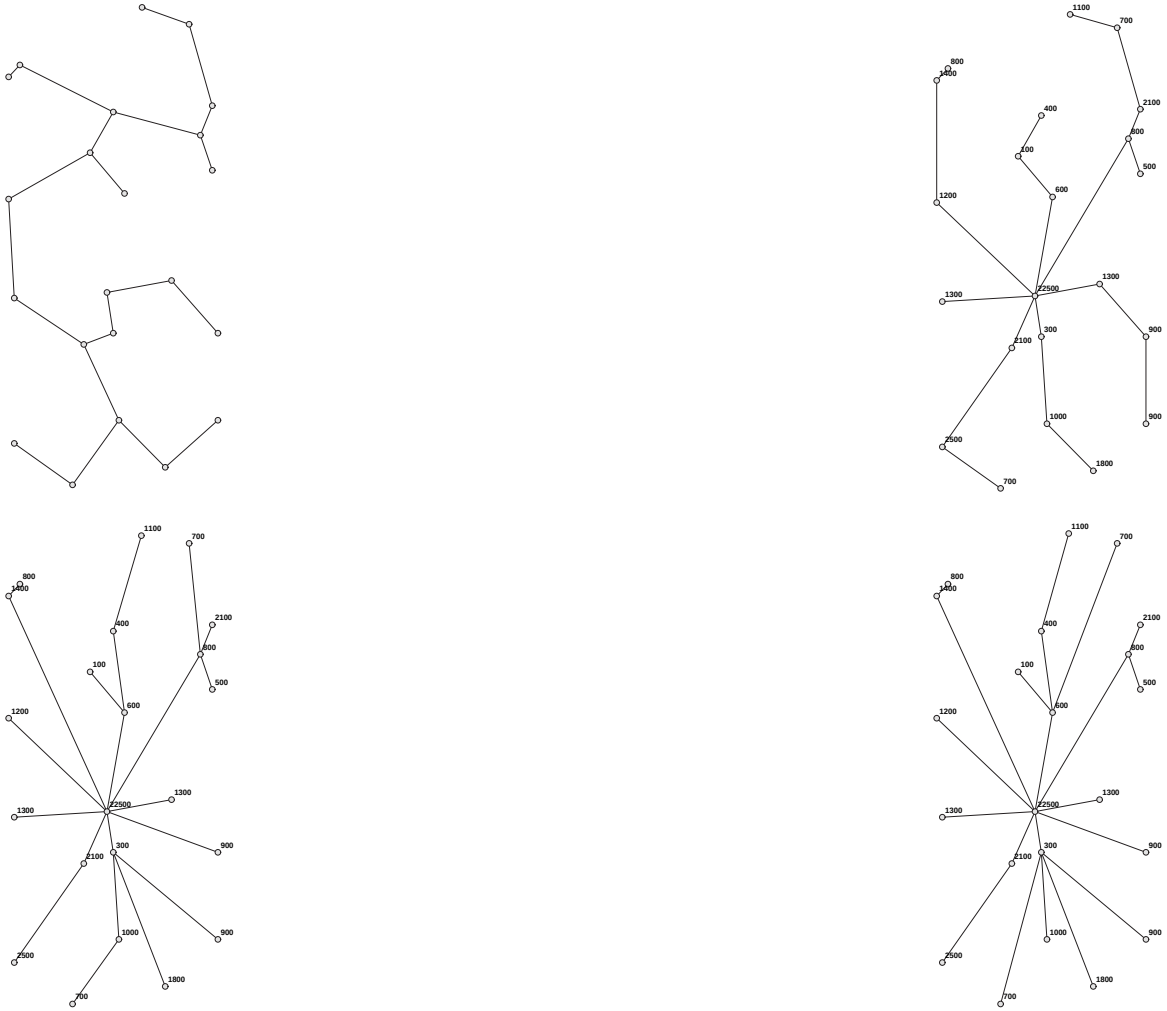



Figure 1: Optimal uncpacitated spanning trees with increasing $\tau/\gamma$ ratios


Figure 2: Uncapacitated vs. capacitated spanning trees ($\tau = 0$)

# Connection to Other Models

- There are connections to many well-studied models that may provide better templates.
- The basic model can be seen as an instance of the Fixed-charge Network Flow Problem.
- Removing the upper bounds on the fixed-charge variables yields the Capacitated Network Design Problem.
- We have already mentioned several other related combinatorial models.
- We are looking to make stronger connections among these varied areas of the literature.

# Valid Inequalities

- Note that any inequalities valid for the TSP, VRP, or CSTP have counterparts here.
- Many can be strengthened by taking advantage of the directed formulation.
- Fractional Capacity Constraints

$$\sum_{i \notin S,\, j \in S} x_{ij} \geq d(S)/C,\ 0 \notin S$$

- Multi-star Inequalities

$$\sum_{i \notin S,\, j \in S} x_{ij} \geq d(S)/C + \frac{\sum_{i \notin S,\, j \in S} x_{ji} d_i}{C},\ 0 \notin S$$

# Valid Inequalities

- Rounded Capacity Constraints

$$\sum_{i \notin S,\, j \in S} x_{ij} \geq \lceil d(S)/C \rceil$$

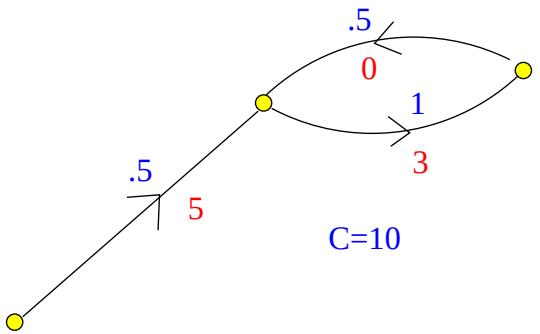
- Generalized, framed capacity constraints
- Combs, Hypo-tours, Clique Clusters
- Path-bin inequalities

# Flow Linking

- Note that only the edge variables are required to be integral.
- We use the flow variables to force integrality of the edge variables through *flow linking constraints*.
  - Flow Linking Constraints

$$f_{ij} \leq (C - d_i)x_{ij} \Leftrightarrow x_{ij} \geq \frac{f_{ij}}{C - d_i}$$

$$f_{ij} - \sum_{k \neq j} f_{jk} \leq x_{ij} d_j$$

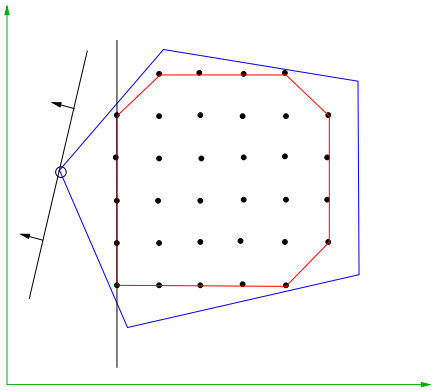



  - Edge Cuts

$$x_{ij} + x_{ji} \leq 1$$

# Separation

- The fractional capacity constraints and multi-star inequalities are automatically satisfied.
- Flow linking constraints and edge cuts can be included explicitly or separated in polynomial time.
- Separating rounded capacity constraints is NP-complete, but can be done effectively.
- Other classes must be separated using heuristic procedures.



# Solver Implementation

- The implementation uses SYMPHONY, a parallel framework for branch, cut, and price (relative of COIN/BCP).
- In SYMPHONY, 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`

# Computation: Formulation Issues

- The new formulation is polynomial in size and yields stronger relaxations initially, but there are drawbacks.
- For the VRP, the formulation creates symmetry.
- It also seems to make branching less effective.
- There is a related "undirected" formulation which uses one fixed-charge variable per edge.
  - This formulation is smaller and performs much better for the VRP.
  - For the CSTP and CTP, however, the undirected formulation is extremely weak.

# Computation: Comparison of Models

- So far, the presence of the flow variables does not seem to help.
- Capacitating the model does increase difficulty significantly.
- Consider relaxations of the VRP.
  - The TSP is very easy relative to the VRP.
  - The CSTP is not much easier than the VRP.
- Versions of these models with positive variable (flow) costs are extremely difficult.
  - Is this due to the upper bound or lower bound?
  - The flow linking constraints are important for these models.

| | $TSP$ | | $CSTP$ | | $VRP$ | |
|---|---|---|---|---|---|---|
| problem | Tree Size | CPU sec | Tree Size | CPU sec | Tree Size | CPU sec |
| eil13 | 1 | 0.00 | 13 | 0.09 | 1 | 0.00 |
| eil22 | 1 | 0.11 | 2 | 0.10 | 1 | 0.02 |
| eil33 | 1 | 0.02 | 69 | 3.97 | 2 | 0.44 |
| bayg29 | 1 | 0.12 | 1 | 0.04 | 4 | 0.32 |
| bays29 | 1 | 0.17 | 15 | 1.12 | 5 | 0.55 |
| ulysses16.tsp | 1 | 0.00 | 1 | 0.03 | 1 | 0.01 |
| ulysses22.tsp | 1 | 0.00 | 1 | 0.06 | 1 | 0.03 |
| gr17 | 1 | 0.01 | 5 | 0.05 | 1 | 0.01 |
| gr21 | 1 | 0.00 | 1 | 0.02 | 1 | 0.03 |
| gr24 | 1 | 0.02 | 5 | 0.27 | 4 | 0.40 |
| fri26 | 1 | 0.02 | 1 | 0.07 | 8 | 0.39 |
| swiss42 | 1 | 0.02 | 35 | 3.66 | 10 | 2.45 |
| att48 | 2 | 0.30 | 92 | 5.04 | 193 | 30.10 |
| gr48 | 2 | 1.38 | 1 | 0.07 | 16 | 4.17 |
| hk48 | 1 | 0.19 | 209 | 22.88 | 45 | 21.19 |
| eil51 | 1 | 0.16 | 77 | 15.11 | 11 | 10.79 |
| A − n32 − k5 | 1 | 0.02 | 1 | 0.07 | 2 | 0.20 |
| A − n33 − k5 | 3 | 0.81 | 3 | 0.21 | 7 | 0.90 |
| A − n34 − k5 | 6 | 2.06 | 4 | 0.40 | 9 | 2.63 |
| A − n36 − k5 | 1 | 0.03 | 52 | 5.17 | 51 | 7.95 |
| A − n37 − k5 | 1 | 0.03 | 5 | 0.22 | 11 | 0.97 |
| A − n38 − k5 | 1 | 0.10 | 1 | 0.13 | 111 | 21.80 |
| A − n39 − k5 | 1 | 0.30 | 11 | 0.99 | 480 | 310.92 |
| A − n44 − k6 | 3 | 1.72 | 586 | 84.08 | 1185 | 1525.78 |
| A − n45 − k6 | 2 | 0.27 | 47 | 6.19 | 133 | 145.59 |
| A − n46 − k7 | 1 | 1.25 | 3 | 0.20 | 2 | 1.95 |
| A − n48 − k7 | 2 | 2.01 | 775 | 507.41 | 1949 | 1620.57 |
| A − n53 − k7 | 1 | 0.62 | 115 | 19.99 | 619 | 881.05 |
| B − n31 − k5 | 1 | 0.01 | 3 | 0.63 | 1 | 0.08 |
| B − n38 − k6 | 1 | 0.04 | 5 | 0.56 | 14 | 1.73 |
| B − n39 − k5 | 1 | 0.03 | 188 | 9.67 | 1 | 0.05 |
| B − n41 − k6 | 1 | 0.08 | 216 | 18.96 | 20 | 2.89 |
| B − n43 − k6 | 1 | 0.09 | 1 | 0.36 | 138 | 34.92 |

# Conclusions and Future Directions

- We have established interesting connections to other well-studied models.
- The TSP does not seem to be the right template to follow.
- We have yet to take full advantage of the information provided by the flow variables.
- Better flow linking seems to be the key.
- We also need some new branching rules.
- The connection to the network design literature needs to be explored.
- We are also considering decomposition-based methods.