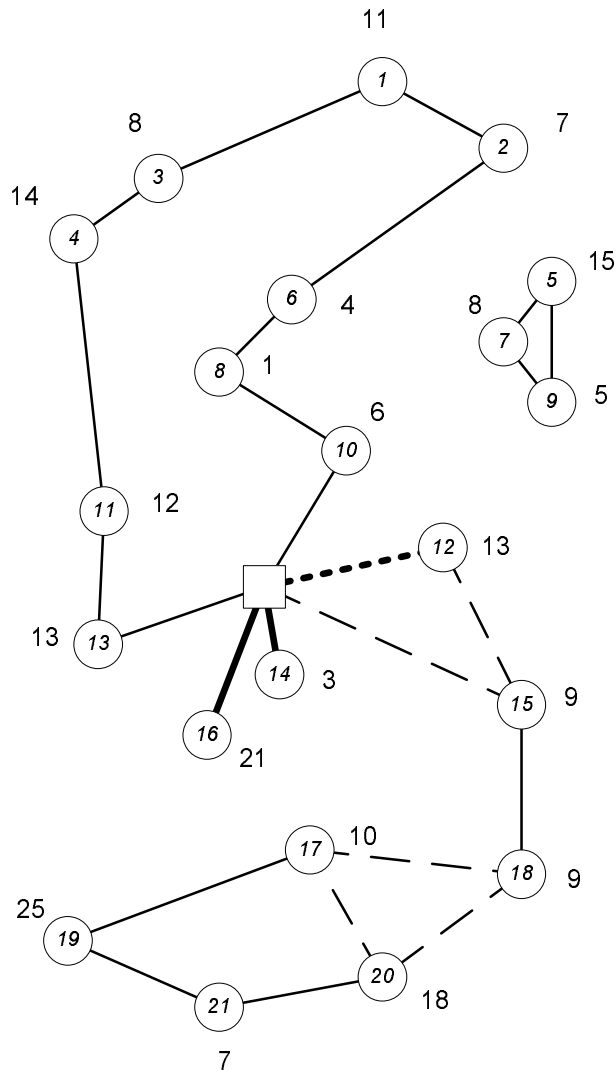


# Solving the Symmetric Capacitated Vehicle Routing Problem to optimality



Kombinatorisk optimering

5. marts 2004

Stefan Røpke

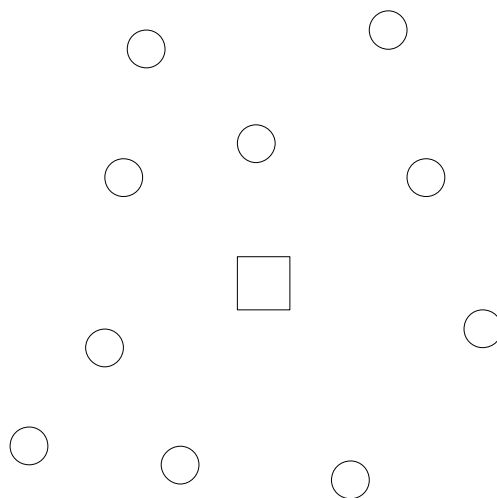
# Solving the Symmetric Capacitated Vehicle Routing Problem (SCVRP) to optimality

## Agenda

- Short introduction to the Symmetric Capacitated Vehicle Routing Problem (SCVRP)
- Some ILP Models for the SCVRP
- A basic branch and cut method for the SCVRP

# The Symmetric Capacitated Vehicle Routing Problem

- $n$  customers that each demand a certain amount of goods.
- $K$  identical vehicles, all located at one depot that supplies the goods to the customers. All vehicles should be used to serve the customers. Each vehicle has a certain capacity.
- A symmetric cost matrix that indicates the cost of traveling between two nodes in the graph. It is assumed that the cost matrix satisfies the triangle inequality
- Our task is to assign the customers to vehicles such that the capacity of the vehicles are obeyed and such that all customers are served. Furthermore we must construct routes for the vehicles. Our objective is to minimize the total cost of the routes (as given by the cost matrix). Each customer should be visited precisely once.



# Motivation for studying SCVRP

## The Good news:

- The real life applications of Vehicle Routing Problems should be obvious!
- Companies with large transportation needs are using software to make plans for their trucks already.
- Some examples: Arla, Carlsberg & Statoil.
- Plans designed by humans are typically far from the optimal solutions. Thus, large companies can potentially save millions of kroner by switching from manual to computer based planning.

## The Bad News:

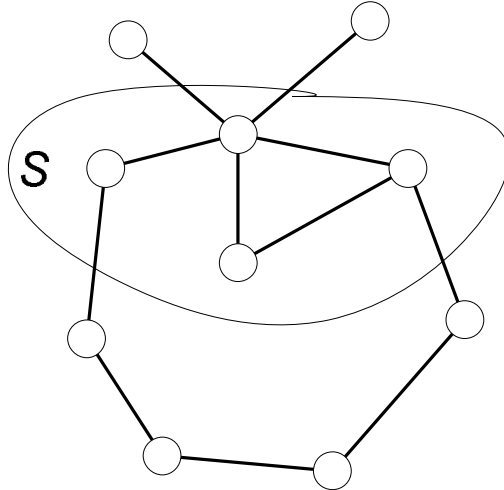
- SCVRP is NP-hard.
- Real life problems are (much) more complicated than the SCVRP model just presented.
- Real life problems are typically too large to be solved to optimality by the currently known methods.
- Consequently, software used for practical purposes are typically based on heuristics.

# Notation

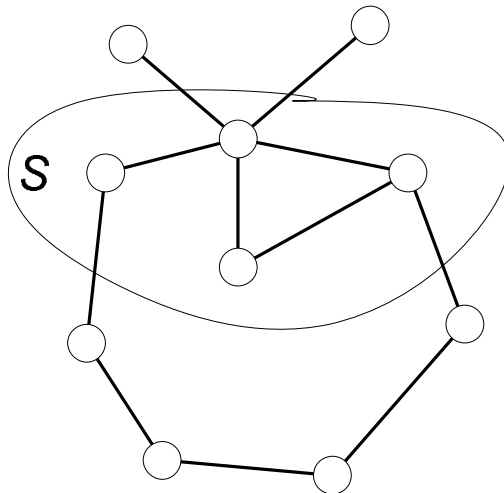
- $V = \{0, 1, \dots, n\}$  : The set of all nodes (customers), Node 0 is the depot.
- $V_0 = V \setminus \{0\}$
- $E = \{(i, j) : i, j \in V, j > i\}$  : The set of all edges.
- $c_{ij}$  : The elements of cost matrix that defines the cost of traveling between nodes (we assume that the cost matrix satisfies the triangle inequality).
- $G = (V, E)$  : The graph on which the problem is defined.
- $d_i$ : demand of customer  $i$ . It is assumed that  $\forall i \in V_0 : d_i > 0$
- $d(S) = \sum_{i \in S} d_i, S \subseteq V$
- $C$  : Capacity of a vehicle.
- $\delta(S), S \subseteq V$  : The set of edges with one endpoint in  $S$  and one endpoint in  $V \setminus S$ . ( $\delta(S)$  is called the *coboundary* of  $S$ )
- $E(S), S \subseteq V$  : The set of edges with both endpoints in  $S$ .
- $(S : T)$  : the set of edges with one endpoint in  $S$  and the other endpoint in  $T$ .

# Notation (cont.)

$\delta(S)$



$E(S)$



# SCVRP, 2-index model

(VRP1)

$$\min \sum_{e \in E} c_e x_e \quad (1)$$

Subject to:

$$\sum_{e \in \delta(i)} x_e = 2 \quad \forall i \in V \setminus \{0\} \quad (2)$$

$$\sum_{e \in \delta(0)} x_e = 2K \quad (3)$$

$$\sum_{e \in \delta(S)} x_e \geq 2r(S) \quad \forall S \subseteq V \setminus \{0\}, S \neq \emptyset \quad (4)$$

$$x_e \in \{0, 1\} \quad \forall e \notin \delta(0) \quad (5)$$

$$x_e \in \{0, 1, 2\} \quad \forall e \in \delta(0) \quad (6)$$

$x_e$ : One variable for each edge in the graph.  $x_e$  is 1 if the corresponding edge is used in the solution and 0 otherwise.

$r(S)$ : The minimum number of vehicles needed to serve the customers in  $S$ . If we replace  $r(S)$  by  $\frac{d(S)}{C}$  we still have a valid IP model. [-show example of how the capacity constraints work].

# SCVRP, 2-index model (cont.)

(VRP2)

$$\min \sum_{i \in V \setminus \{n\}} \sum_{j > i} c_{ij} x_{ij} \quad (7)$$

Subject to:

$$\sum_{h < i} x_{hi} + \sum_{i > j} x_{ij} = 2 \quad \forall i \in V \setminus \{0\} \quad (8)$$

$$\sum_{j \in V \setminus \{0\}} x_{0j} = 2K \quad (9)$$

$$\sum_{i \in S} \sum_{\substack{h < i \\ h \notin S}} x_{hi} + \sum_{i \in S} \sum_{\substack{j > i \\ j \notin S}} x_{ij} \geq 2r(S) \quad \forall S \subseteq V \setminus \{0\}, S \neq \emptyset \quad (10)$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j \in V \setminus \{0\} \quad (11)$$

$$x_{ij} \in \{0, 1, 2\} \quad \forall j \in V \setminus \{0\} \quad (12)$$

# SCVRP, 3-index model

$x_{ek}$ : Variable that indicates if edge  $e$  is traversed by vehicle  $k$ .

$y_{ik}$ : Variable that indicates if customer  $i$  is served by vehicle  $k$ .

(VRP3)

$$\min \sum_{e \in E} c_e \sum_{k=1}^K x_{ek} \quad (13)$$

Subject to:

$$\sum_{k=1}^K y_{ik} = 1 \quad \forall i \in V \setminus \{0\} \quad (14)$$

$$\sum_{k=1}^K y_{0k} = K \quad (15)$$

$$\sum_{e \in \delta(i)} x_{ek} = 2y_{ik} \quad \forall i \in V, k = 1, \dots, K \quad (16)$$

$$\sum_{i \in V} d_i y_{ik} \leq C \quad \forall k = 1, \dots, K \quad (17)$$

$$\sum_{e \in \delta(S)} x_{ek} \geq 2y_{hk} \quad \forall S \subseteq V \setminus \{0\}, S \neq \emptyset, h \in S, k = 1, \dots, K \quad (18)$$

$$y_{ik} \in \{0, 1\} \quad \forall i \in V, k = 1, \dots, K \quad (19)$$

$$x_{ek} \in \{0, 1\} \quad \forall e \notin \delta(0), k = 1, \dots, K \quad (20)$$

$$x_{ek} \in \{0, 1, 2\} \quad \forall e \in \delta(0), k = 1, \dots, K \quad (21)$$

# CVRP, two commodity flow model

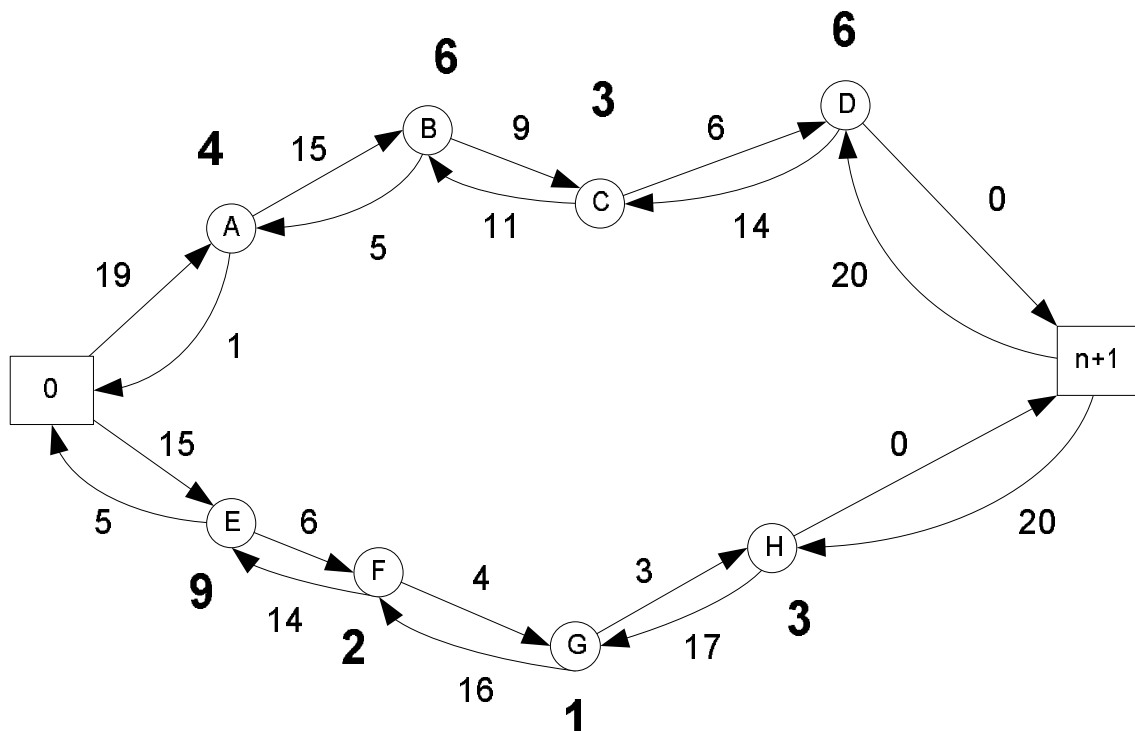
Construct a new graph  $G' = (V', E')$  from  $G$  by creating a copy of the depot as node  $n + 1$ . Connect node  $n + 1$  to all nodes  $i \in V$  by creating edges with weights corresponding to the weight of the edge from node 0 to  $i$ .

A route in  $G'$  is represented as a path starting at node 0 and ending at node  $n + 1$ .

The variables  $x_{ij}$  indicates if a vehicle uses edge  $(i, j)$ .

The variables  $y_{ij}$  defines a *commodity flow*. If a vehicle travels from  $i$  to  $j$  using edge  $(i, j)$  then  $y_{ij}$  gives the vehicle load while travelling along the arc and  $y_{ji}$  gives the empty space on the vehicle while traversing the edge.

**Veh. Cap = 20**



# CVRP, two commodity flow model (cont.)

(VRP4)

$$\min \sum_{(i,j) \in A'} c_{ij} x_{ij} \quad (22)$$

Subject to:

$$\sum_{j \in V'} (y_{ji} - y_{ij}) = 2d_i \quad \forall i \in V' \setminus \{0, n+1\} \quad (23)$$

$$\sum_{j \in V' \setminus \{0, n+1\}} y_{0j} = d(V \setminus \{0, n+1\}) \quad (24)$$

$$\sum_{j \in V' \setminus \{0, n+1\}} y_{j0} = KC - d(V \setminus \{0, n+1\}) \quad (25)$$

$$\sum_{j \in V' \setminus \{0, n+1\}} y_{n+1,j} = KC \quad (26)$$

$$y_{ij} + y_{ji} = Cx_{ij} \quad \forall (i, j) \in A' \quad (27)$$

$$\sum_{j \in V'} (x_{ij} + x_{ji}) = 2 \quad \forall i \in V' \setminus \{0, n+1\} \quad (28)$$

$$y_{ij} \geq 0 \quad \forall (i, j) \in A' \quad (29)$$

$$x_{ij} \in \{0, 1\} \quad \forall (i, j) \in A' \quad (30)$$

# SCVRP, set partitioning model

(VRP5)

$$\min \sum_{j=1}^q c_j x_j \quad (31)$$

Subject to:

$$\sum_{j=1}^q a_{ij} x_j = 1 \quad \forall i \in V \setminus \{0\} \quad (32)$$

$$\sum_{j=1}^q x_j = K \quad (33)$$

$$x_j \in \{0, 1\} \quad \forall j \in \{1, \dots, q\} \quad (34)$$

# Cutting planes

- Ultimately we want to solve our IP problem, for example the two-index formulation (VRP1) or the two-commodity flow model (VRP4). Let us only consider (VRP1) in the following
- We could do this by typing the model into CPLEX and let the CPLEX MIP solver do the hard work, but we wouldn't get very far because:
  - The number of constraints in the VRP1 grows exponentially with the size of the problem
  - The CPLEX solver is a general purpose solver. We must expect to be able to do better by making specialized algorithms

## Cutting planes (cont.)

- What we can do is remove the capacity inequalities:

$$\sum_{e \in \delta(S)} x_e \geq 2r(S) \quad \forall S \subseteq V \setminus \{0\}, S \neq \emptyset$$

and solve the linear relaxation of the resulting problem to get a lower bound. This gives us the following linear program.

$$\min \sum_{e \in E} c_e x_e \quad (35)$$

Subject to:

$$\sum_{e \in \delta(i)} x_e = 2 \quad \forall i \in V \setminus \{0\} \quad (36)$$

$$\sum_{e \in \delta(0)} x_e = 2K \quad (37)$$

$$0 \leq x_e \leq 1 \quad \forall e \notin \delta(0) \quad (38)$$

$$0 \leq x_e \leq 2 \quad \forall e \in \delta(0) \quad (39)$$

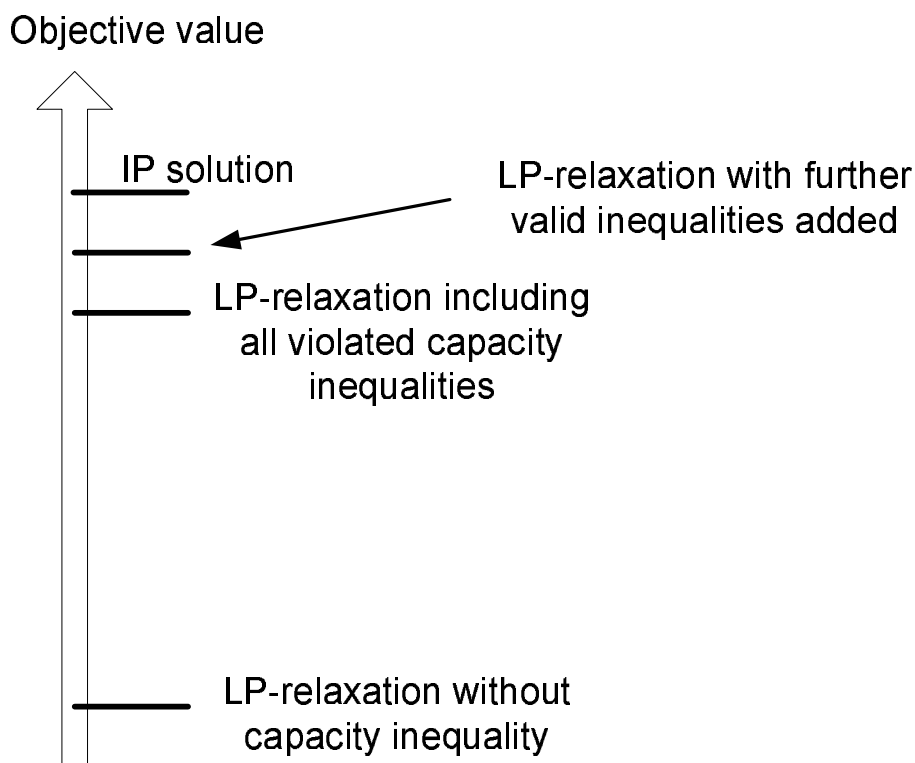
- The lower bound provided by (35)-(39) could be used in a branch and bound algorithm, and we thereby have an algorithm to solve the IP problem.
- Unfortunately the lower bound associated with the linear program is very poor and we would not be able to solve very large problems.

## Cutting planes (cont.)

- The linear relaxation (35)-(39) would almost always violate one or more of the constraints

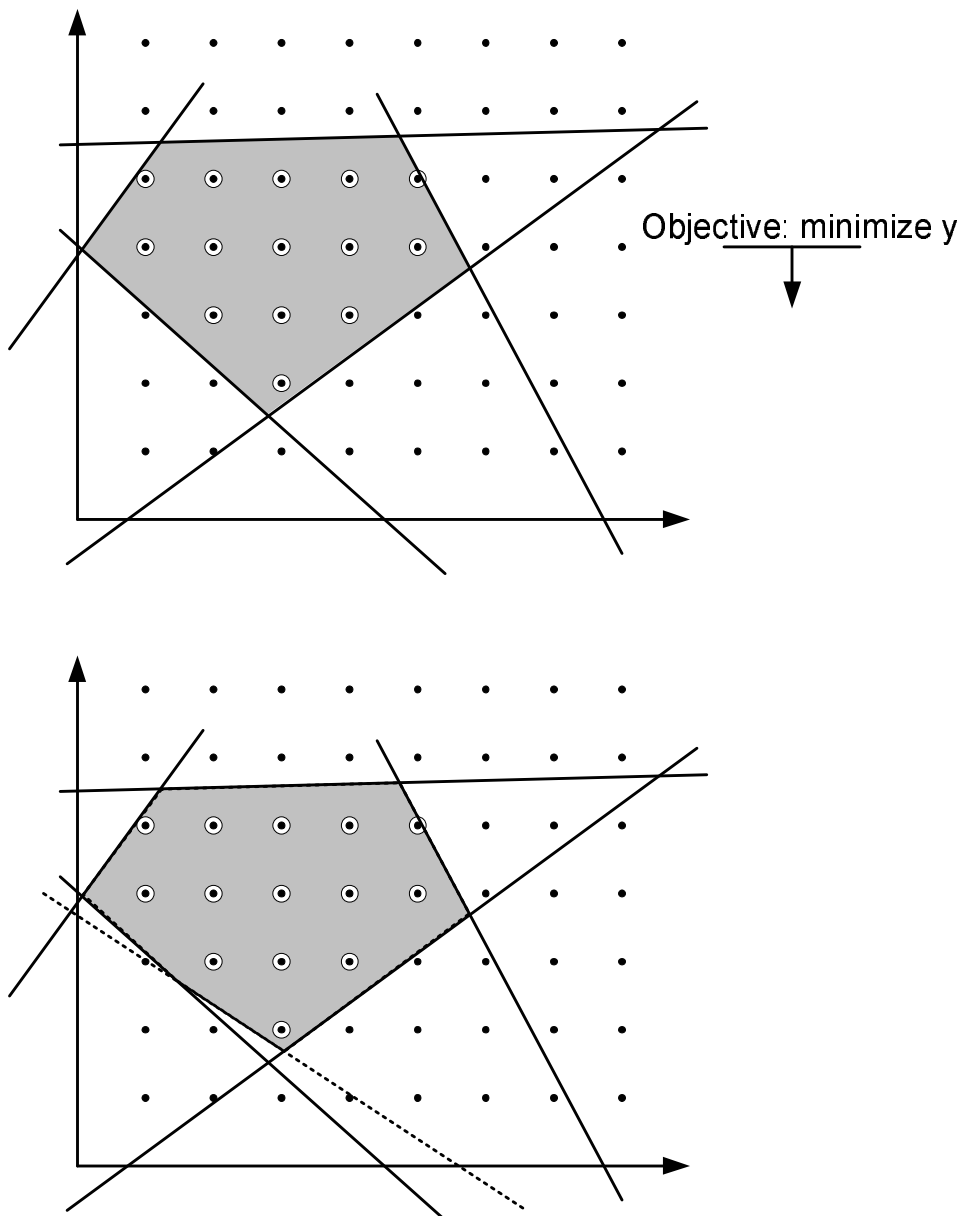
$$\sum_{e \in \delta(S)} x_e \geq 2r(S) \quad \forall S \subseteq V \setminus \{0\}, S \neq \emptyset$$

we removed from the problem earlier. Thus if we can find the constraints that are violated, we can add these constraints to our LP problem. This would most likely increase the objective of the linear program as we are minimizing over a smaller domain.

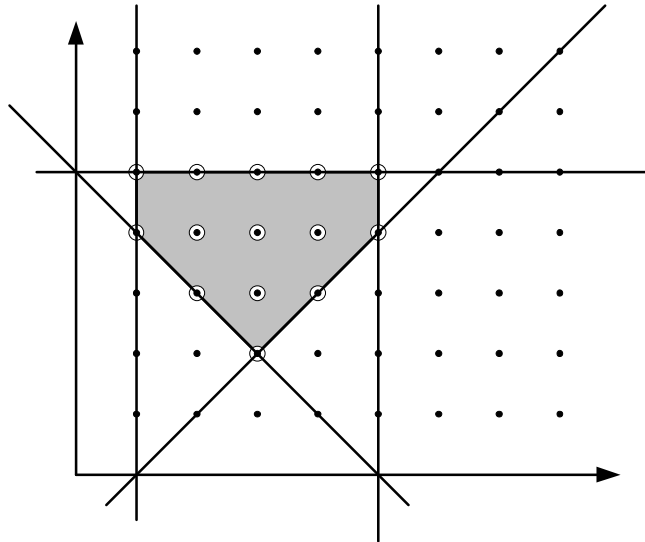


## Cutting planes (cont.)

- Even when we have added all violated capacity inequalities we can improve our lower bound further by adding more valid inequalities.



# Cutting planes (cont.)



# Branch & Cut pseudo code

Solution : GlobalBest

Branch-and-cut(subproblem  $s$ , upper bound  $u$ )

repeat

$v$  = find violated valid inequalities in  $s$

    add valid inequalities  $v$  to  $s$

until  $v = \emptyset$

if (LP( $s$ ).val >  $u$ )

    return  $u$ ;

if (LP( $s$ ).sol is integral)

    GlobalBest = LP( $s$ ).sol

    return LP( $s$ ).val

else

    ( $s_1$ ,  $s_2$ ) = genSubProbs( $s$ );

    return Branch-and-cut( $s_2$ , Branch-and-cut( $s_1$ ,  $u$ ))

## Cuts - Capacity cuts

Consider (VRP1) again:

$$\sum_{e \in \delta(S)} x_e \geq 2r(S) \quad \forall S \subseteq V \setminus \{0\}, S \neq \emptyset$$

- $r(S)$  lower bound on number of vehicles needed to serve the customers in  $S$ .
  1.  $r_1(S) = d(S)/C$
  2.  $r_2(S) = \lceil d(S)/C \rceil$
  3.  $r_3(S)$  : Number of bins of capacity  $C$  needed to pack “the items” in  $S$  (Bin packing problem - NP-hard)
- Example  $C = 20$  demands in  $S : \{15, 15, 6\}$   
 $r_1(s) = 36/20 = 1.8$ ,  $r_2(s) = \lceil 36/20 \rceil = 2$ ,  $r_3(s) = 3$
- When plugged into (VRP1-3) the three bounds all defines the same problem, but when calculating lower bounds using the LP formulation, we would prefer the tightest of the three bounds, that is  $r_3$ , in order to create tight cuts.
- $r_3$  is called a *weak capacity* constraint in the paper as it can be improved further:

## Cuts 2 - Capacity cuts (cont.)

- *K-Partition*: A subdivision of  $V_0$  into  $K$  disjoint subsets  $S_1, \dots, S_K$ .
- $\mathcal{P}$  : The set of all feasible K-Partitions.
- $\forall P \in \mathcal{P}, S \subseteq V_0 : \beta(P, S) = |\{i : S_i \cap S \neq \emptyset\}|$
- $\beta(P, S)$  measures the number of partitions (routes) from  $P$  that are needed to cover (serve) the elements (customers) in  $S$ .

$$r_4(S) = \min_{P \in \mathcal{P}} \beta(P, S)$$

- We have that  $\forall S \subseteq V_0, S \neq \emptyset : r_4(S) \geq r_3(S)$  because if there exists a  $S \subseteq V_0, S \neq \emptyset$  such that  $r_4(S) < r_3(S)$  then it must be possible to pack the elements of  $S$  into less bins than given by  $r_3(S)$  and this is in conflict with the definition of  $r_3$ . On the other hand it can happen that  $r_4(S) > r_3(S)$  for some subsets as the following example shows.
- $K = 4, C = 7, d = \{5, 3, 3, 3, 4, 4, 4, 2\}$ .  $S$  is given by the first four customers:

$$r_1(S) = \frac{14}{2} = 2 = r(S) = \left\lceil \frac{14}{2} \right\rceil < r_3(S) = 3 < r_4(S) = 4$$

- Usually at most  $r_3$  is used in branch and cut implementations.

# Separating capacity constraints

- Separation of capacity constraints given by  $r_1$  can be done in polynomial time.
- It is also possible to find violated constraints of type  $r_2$  that violates the constraint by at least  $\epsilon$  in polynomial time. For practical purposes, researchers have used heuristics to identify violated constraints of this type.
- It is NP hard to separate the constraints of type  $r_3$  and  $r_4$ , as a polynomial time separation algorithm would allow us to solve the decision version of the Bin Packing problem in polynomial time.
- If solution is integer, then it is easy to discover violated constraints (by detecting sub-tours or routes with violated capacity).

# Separating type $r_1$ capacity constraints

- Problem: Given a fractional vector  $\bar{x}_e$ , determine if the vector violates any of the constraints:

$$\sum_{e \in \delta(S)} x_e \geq 2 \frac{d(S)}{C} \quad \forall S \subseteq V \setminus \{0\}, S \neq \emptyset$$

- Method: Define a new graph  $G' = (V, E)$  with edge weights  $\bar{x}'_e$  where  $\bar{x}'_e = \bar{x}_e, \forall e \notin \delta(0)$  and  $\bar{x}'_e = \bar{x}_e - 2d_i/C, \forall e = (0, i) \in \delta(0)$ .
- Find a minimum cut (classic graph problem) in this graph, that is a set  $S^*$  that minimizes:

$$f(S) = \sum_{e \in \delta(S)} \bar{x}'_e$$

- If the value of the cut,  $f^* = \min_{S \subseteq V} \{f(S)\}$  is less than zero, then the cut identifies a violated capacity constraint because:
- Assume  $0 \notin S^*$  (if  $0 \in S^*$  then we can use the cut  $T = V \setminus S^*$  instead).

## Separating type $r_1$ capacity constraints

- Define  $\delta_0(S) = \delta(0) \cap \delta(S) = (\{0\} : S)$
- By assumption we have that

$$\begin{aligned}
 & \sum_{e \in \delta(S)} \bar{x}'_e < 0 \\
 \Rightarrow & \sum_{e \in \delta(S) \setminus \delta_0(S)} \bar{x}'_e + \sum_{e \in \delta_0(S)} \bar{x}'_e < 0 \\
 \Rightarrow & \sum_{e \in \delta(S) \setminus \delta_0(S)} \bar{x}_e + \sum_{e=(0,i) \in \delta_0(S)} (\bar{x}_e - 2d_i/C) < 0 \\
 \Rightarrow & \sum_{e \in \delta(S) \setminus \delta_0(S)} \bar{x}_e + \sum_{e \in \delta_0(S)} \bar{x}_e - \sum_{e=(0,i) \in \delta_0(S)} 2d_i/C < 0 \\
 \Rightarrow & \sum_{e \in \delta(S)} \bar{x}_e - \sum_{(0,i) \in \delta_0(S)} 2d_i/C < 0 \\
 \Rightarrow & \sum_{e \in \delta(S)} \bar{x}_e < 2 \frac{d(S)}{C}
 \end{aligned}$$

- That is, the set  $S$  violates the capacity constraint.
- When such a  $S$  is found the constraint can be strengthened by using the  $r_2$ ,  $r_3$  or  $r_4$  bound instead of  $r_1$ .

# Solving the min-cut problem

- The general min-cut problem with arbitrary edge weights is NP-hard.
- Polynomial time algorithms for determining the minimum cut that separates nodes  $s$  and  $t$  (called *min s-t cut*) in a graph with positive edge weights are known. We are going to transform our min-cut problem to such a problem.

- **Lemma 1:**

If  $G$  is a complete graph where all the edges with negative edge weights are adjacent to  $s$  or  $t$  (such a graph is called *s-t negative*) then the min s-t cut problem can be solved in polynomial time.

**Proof:**

For all nodes  $i \neq s, t$  we know that precisely one of the two edges  $(s, i)$  and  $(i, t)$  must be part of the cut. Thus we can add an arbitrarily large constant to both edge weights without changing the min s-t cut. This can be done for all nodes and thereby we can transform the problem to a min s-t cut problem with positive edge weights.

## Solving the min-cut problem (cont.)

- **Lemma 2:**

If  $G$  is a complete graph where all the edges with negative edge weights are adjacent to a single node  $s$  (such a graph is called *star negative*) then we can solve the min cut problem on  $G$  in polynomial time

**Proof:**

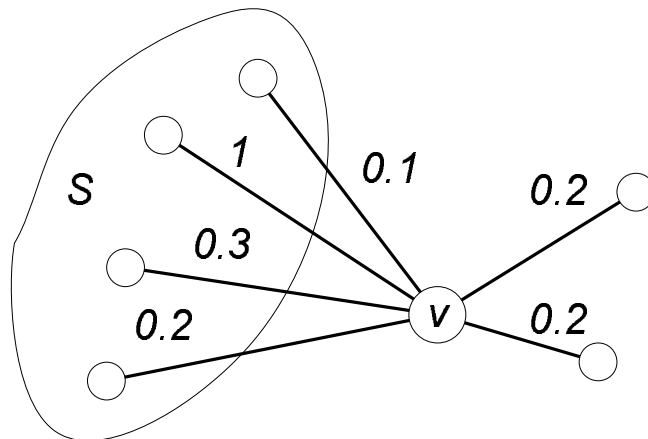
Construct a new graph  $G''$  by adding a new node  $t$  to  $G$  and connecting  $t$  to all nodes in  $G$  by edges with weight 0.  $G''$  is obviously  $s$ - $t$  negative and we can solve the min  $s$ - $t$  cut problem in polynomial time.

Let  $S''$  be the "subset in the cut that does not contain  $t$ ". If  $S'' = V$  then there are no negative weight cuts in  $G$  and the solution to our min cut problem is  $\emptyset$  otherwise the solution to the min cut problem in  $G$  is  $S''$ .

- Our graph  $G'$  (the one we constructed from our  $\bar{x}_e$  vector) is star negative so it is possible to solve the min cut problem on this graph.

# Separating type $r_2$ capacity constraints

- As stated before, it is possible to find violated constraints of type  $r_2$  that violates the constraint by at least  $\epsilon$  in polynomial time but typically these constraints are identified using heuristics in branch and cut codes.
- Greedy heuristic:** As earlier,  $\bar{x}$  is the fractional solution to (VRP1).  
Given  $S \subseteq V, v \notin S$  we call  $\bar{x}(\{v\} : S)$  the amount by which  $v$  sees  $S$ .



In the example,  $\bar{x}(\{v\} : S) = 1.6$ . Notice that because of the degree constraint:

$$\sum_{e \in \delta(i)} x_e = 2 \quad \forall i \in V \setminus \{0\}$$

$\bar{x}(\{v\} : S)$  can at most become 2 no matter how we choose  $v \in V_0$  and  $S$ .

## Greedy heuristic (cont.)

- The starting point of the greedy heuristic is a set  $S_0 \subset V$ . The paper says that  $\bar{x}(\delta(S_0))$  should be 2, but the heuristic also works without this restriction. A good candidate for a  $S_0$  is a set containing only one node.
- Now the heuristics “grows”  $S$  starting from  $S_0$  until  $S = V_0$ . In each iteration, the node  $v$  that maximizes  $\bar{x}(\{v\} : S)$  (the node that sees  $S$  the most) is added.
- The reason for this choice is that the addition of this  $v$  increases  $\bar{x}(\delta(S))$  the least because of the degree constraint (the sum of the  $\bar{x}$  variables corresponding to edges adjacent to any node must be 2). At the same time the addition of  $v$  increases  $d(S)$ . This is good as we want to find sets  $S$  that violates:

$$\sum_{e \in \delta(S)} x_e \geq 2 \frac{d(S)}{C} \quad \forall S \subseteq V \setminus \{0\}, S \neq \emptyset$$

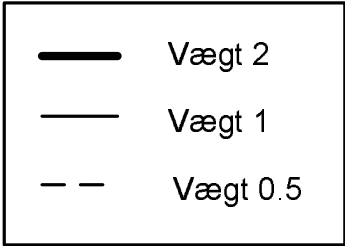
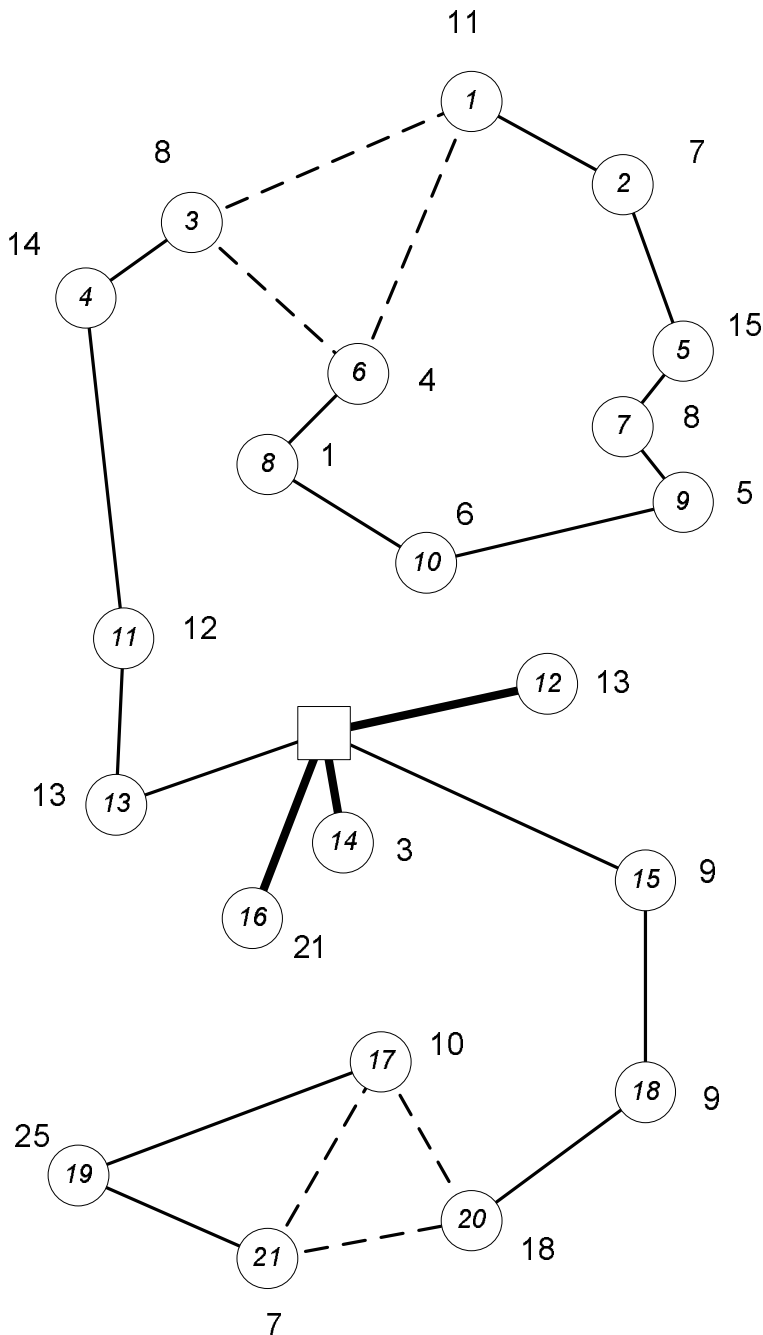
- Alternative selection strategies?

## Greedy heuristic (cont.)

- Whenever we encounter a set  $S$  that violates the capacity constraint it is stored as a candidate for a new cut we should add to our LP problem.
- The heuristic is restarted from different starting sets  $S_0$ , for example all sets containing one node.
- A randomized version of the heuristic can be obtained by adding some randomization to the process of selecting the next node to add. This could for example be done by allowing the heuristic to choose randomly between say the 5 nodes that see  $S$  the most.  
This would lead to a heuristic that produces different results when applied to the same starting set. This might enable us to find more violated constraints.
- It has been proposed to use tabu search to find violated constraints. Roughly speaking, the current solution is a set  $S$  of nodes, and the neighborhood of a solution is the set of nodes that can be obtained by adding or removing a node from the current set. The objective is to minimize  $\bar{x}(\delta(S))$  while  $d(S)$  is kept within a certain interval.

# Example

E-n22-k4, C=60



LP-sol: 308.5

IP-Opt: 375

S	$X(\delta(S))$	$d(S)$	$2 \left\lceil \frac{d(s)}{C} \right\rceil$
19			

S	$X(\delta(S))$	$d(S)$	$2 \left\lceil \frac{d(s)}{C} \right\rceil$
5			



# Branching

- Several strategies for branching. Simplest: branch on edge. Problem: creates “asymmetric” subproblems (enforcing an edge to be in the solution is stronger than enforcing the variable to be zero).
- Three strategies for branching on an edge is investigated in the paper:
  - A1: Chose edge with value close to 0.5 and high cost
  - A2: Chose edge with value close to 0.75 and high cost
  - A3: Chose 10 edges to branch on with value between 0.45 and 0.65. Calculate LP bound in subproblems (no cut generation). Chose edge where the minimum of the two children is largest (strong branching).

## Branching on inequalities

- For any subset  $S \subseteq V$  we know that  $x(\delta(S))$  must be a positive even integer in an optimal solution as any tour entering the subset also must leave the subset.
- Thus if  $\bar{x}(\delta(S)) \approx 2t + 1$  where  $t$  is an integer then it seems profitable to create the branches:  $\bar{x}(\delta(S)) \leq 2t$  and  $\bar{x}(\delta(S)) \geq 2t + 2$ . Also imbalance in search tree when  $t = 1$ .
- Six strategies:
  - B1: Select  $S$  for which  $x(\delta(S))$  is closest to 3.0
  - B2: Select  $S$  for which  $x(\delta(S))$  is closest to 2.85
  - B3: Select  $S$  for which  $x(\delta(S))$  is closest to 3.15
  - B4: Select  $S$  for which  $2.75 \leq x(\delta(S)) \leq 3$  and  $d(S)$  maximum
  - B5: Select  $S$  for which  $2.75 \leq x(\delta(S)) \leq 3$  and distance from  $S$  to depot is maximum.
  - B6: Select  $S$  for which  $2.75 \leq x(\delta(S)) \leq 3$  and  $S$  contains the largest number of *super-nodes*.

# Tables comparing branching strategies

# Cut pool

- While generating cuts we may find that some of the constraints that we have added earlier no longer are binding. We can discover this by noting that the dual variable corresponding to a constraint is zero if the constraint is not binding.
- One could throw away such constraints, but practice shows that it is worthwhile to store these constraints in what we call a *cut pool* as the constraints might become binding as other constraints are added to the linear program or when variables are fixed by branching.
- One would usually check the cut pool for violated constraints before invoking any separation routines.
- It may be profitable to clean up the cut pool from time to time.

# Brief introduction to set partitioning combined with cutting planes solution method

$$\min \sum_{j=1}^q c_j x_j \quad (40)$$

Subject to

$$\sum_{j=1}^q a_{ij} x_j = 1 \quad \forall i \in V \setminus \{0\} \quad (41)$$

$$\sum_{j=1}^q x_j = K \quad (42)$$

$$x_j \in \{0, 1\} \quad \forall j \in \{1, \dots, q\} \quad (43)$$

When we are solving the linear relaxation of the set partition formulation we typically get a fractional solution (if we get an integer solution, then we are done). This solution may for example tell us to use one fourth of one route, the half of another route and one fourth of a third route. These fractional routes implicitly create a fractional  $\bar{x}$  vector with one entry per edge. We can apply standard cuts to this vector to improve the lower bound (this is not as simple as it sounds in reality). This strengthens the LP relaxation and leads to state of the art results.

# Computational experiments

Insert tables

# Topics covered

- The following sections in the distributed paper should be read:  
Section 3.1-3.3.1, 3.4.1-3.4.2, 3.5-3.7.

# References

- S. T. McCormick, M.R. Rao, G. Rinaldi, *easy and difficult objective functions for max cut*, Math. Program. Ser. B 94, 459-466 (2003).
- R. Fukasawa, M. Poggi de Aragao, M. Reis, E. Uchoa, *Robust Branch-and-Cut-and-Price for the Capacitated Vehicle Routing Problem*, Technical Report (2003).
- T.K. Ralphs, L. Kopman, W.R. Pulleyblank, L.E. Trotter, *On the capacitated vehicle routing problem*, Math. Program. Ser. B 94, 343-359 (2003).
- P Augerat, J.M. Belenguer, E. Benavent, A. Corberán, D. Naddef, *Seperating capacity constraints in the CVRP using tabu search*, European Journal of Operational Research 106, 546-557 (1998)
- U. Blasum, W. Hochstättler, *Application of the Branch and Cut Method to the Vehicle Routing Problem*, technical report (2002).
- R. Baldacci, E. Hadjiconstantinou, A. Mingozzi, *An Exact Algorithm for the Capacitated Vehicle Routing Problem based on a Two-Commodity Network Flow Formulation*, to appear in Operations Research (2004?).
- J. Lysgaard, A. Letchford, R.W. Eglese, *A new branch-and-cut algorithm for the capacitated vehicle routing problem*, Math. Program. Ser. A (published online 2003).
- D. Naddef, G. Rinaldi, *Branch-and-Cut Algorithms for the Capacitated VRP*, In "The Vehicle Routing Problem", P. Toth, D. Vigo (eds.), SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, 2002