1. Introduction

2. Forecasting

3. Transportation Systems

4. Transportation Models

5. Inventory Systems

6. Supply Chain Systems

**Vehicle Routing Problems Overview**

- **Single Origin-Destination Routing**
- **Multiple Origin-Destination Routing**
- **Single Vehicle Round-Trip Routing**
- **Vehicle Routing and Scheduling**

**Shortest Path Applications: Route Planning to ATL Airport**

<table>
<thead>
<tr>
<th>Time</th>
<th>Mile</th>
<th>Instruction</th>
<th>For</th>
</tr>
</thead>
<tbody>
<tr>
<td>5:00 PM</td>
<td>0</td>
<td>Depart 765 Peachtree Dr NW, Atlanta, GA 30318 on Peachtree Dr NW (North)</td>
<td>0.0 mi</td>
</tr>
<tr>
<td>5:01 PM</td>
<td>0.3</td>
<td>Turn LEFT (North) onto Dalney St NW</td>
<td>0.3 mi</td>
</tr>
<tr>
<td>5:02 PM</td>
<td>0.5</td>
<td>Turn RIGHT (East) onto 10th St NW</td>
<td>0.5 mi</td>
</tr>
<tr>
<td>5:03 PM</td>
<td>0.9</td>
<td>Turn RIGHT (South) onto Ramp</td>
<td>0.9 mi</td>
</tr>
<tr>
<td>5:03 PM</td>
<td>1.1</td>
<td>Merge onto I-85 (S) (South)</td>
<td>1.1 mi</td>
</tr>
<tr>
<td>5:08 PM</td>
<td>1.8</td>
<td>Continue (South) on I-85</td>
<td>1.8 mi</td>
</tr>
<tr>
<td>5:12 PM</td>
<td>1.3</td>
<td>Continue (South) on I-85</td>
<td>1.3 mi</td>
</tr>
<tr>
<td>5:14 PM</td>
<td>1.5</td>
<td>Continue (South) on I-85</td>
<td>1.5 mi</td>
</tr>
<tr>
<td>5:17 PM</td>
<td>1.9</td>
<td>Continue (West) on Airport Blvd [S Terminal Pkwy]</td>
<td>1.9 mi</td>
</tr>
<tr>
<td>5:20 PM</td>
<td>2.0</td>
<td>Continue (South-West) on Airport Circle</td>
<td>2.0 mi</td>
</tr>
<tr>
<td>5:21 PM</td>
<td>2.1</td>
<td>Bear RIGHT (East) onto S Terminal Pkwy</td>
<td>2.1 mi</td>
</tr>
<tr>
<td>5:22 PM</td>
<td>2.3</td>
<td>Bear RIGHT (East) onto Local roads</td>
<td>2.3 mi</td>
</tr>
<tr>
<td>5:27 PM</td>
<td>2.4</td>
<td>Arrive Hartsfield-Atlanta International Airport</td>
<td>2.4 mi</td>
</tr>
</tbody>
</table>

**Shortest Path Applications: Point-To-Point Instructions**

- **Summary:** 13.9 miles (23 minutes)
Shortest Path Application: Excite Driving Instructions

Single Vehicle Origin-Destination Routing

- Shortest Path Problem (SPP)
  - Network nodes = points to be visited
  - Network links = connecting the nodes
  - Dijkstra’s optimal algorithm (1959)
  - 100,000 nodes

Shortest Path Problem (SPP) Variants

- One Source to One Sink (s to t)
- One Source to All Sinks (s to all)
- All Pairs
- k Shortest Paths (Sensitivity)
- All Non-Negative Costs (Label Setting)
- General Costs (Label Correcting)
- Longest Path in Acyclic Graphs (PERT and CPM)

Labeling Algorithms

- Temporary Labels = Upper Bound
  - Permanent Label = Exact Path Length
- Reduce Labels by Iterative Procedure
- Label Setting
  - One temporary label becomes permanent per iteration
- Label Correcting
  - All temporary labels become permanent at the last iteration
Dijkstra’s Algorithm Illustration

Dijkstra’s Algorithm Solution

Dijkstra’s Algorithm Notation

- $N$ = Set of all nodes
- $P$ = Set of permanently labeled nodes
- $T$ = Set of nodes with temporary labels (complement of $P$, $N=P+T$)
- $L(k)$ = Label of node $k$
- $c_{ij}$ = Arc length from node $i$ to $j$
- $\Gamma(k)$ = all successor nodes of node $k$ (forward star)
- $\text{pred}(k)$ = predecessor node of node $k$ on the shortest path to node $k$

Dijkstra’s SPP Algorithm Description

1. Set all node labels $l(x) = \infty$, set $l(s) = 0$, set all nodes to temporary
2. Find temporary node with minimum label, $l(p) = \min\{l(x)\}$
3. For all temporary $x \in \Gamma(p)$ update labels $l(x) = \min\{l(x), l(p) + c(p,x)\}$
4. Mark node $p$ as permanent
5. If all destinations are permanent stop, else go to step 2
Dijkstra’s SPP Formal Algorithm Description

\[ P = \emptyset, \quad T = N, \quad l(s) = 0, \quad l(i) = \infty \quad \forall i \in N \]

while \(|P| < n\) {
    \( k \leftarrow l(k) = \min\{l(j) : j \in T\} \)
    \( P \leftarrow P \cup \{k\}, \quad T \leftarrow T - \{k\} \)
    for \( j \in \Gamma(k) \cap T \)
        if \( l(j) > l(k) + c_{kj} \)
            \( l(j) = l(k) + c_{kj}, \quad \text{pred}(j) = k \)
    \}
}

Dijkstra’s SPP Algorithm Characteristics

- Forward dynamic programming
- Nonnegative arc “lengths” or “costs”
- \(O(n^2) + O(m)\) or \(O(n^2)\) for fully dense graphs
- Directed out-tree rooted at \(s\)
- Node selection computationally most expensive
- 100,000 nodes

SPP Example Network

SPP Example Distances

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>18</td>
</tr>
<tr>
<td>4</td>
<td>25</td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>20</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>12</td>
</tr>
</tbody>
</table>
**Network Flow Formulation**

\[
\begin{align*}
\text{Min } & \sum_{i} \sum_{j} c_{ij} x_{ij} \\
\text{s.t. } & \sum_{j} x_{ij} - \sum_{k} x_{ki} = b_{i} \quad \forall i \\
& 0 \leq x_{ij} \leq u_{ij} \quad \forall ij
\end{align*}
\]

*Flow Balance: Out - In = External In*

**Network Formulation Characteristics**

- One variable for each arc and commodity (flow)
- One conservation of flow constraint for each node and commodity
- Flows from the outside (sign convention: entering = positive)
- Individual and joint upper (and lower) bounds

**Network Example**

---

*Copyright © 1994-2003, Marc Goetschalckx, All rights reserved.*
**Algebraic Network Formulation Example**

\[
\begin{align*}
\text{min} & \quad 3x_{s1} + 5x_{s2} + 7x_{s3} + 6x_{s1} + 5x_{23} + 3x_{3t} + 6x_{st} \\
\text{s.t.} & \quad 1 - x_{s1} - x_{s2} - x_{s3} = 0 \\
& \quad x_{s1} - x_{13} = 0 \\
& \quad x_{s2} - x_{23} = 0 \\
& \quad x_{13} + x_{23} - x_{3t} = 0 \\
& \quad x_{s4} - x_{4t} = 0 \\
& \quad x_{3t} + x_{4t} - 1 = 0
\end{align*}
\]

**Shortest Path Example Network**

**Shortest Path Example Excel Arc Capacities**

**Shortest Path Example Excel Arc Costs**
Shortest Path Example
Excel Solution Arc Flows

Dijkstra’s Algorithm
Sparse Graphs
- Heap implementations (binary, Fibonacci, radix, ...)
- Running times
  - $O(m+n \log n)$ for Fibonacci heap
  - $O(m \log n)$ for binary heap
- Intricate implementation

Shortest Path Example: Excel Solution Objective Function

Shortest Path Exercise
Shortest Path Exercise 2

Directed Graph Representation

- Adjacency matrix
- Successor list
- Successor linked list

Directed Graph Adjacency Matrix

- $A[i, j]$ present if arc from node $i$ to node $j$

Directed Graph Adjacency Matrix Characteristics

- $N^2$ storage memory
- Constant $O(1)$ lookup, addition, deletion
- Linear $o(N)$ successor and predecessor loops
Directed Graph Successor Array

- Head or first arc array [nodes+1]
- to-node array with every arc [arcs]

**Characteristics**
- N + 2M memory storage
- Linear o(N) lookup, linear o(N) insertion and deletion
  - Plus constant time to move the array elements for insertion and deletion
- Linear o(N) successor loop, linear o(M) predecessor loop

Directed Graph Successor Double Linked List

- Head or first arc array [nodes]
- to-node double linked list with every arc [arcs]

**Characteristics**
- N + 4M memory storage
- Linear O(N) Lookup,
  - Constant O(1) Insertion,
  - Linear O(N) Deletion
- Linear O(N) Successor Loop,
  - Linear O(M) Predecessor Loop
Double Linked List Deletion

- k to be deleted element
  - If Pred(k) then Succ(Pred(k)) = Succ(k)
  - If Succ(k) then Pred(Succ(k)) = Pred(k)

Double Linked List Insertion

- k to be inserted element after p
  - Succ(k) = Succ(p), Pred(k)=p, Succ(p)=k
  - If Succ(k) then Pred(Succ(k))=k
- k to be inserted element before s
  - Pred(k)=Pred(s), Succ(k)=s, Pred(s)=k
  - if Pred(k) then Succ(Pred(k))=k

Directed Graph Example

Directed Graph Example

Adjacency Matrix

Copyright © 1994-2003, Marc Goetschalckx, All rights reserved.
Directed Graph Example
Successor Array

1 2 1
4 3 4
5 6
6 2
7 4 5
8 6
8

Directed Graph Example
Successor Linked List

1 - 2 1 2
4 1 3 4 3
5 2 5 6 4
6 3 4 2 5
7 4 4 5 6
- 5 6 3 7
6 6 7 -

Arc (3-4) Deletion:
Adjacency Matrix

1 2 3 4 5 6
1 1 4 6
2 2
3
4 3
5 7
6

Arc (3-4) Deletion:
Successor Array

1 2 1
4 3 4
5 5 6
6 4 2
7 6
8 6 7
7
Arc (3-4) Deletion
Successor Linked List

Arc (3-6) Addition (Cost = 8)
Successor Array

Arc (3-6) Addition (Cost = 8)
Adjacency Matrix

Arc (3-6) Addition (Cost = 8)
Successor Linked List
Predecessor Loop: Data Independent Pseudo Code

* Perform action on all predecessor nodes of node k

\[
p = \text{find_first_predecessor}(k)\]
\[
\text{while } (p \neq \text{null}) \{
\text{perform_action}(p)\]
\[
p = \text{find_next_predecessor}(k, p)\]
\[
\} // \text{end while}
\]

Data Structures and Algorithms


Vehicle Routing Problems Overview

- *Single Origin-Destination Routing*
- *Multiple Origin-Destination Routing*
- *Single Vehicle Round-Trip Routing*
- *Vehicle Routing and Scheduling*
Multiple Vehicle Origin-Destination Routing Overview

- Problem Description & Variants
- Examples
- Successive Shortest Path Algorithm
- Formulations
- References

Network Flow Illustration

Network Components

- Nodes
  - Sources, sinks, intermediate
  - No capacities or cost
  - Conservation of flow
- Channels or Arcs
  - Directed (non-negative flow variables)
  - Costs and capacities

Multiple Origin-Destination Routing

- Max flow network flow problem
  - Capacity models, public sector
- Min cost network flow problem
  - Economic models, private industry
- Network simplex algorithm is very efficient
- 100,000 channels
Network Variants

- Transportation
- Transshipment
- Min Cut - Max Flow Network
- Min Cost Network
- Multicommodity Network
- Generalized Network

Integrality Property

- Solution is naturally integer if
  - Integer external flows
  - Integer capacity bounds
  - Binary conservation of flow coefficients
  - Single commodity
- Desirable for unit load transportation moves
- Does not hold for generalized network and multicommodity networks

Tactical Production-Distribution Planning Problem

- Products \( p \)
- Customers \( j \), demand \( \text{dem}_{jp} \)
- Plants \( i \), capacity \( \text{cap}_i \)
- Marginal production cost \( a_{ip} \) and resource consumption \( \text{req}_{ip} \)
  - generalized network \( \text{req} \neq 1 \)
- Transportation cost \( c_{ijp} \) and quantity \( x_{ijp} \)

Tactical Production-Distribution Planning Network
**Tactical Production-Distribution Planning Model**

\[
\text{Min } \sum_{i=1}^{F} \sum_{j=1}^{C} \sum_{p=1}^{P} (a_{ijp} + c_{ijp}) x_{ijp}
\]

\[
\text{s.t. } \sum_{i=1}^{F} x_{ijp} = \text{dem}_{jp} \quad \forall jp
\]

\[
\sum_{j=1}^{C} \sum_{p=1}^{P} \text{req}_{ijp} x_{ijp} \leq \text{cap}_i \quad \forall i
\]

\[
x_{ijp} \geq 0
\]

**Operator Scheduling Problem**

* Parameters and variables
  - Time periods with coverage requirements \( b_i \)
  - Operator shifts with costs \( c_i \) cover consecutive time periods
  - Number of operators for each shift \( x_i \)

**Operator Scheduling Linear Programming Formulation**

\[
\text{Min } cx
\]

\[
\text{s.t. } Ax \geq b
\]

\[
x \geq 0
\]

\[
\begin{bmatrix}
0 & 1 & 0 & 1 & 1 & 5 \\
1 & 1 & 0 & 0 & 1 & 12 \\
1 & 1 & 0 & 0 & 1 & 8 \\
1 & 1 & 0 & 0 & 0 & 10 \\
1 & 1 & 1 & 0 & 0 & 4
\end{bmatrix}
\]

**Operator Scheduling Transformation**

* Consecutive ones in each column
* Add negative identity matrix (row surplus variables \( s: Ax-s=b \))
* Add “zero” row (node \( N+1 \) flow balance constraint)
* Linear row operation
  - For \( r = N \) Down To 1
    \[
    \text{Row}[r+1] = \text{Row}[r+1] - \text{Row}[r]
    \]
Operator Scheduling Network Formulation (Initial)

\[
\begin{bmatrix}
0 & 1 & 0 & 1 & 1 & -1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3 \\
x_4 \\
x_5 \\
x_6
\end{bmatrix}
= \begin{bmatrix}
5 \\
12 \\
8 \\
10 \\
4 \\
0
\end{bmatrix}
\]

Operator Scheduling Network Formulation (Final)

\[
\begin{bmatrix}
0 & 1 & 0 & 1 & 1 & -1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & -1 & 0 & 1 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & -1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\
-1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3 \\
x_4 \\
x_5 \\
x_6
\end{bmatrix}
= \begin{bmatrix}
5 \\
7 \\
-4 \\
2 \\
-6 \\
-4
\end{bmatrix}
\]

Successive Shortest Paths Illustration (1)
Successive Shortest Paths
Illustration (2)

Successive Shortest Paths
Illustration (3)

Successive Shortest Paths
Illustration (4)

Successive Shortest Paths
Algorithm (1)

1. Start all flows $x = 0$, all node potentials $\pi = 0$
Successive Shortest Paths Algorithm (2)

1. Construct incremental/residual graph
   - Same nodes as original graph
   - If $x_{ij} > 0$ then add artificial arc $ji$
   - If $x_{ij} = u_{ij}$ then eliminate arc $ij$ or $d_{ij} = \infty$
   - If $x_{ij} < u_{ij}$ then $d_{ij} = c_{ij} - \pi_i + \pi_j$
   - If $x_{ij} > 0$ then $d_{ji} = -c_{ij} - \pi_j + \pi_i = -d_{ij}$
   - Add super source (sink), arcs to (from) sources (sinks) with 0 cost, capacity = remaining supply (demand)

Successive Shortest Paths Algorithm (3)

1. Find shortest path from source to sink
   - Find shortest path to any sink node $k$ with Dijkstra's algorithm
   - If no such path, stop, network flow problem is infeasible

Successive Shortest Paths Algorithm (4)

1. Compute maximum flow change on the shortest path
   - on backflow arcs, $\delta_i = x_{ij}$
   - on regular arcs, $\delta_i = u_{ij} - x_{ij}$
   - $\delta = \min \{\delta_i\}$
2. Augment flow on the shortest path
   - on backflow arcs, $x_{ji} = x_{ji} - \delta$
   - on regular arcs, $x_{ij} = x_{ij} + \delta$
   - update remaining supply and demand

Successive Shortest Paths Algorithm (5)

1. If all remaining demands are zero, stop, network is optimal
2. Update node potentials
   - $k =$ shortest path sink node
   - if node $i$ is permanent, $\pi_i = \pi_i - SP_i$
   - if node $i$ is temporary, $\pi_i = \pi_i - SP_k$
3. Goto step 2
Successive Shortest Paths Example

The diagram illustrates the successive shortest paths in a network graph. Each vertex represents a node, and the edges between the nodes represent the connections with associated costs, capacities, and flows.

Successive Shortest Paths Example (2)

The second example further demonstrates the shortest paths algorithm with different costs, capacities, and flows.

Successive Shortest Paths Example (3)

This example shows an expanded network with additional connections and parameters, highlighting the complexity of the shortest paths problem.

Successive Shortest Paths Example (4)

The fourth example provides a comprehensive view of the network, illustrating various paths and their corresponding costs, capacities, and flows.
Successive Shortest Paths Example (5)

Successive Shortest Paths Example (6)

Successive Shortest Paths Example (7)

Successive Shortest Paths Example
Min Cost Network Example
Excel Arc Capacities

Min Cost Network Flow Example
Arc Capacities

Min Cost Network Example Excel Arc Costs

Min Cost Network Example Excel Initial Zero Arcs Flows

Min Cost Network Example Excel Initial Zero Objective

Copyright © 1994-2003, Marc Goetschalckx, All rights reserved.
Min Cost Network Example
Excel Solver Parameters

Min Cost Network Example
Excel Optimal Arc Flows

Min Cost Network Example
Excel Optimal Objective

Min Cost Network Flow Formulation

\[
\min \sum_{i,j} c_{ij} x_{ij}
\]

\[\text{s.t.} \quad \sum_{h} x_{hi} + \sum_{j} x_{ij} = b_i \quad \forall i \quad [\pi_i] \]

\[0 \leq x_{ij} \leq u_{ij} \quad \forall ij \quad [\alpha_{ij}]\]

\* $b_j$ external node flow

(supply >0, or demand <0)
Residual Network

- arc(i, j) with $c_{ij}$ and $u_{ij}$
- if $x_{ij} > 0$ then
- $arc(i, j) c_{ij}^\pi = c_{ij}^{orig} + \pi_j - \pi_i$
- $r_{ij} = u_{ij} - x_{ij}$
- $arc(j, i) c_{ji}^\pi = -c_{ij}^\pi$
- $r_{ji} = x_{ij}$

Optimality Conditions

- Shortest path reduced cost
  - $d_j \leq d_i + c_{ij} \forall (i, j)$
  - $c_{ij}^d = c_{ij} + d_i - d_j \geq 0 \forall (i, j)$

- Dual variable reduced cost
  - $c_{ij}^\pi = c_{ij} - \pi_i + \pi_j \geq 0 \forall (i, j)$

Min Cost Network Flow

Primal Formulation

- Min $\sum_{i}^{N} \sum_{j}^{N} c_{ij}x_{ij}$
- s.t. $\sum_{j}^{N} x_{ij} = b_i \forall i \; [\pi_j]$ 
- $x_{ij} \geq -u_{ij} \forall ij \; [\alpha_{ij}]$

Dual Formulation

- Max $\sum_{i}^{N} b_i \pi_i - \sum_{i}^{N} \sum_{j}^{N} u_{ij} \alpha_{ij}$
- s.t. $\pi_i - \pi_j - \alpha_{ij} \leq c_{ij} \forall (i, j) \; [x_{ij}]$
- $\alpha_{ij} \geq 0$
- $\pi_i$ unrestricted
**Transformed Dual Formulation**

\[
\begin{align*}
\text{max} & \quad \sum_{i}^{N} b_i \pi_i - \sum_{i}^{N} \sum_{j}^{N} u_{ij} \alpha_{ij} \\
\text{s.t.} & \quad \alpha_{ij} + c_{ij}^\pi \geq 0 \quad \forall (i, j) \quad [x_{ij}] \\
& \quad \alpha_{ij} \geq 0 \\
& \quad \pi_i \text{ unrestricted}
\end{align*}
\]

**Complementary Slackness Conditions**

\[
\begin{align*}
(u_{ij} - x_{ij}^*) \alpha_{ij}^* &= 0 \\
(\alpha_{ij}^* + c_{ij}^\pi) x_{ij}^* &= 0 \\
\text{if } c_{ij}^\pi > 0 \text{ then } \alpha_{ij}^* + c_{ij}^\pi > 0 \text{ then } x_{ij}^* &= 0 \\
\text{if } 0 < x_{ij}^* < u_{ij} \text{ then } \alpha_{ij}^* = 0 \text{ then } c_{ij}^\pi &= 0 \\
\text{if } c_{ij}^\pi < 0 \text{ then } \alpha_{ij}^* > 0 \text{ then } x_{ij}^* &= u_{ij}
\end{align*}
\]

**Ballou Ch7-4 Network Exercise**

**Network Exercise 2**
Network Flow References


Vehicle Routing Problems Overview

- Single Origin-Destination Routing
- Multiple Origin-Destination Routing
- Single Vehicle Round-Trip Routing
- Vehicle Routing and Scheduling

Single Round-Trip Vehicle Routing

- Traveling Salesman Problem (TSP)
- Specialized branch and bound algorithms
- 2000 nodes (million nodes)
- Many heuristic algorithms

TSP Applications

- Traveling salesman
- Shortest Hamiltonian cycle
- Knight’s tour
- Person-aboard order picking
- Running domestic errands
- Sequencing jobs in a paint booth
Person-Aboard Order Picking: Outside View

Person-Aboard Order Picking: Inside View

Random Sequence Tour

Local Improvement (3 Opt)
Asymmetric Traveling Salesman Problem Illustration

\[ c_{ij} \text{ cost of taking arc}(i, j) \]
\[ x_{ij} \in \{0, 1\} \text{ to take arc}(i, j) \text{ or not} \]

Asymmetric Traveling Salesman Problem Formulation

\[
\begin{align*}
\text{Min} & \quad \sum_{i=1}^{N} \sum_{j=1}^{M} c_{ij} x_{ij} \\
\text{s.t.} & \quad \sum_{i=1}^{N} x_{ij} = 1, \quad \forall j \\
& \quad \sum_{j=1}^{N} x_{ij} = 1, \quad \forall i \\
& \quad \sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1, \quad \forall S \subset N
\end{align*}
\]

Subtour Elimination Constraints

\[
\begin{align*}
\sum_{i \in S} \sum_{j \in S} x_{ij} & \leq |S| - 1, \quad \forall S \subset N \\
S = \{i, j, k\}, |S| = 3 \\
x_{ij} + x_{jk} + x_{ki} + x_{ji} + x_{kj} + x_{jk} & \leq 2
\end{align*}
\]

Symmetric Traveling Salesman Problem Illustration

\[ c_{ij} \text{ cost of taking edge}(i, j) \]
\[ x_{ij} \in \{0, 1\} \text{ to take edge}(i, j) \text{ or not} \]
**Symmetric Traveling Salesman Problem Formulation**

\[
\text{Min} \quad \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} c_{ij} x_{ij} \\
\text{s.t.} \quad \sum_{j=1}^{N} x_{ij} + \sum_{k=j+1}^{N} x_{jk} = 2 \quad \forall j \\
\sum_{i \in S} \sum_{j \in S, j > i} x_{ij} \leq |S| - 1 \quad \forall S \subset N
\]

**Simple TSP Heuristics**

- Create an initial tour
  - convex hull, sweep, nearest neighbor
- Insert remaining free points
  - nearest, cheapest, farthest insertion
- Improve existing tour
  - two, three, or Or (chain) exchanges

**TSP Example: Euclidean Distances**

\[
d_{ij}^E = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}
\]

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>608</td>
<td>0</td>
<td>361</td>
<td>412</td>
<td>825</td>
<td>762</td>
</tr>
<tr>
<td>3</td>
<td>566</td>
<td>361</td>
<td>0</td>
<td>316</td>
<td>500</td>
<td>640</td>
</tr>
<tr>
<td>4</td>
<td>860</td>
<td>412</td>
<td>316</td>
<td>0</td>
<td>361</td>
<td>361</td>
</tr>
<tr>
<td>5</td>
<td>985</td>
<td>825</td>
<td>500</td>
<td>500</td>
<td>0</td>
<td>510</td>
</tr>
<tr>
<td>6</td>
<td>1204</td>
<td>762</td>
<td>640</td>
<td>361</td>
<td>510</td>
<td>0</td>
</tr>
</tbody>
</table>
TSP Example: Point Locations

TSP Example: Nearest Neighbor

TSP Example: Nearest Neighbor Computations

- Start point 3
- Min \{566, 361, 316, 500, 640\} = 316 (4)
- Min \{860, 412, 500, 361\} = 361 (6)
- Min \{1204, 762, 510\} = 510 (5)
- Min \{985, 825\} = 825 (2)
- Min \{608\} = 608 (1)
- Total tour length = 3186

TSP Example: Sweep

- Find a centrally located rotation point
  - Center or average coordinates
- Rotate a ray and add points to the tour in the sequence they are traversed by the ray
TSP Example: Sweep

Rotation point (450,450)
Length = 3195

TSP Example: Quad Initial Tour

Length = 2699

TSP Example: Quad Calculations

- Leftmost 0 (1), bottom 0 (1), rightmost 900 (5), top 900 (6)
- In this case quadrilateral degenerates to triangle, tour {1-5-6-1}

TSP Example: Convex Hull Tour Skeleton

Length = 2865

Copyright © 1994-2003, Marc Goetschalckx, All rights reserved.
Insertion Heuristics

- Two decisions
  - Which free point to insert
  - Where to insert (which link to break)
  - Which decision to make first
- Many variants
  - Nearest Addition, Nearest Insertion
  - Cheapest Insertion, Priciest Insertion
  - Farthest Insertion, Min. Ratio,
    Max. Angle, Optimal Insertion...

Cheapest Insertion

- Insert free point with smallest cost increase (insertion penalty)

\[ \min_k \left\{ \min_{i,j} \left\{ \delta_{ijk} = c_{ik} + c_{kj} - c_{ij} \right\} \right\} \]

Priciest Insertion

- Insert free point with largest minimum cost increase

\[ \max_k \left\{ \min_{i,j} \left\{ \delta_{ijk} = c_{ik} + c_{kj} - c_{ij} \right\} \right\} \]

Nearest Insertion

- Find free point closest to a point on the tour

\[ \min_{k \in T, j \in T} \{ c_{kj} \} = \min_{k \in T} \left\{ \min_{j \in T} c_{kj} \right\} \]

- Insert on best link of the tour for point \( k \)

\[ \min_{(i,j) \in T} \left\{ \delta_{ijk} = c_{ik} + c_{kj} - c_{ij} \right\} \]
**Farthest Insertion**

- Find free point with maximum distance to closest point on the tour
  \[
  \max_{k \in T} \left\{ \min_{j \in T} c_{kj} \right\}
  \]
- Insert on best link of the tour for point \( k \)
  \[
  \min_{(i,j) \in T} \{ \delta_{ijk} = c_{ik} + c_{kj} - c_{ij} \}
  \]

**Nearest Addition**

- Find free point closest to a point on the tour
  \[
  \min_{k \notin T, j \in T} \{ c_{kj} \}
  \]
- Insert on best of two links out of point \( j \) for point \( k \)
  \[
  \min \left\{ \delta_{ijk} = c_{ik} + c_{kj} - c_{ij}, \delta_{jkm} = c_{jk} + c_{km} - c_{jm} \right\}
  \]

**TSP Example: Cheapest Insertion**

- Insert 4, 3
- Length = 2957

**TSP Example: Cheapest Insertion Computations**

- **Point 3 & 4**
  - \( \min (566+500-985=81 (1-5), 500+640-510=630 (5-6), 640+361-762=239 (6-2), 361+566-608=319 (2-1)) = 81 (1-5) \)
  - \( \min (860+500-985=375 (1-5), 500+361-510=351 (5-6), 361+412-762=11 (2-6), 412+860-608=644 (2-1)) = 11 (2-6) \)
- **Point 3**
  - \( \min (81 (1-5), 595 (6-4), 265 (4-2)) = 81 (1-5) \)
**TSP Example: Priciest Insertion Computations**

*point 3 & 4*
- \( \min \{566+500-985=81 \text{ (1-5)}, 500+640-510=630 \text{ (5-6)}, 640+361-762=239 \text{ (6-2)}, 361+566-608=319 \text{ (2-1)} \} = 81 \text{ (1-5)} *\)
- \( \min \{860+500-985=375 \text{ (1-5)}, 500+361-510=351 \text{ (5-6)}, 361+412-762=11 \text{ (6-2)}, 412+860-608=644 \text{ (2-1)} \} = 11 \text{ (2-6)} *\)

*point 4*
- \( \min \{610 \text{ (1-3)}, 316 \text{ (3-5)}, 11 \text{ (6-2)} \} = 11 \text{ (6-2)} *\)

**TSP Example: Nearest Insertion**

*points 3 & 4*
- \([3] \min \{566 \text{ (3-1)}, 361 \text{ (3-2)}, 500 \text{ (3-5)}, 640 \text{ (3-6)} \} = 361 \text{ (3-2)} * \text{-- tie}\)
- \([4] \min \{860 \text{ (4-1)}, 412 \text{ (4-2)}, 500 \text{ (4-5)}, 361 \text{ (4-6)} \} = 361 \text{ (4-6)} *\)
- \([3] \min \{566+500-985=81 \text{ (1-5)}, 500+640-510=630 \text{ (5-6)}, 640+361-762=239 \text{ (6-2)}, 361+566-608=319 \text{ (2-1)} \} = 81 \text{ (1-5)} *\)

**TSP Example: Nearest Insertion Continued**

*point 4*
- \([4] \min \{361 \text{ (4-6)}, 316 \text{ (4-3)} \} = 361 \text{ (4-3)} *\)
- \([4] \min \{860+316-566=610 \text{ (1-3)}, 316+500-500=316 \text{ (3-5)}, 500+361-510=351 \text{ (5-6)}, 361+412-762=11 \text{ (6-2)}, 412+860-608=644 \text{ (2-1)} \} = 11 \text{ (2-6)} *\)

**TSP Example: Farthest Insertion Computations**

*point 3 & 4*
- \([3] \min \{566 \text{ (3-1)}, 361 \text{ (3-2)}, 500 \text{ (3-5)}, 640 \text{ (3-6)} \} = 361 \text{ (3-2)} * \text{-- tie}\)
- \([4] \min \{860 \text{ (4-1)}, 412 \text{ (4-2)}, 500 \text{ (4-5)}, 361 \text{ (4-6)} \} = 361 \text{ (4-6)} *\)
- \([3] \min \{566+500-985=81 \text{ (1-5)}, 500+640-510=630 \text{ (5-6)}, 640+361-762=239 \text{ (6-2)}, 361+566-608=319 \text{ (2-1)} \} = 81 \text{ (1-5)} *\)
**TSP Example: Farthest Insertion Calculations Continued**

- **Point 4**
  - $\min \{361 (4-6), 316 (4-3)\} = 316 (4-3)$
  - $\min \{860+316-566=610 (1-3), 316+500-500=316 (3-5), 500+361-510=351 (5-6), 361+412-762=11 (2-6), 412+860-608=644 (2-1)\} = 11 (2-6)^*$

**Two Exchange Improvement Illustration**

**Three Exchange Improvement Illustration**

**Or (2 Chain) Exchange Improvement Illustration**
Exchange Improvement Variants

- First descent exchange
- Steepest descent exchange
- Simulated annealing
- Tabu search

Simulated Annealing

- Evaluate random exchange $\Delta$
- Execute exchange with probability
  
  \[
  \text{if } \Delta < 0 \quad P[\text{Exch}] = 1 \\
  \text{if } \Delta \geq 0 \quad P[\text{Exch}] = e^{-\Delta/T}
  \]
- $T$ search control parameter (temperature) systematically decreasing

TSP Example: Two Exchange Improvement

- Original tour length = 3186
- Crossing edges (3-4) and (2-5) in geometric TSP
- Exchange edges (3-4) and (2-5) with (2-4) and (3-5)
- Savings = 316 + 825 - 412 - 500 = 229
- Improved tour length = 2957
**Clarke and Wright Savings Illustration**

\[ s_{ij} = c_{i0} + c_{0j} - c_{ij} \]

**Clarke and Wright Savings**

- Select base point \{0\} (somewhere on perimeter or corner)
- Compute savings of combining tours
  \[ \max_j \{s_{ij} = c_{i0} + c_{0j} - c_{ij}\} \]
- Construct tour primitive \{0ij0\} (max savings)
- Append point with largest savings to either end of tour

**TSP Example: Clarke and Wright Savings Computations**

- Initial route primitive (base point 1):
  - \{(1-2-3-1) [2,3]\} = 608+566-361 = 813
  - \{(1-2-4-1) [2,4]\} = 608+860-412 = 1056
  - \{(1-4-6-1) [4,6]\} = 860+1204-361 = 1703 *
  - \{(1-5-6-1) [5,6]\} = 985+1204-510 = 1679

<table>
<thead>
<tr>
<th></th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>813</td>
<td>1056</td>
<td>768</td>
<td>1050</td>
</tr>
<tr>
<td>3</td>
<td>1110</td>
<td>1051</td>
<td>1130</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1345</td>
<td>1703</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>1679</td>
<td></td>
</tr>
</tbody>
</table>

- Tour primitive (1-6-4-1)
  - Max \{1050 (6-2-1), 1130 (6-3-1), 1679 (6-5-1), 1056 (4-2-1), 1110 (4-3-1), 1345 (4-5-1)\} = 1679 (6-5-1)
- Tour primitive (1-4-6-5-1)
  - Max \{1056 (1-2-4), 1110 (1-3-4), 768 (1-2-5), 1051 (1-3-5)\} = 1110 (1-3-4)
- Tour primitive (1-3-4-6-5-1)
  - Max \{813 (1-2-3), 768 (1-2-5)\} = 813 (1-2-3)
TSP Example:
Clarke & Wright Savings

- Base point = 1
- Length = 3141

TSP Example:
Spacefilling Curve

- Length = 3195

Christofides’ Heuristic

1. Construct minimum spanning tree
2. Construct minimum matching of odd degree nodes
3. Find maximum savings shortcut for nodes with degree four or higher

\[
\min_{k \mid \text{odd} \geq 2} \{ \delta_{ijk} = c_{ik} + c_{kj} - c_{ij} \}
\]

- Worst case performance ratio = 1.5

Spacefilling Curve Examples

- Serpentine Curve
- Hilbert Curve (19)
Christofides: Minimum Spanning Tree

Christofides: Minimum Matching and Max Shortcuts

- Minimum matching on \{1, 2, 5, 6\}
  - Min \((12)+(56)=1118, (15)+(26)=1747, (16)+(25)=2029\) = 1118 \((12)+(56)\)

- Maximum shortcuts for node \{3\}
  - Max \(\{265 (2-3-4), 81 (1-3-5), 36 (2-3-5), 22 (1-3-4)\}\) = 265 \((24)\)

Christofides: Minimum Spanning Tree and Matching

Christofides: Max Savings Shortcut
Symmetric Traveling Salesman Problem Formulation

\[
\text{Min } \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} c_{ij} x_{ij}
\]

s.t. \[\sum_{i=1}^{N-1} x_{ij} + \sum_{k=j+1}^{N} x_{jk} = 2 \quad \forall j \in \mathcal{N}\]
\[\sum_{i \in S} \sum_{j \in S, j > i} x_{ij} \leq |S|-1 \quad \forall S \subset \mathcal{N}\]

Lagrangean Relaxation of 1-Tree (Held & Karp)

1. Initialize node degree penalties \( \lambda \) to zero
2. Compute adjusted distances \( d_{ij}^\lambda = d_{ij} + \lambda_i + \lambda_j \)
3. Construct minimum spanning tree on \( \{N\} \)-base point
4. Connect base point with two cheapest edges to spanning tree

Lagrangean Relaxation of 1-Tree cont.

- If all node degrees = 2, stop
- else update node degree penalties (subgradient method)
  - if \( nd_i = 2 \) \( \lambda_i \) unchanged
  - if \( nd_i < 2 \) \( \lambda_i \) decreased
  - if \( nd_i \geq 2 \) \( \lambda_i \) increased
- Go to step 2

1-Tree: First Minimum Spanning Tree (Base Point = 1)
**1-Tree: First 1-Tree (Base Point = 1)**

**1-Tree: Second 1-Tree (Base Point = 1)**

**TSP Example: Tours Illustration**

**Tours Illustration: Test 125**
Vehicle Routing Problems Overview

- Single Origin-Destination Routing
- Multiple Origin-Destination Routing
- Single Vehicle Round-Trip Routing
- Vehicle Routing and Scheduling

Traveling Salesman Problem References

**Vehicle Routing Problem**

- Goal is to efficiently use a fleet of vehicles
- Given a number of stops to pick up or deliver passenger or goods
- Under a variety of constraints
  - Vehicle capacity
  - Delivery time restrictions
  - Precedence constraints

**Vehicle Routing Decisions**

- Which customers served by which vehicle (allocation or clustering)
- Sequence of stops for each vehicle (sequencing)
- Number of vehicles (fleet planning)

**Vehicle Routing Variants**

- Traveling salesman problem
- Pure vehicle routing
- Linehaul-backhaul
- Vehicle routing with time windows
- Vehicle routing and scheduling
- Mixed pickup and delivery

**Vehicle Routing Algorithms Classification**

- Route generating versus route selecting
- Basic methodology
Vehicle Routing Algorithms
Classification by Required Input

- Route generating
  - Large variety
  - Tightly capacitated
- Route selecting
  - Set partitioning algorithm (SPP)
  - Complex costs and constraints

Vehicle Routing Algorithms
Classification by Methodology

- VRP problem based
  - Sweep, savings, exchange, nearest neighbor
- Optimization based
  - GAP, SPP, k-trees
  - Simulated annealing, tabu search
- Artificial intelligence based
  - Genetic search

Pure Vehicle Routing – VRP Overview

- Problem definition
  - Mathematical formulation
  - Optimal algorithms
- Heuristic algorithms
  - Conclusions

VRP Problem Definition

- Single depot
  - N customers \((x_i, y_i, \text{dem}_i)\)
    - Known location and demand
- K vehicles \((\text{cap}_k)\)
  - Known and equal size
- Minimum travel cost objective
  - Travel distance norm
Vehicle Routing Example: Distance Data

```
2
3 4
3 2
2 2
1

Truck capacity = 15
```

Vehicle Routing Example: Demand Data

```
<table>
<thead>
<tr>
<th>Customer</th>
<th>Demand</th>
<th>Distance to Depot</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>1</td>
</tr>
</tbody>
</table>

Truck capacity = 15
```

VRP Route Generating Algorithms

- Route construction
  - Clarke and Wright, nearest neighbor
- Route improvement
  - Exchange improvements
- Two phase algorithms
  - Cluster first, route second
    - Sweep A, Fisher and Jaikumar (GAP)
  - Route first, cluster second
    - Sweep B, great tour

Nearest Neighbor

1. Start a new route with the depot
2. Find nearest unvisited customer
3. If customer demand is less than remaining vehicle capacity, append customer, otherwise go to 5
4. If all customers are visited stop
5. If maximum number vehicles is used stop, else go to 1
Sweep Algorithm Variants

- Cluster first, route second sweep (variant A)
  - Ray determines clusters
  - TSP construction routines to route each cluster
- Route first sweep (variant B)
  - Ray determines routes
  - TSP improvement routines for each initial route

Vehicle Routing Example: Distance Data

Vehicle Routing Example: Demand Data

<table>
<thead>
<tr>
<th>Customer</th>
<th>Demand</th>
<th>Distance to Depot</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>1</td>
</tr>
</tbody>
</table>

Truck capacity = 15

Number of Routes

- Maximum number of routes usually parameter (for example max = 3)
- Minimum number of routes

\[
\min \# routes = \left\lceil \frac{\sum_{i=1}^{N} q_i}{\text{Cap}} \right\rceil = \left\lceil \frac{26}{15} \right\rceil = \left\lceil 1.73 \right\rceil = 2
\]
Nearest Neighbor

- Customer 5, distance 1, truck load 6
- Customer 4, distance 2, truck load 13
- Customer 1, distance 2, load 16 > 15
  route length = 5, new route
- Customer 3, distance 2, truck load 4
- Customer 2, distance 3, truck load 10
- Customer 1, distance 5, truck load 13
  route length = 14, total length = 19

Cluster-First Sweep (Variant A)

- Start direction = east
  \[ D - 5[6] - 1[9] - D \quad L_1 = 8 \]
  \[ D - 3[4] - D \quad L_3 = 4 \]
  \[ L = \sum_r L_r = 8 + 11 + 4 = 23 \]
  Note initial groupings are clusters only
  without point sequence

Cluster-First Sweep (Variant A)

- Start direction = north
  \[ D - 2[6] - 3[10] - D \quad L_1 = 10 \]
  \[ D - 4[7] - D \quad L_3 = 4 \]
  \[ L = \sum_r L_r = 10 + 8 + 4 = 22 \]
  Note initial groupings are clusters only
  without point sequence

Route-First Sweep (Variant B)

- Start direction = east
  \[ D - 5 - 1 - 4 - 2 - 3 - D \quad (TSP) \]
  \[ D - 5[6] - 1[9] - D \quad L_1 = 8 \]
  \[ D - 3[4] - D \quad L_3 = 4 \]
  \[ L = \sum_r L_r = 8 + 11 + 4 = 23 \]
Clarke and Wright Savings

- Maximum number of routes = K
- Serial variant
  - One route at-a-time
  - Simpler implementation
- Parallel variant
  - No more than K routes at-a-time
  - More complex programming

Clarke and Wright Savings
Initial Savings (Common)

- Compute savings for every feasible pair of points
  \[ q_1 + q_2 = 3 + 6 = 9 \leq 15 \]
  \[ s_{12} = d_{10} \pm d_{02} - d_{12} = 4 + 5 - 5 = 4 \]
  \[ q_1 + q_4 = 3 + 7 = 10 \leq 15 \]
  \[ s_{14} = d_{40} \pm d_{01} - d_{14} = 4 + 2 - 2 = 4 \]

Clarke and Wright Savings
Initial Savings Matrix

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Selected pair (1-2) with ties broken by first encountered maximum savings

Serial Clarke and Wright Savings: Second Iteration

- Copy corresponding savings rows and columns, eliminating infeasible combinations
- \((4-1-2),(1-2-4) = 3+6+7=16 > 15\)
Serial Clarke and Wright Savings: Third Iteration

- Capacity infeasibilities eliminate all positive savings
- No feasible or profitable extension of the route, so start a new route

Parallel Clarke and Wright Savings: Second Iteration

- Maximum two routes
- Copy corresponding rows and columns, eliminating infeasible combinations

Serial Clarke and Wright Savings: Third Iteration

- Eliminate all visited point rows and columns from the savings matrix and restart

Parallel Clarke and Wright Savings: Third Iteration

- All partial routes remain “extendable”

\[ L = I_{(1-2-3)} + I_{(4-5)} = 14 + 5 = 19 \]
**Improvement Algorithms**

- **Intra-route improvements (TSP)**
  - Always feasible
  - 2 exchange, chain exchange, 3 exchange

- **Inter-route improvements**
  - Test and make only feasible exchanges
  - Move (one point to another route)
  - Swap (exchange two points between two routes)

---

**VRP GAP Formulation: Clustering**

\[
\begin{align*}
\text{min} \quad & \sum_{k} f_k(y_{ik}) \\
\text{s.t.} \quad & \sum_{i=1}^{N} \text{dem}_i y_{ik} \leq \text{cap}_k \quad \forall k \\
& \sum_{k} y_{0k} = K \\
& \sum_{k} y_{ik} = 1 \quad i = 1..N \\
& y_{ik} \in \{0,1\}
\end{align*}
\]

---

**VRP GAP Formulation: Routing**

\[
\begin{align*}
f(y_{ik}) &= \min \sum \sum c_{ij} x_{ijk} \\
\text{s.t.} \quad & \sum_i x_{ijk} = y_{jk} \quad \forall jk \\
& \sum_j x_{ijk} = y_{ik} \quad \forall ik \\
& \sum \sum x_{ijk} \leq |S| - 1 \quad S \subseteq N(y_{ik}) \\
& x_{ijk} \in \{0,1\}
\end{align*}
\]
GAP Characteristics

- Alternative generating algorithm
- Predominant importance of capacity constraints
- Mathematical programming based algorithm (requires solver)
- Strongly capacitated or strongly combinatorial problems

Set Partitioning Problem (SPP) Definition

- Every column \( j \) = feasible alternative action
- Every row \( i \) = service request
- Minimize overall cost while servicing all requests

Set Partitioning Notation

- \( a_{ij} = 1 \) if alternative server \( j \) satisfies customer request \( i \)
- \( c_j \) = cost of alternative \( j \)
- \( p_i \) = cost estimate for servicing customer request \( i \)
- \( x_j = 1 \) if server \( j \) is selected (enabled)

Set Partitioning Formulation

\[
\begin{align*}
\min & \sum_{j=1}^{N} c_j x_j \\
\text{s.t.} & \sum_{j=1}^{N} a_{ij} x_j = 1 & i = 1..M \\
& x_j \in \{0,1\}
\end{align*}
\]
Set Partitioning Characteristics

- Alternative selecting algorithm
- Accurate costs and feasibility constraints
- Optimal solution for “small” problem sizes (IP solver)
- Efficient column generation and pricing
- Complex problems

Set Partitioning Algorithm

1. Start with a feasible partition \( J^* \)
2. Determine row prices by allocating column prices “equitable” such that

\[
 c^+_j = \sum_{i=1}^{M} a_{ij} p_i
\]

VRP example: \( p_i = \frac{\text{dem}_i \cdot \text{d}_{ij}}{\sum_{i \in J_i} \text{dem}_i \cdot \text{d}_{ij}} c^+_j \)

Set Partitioning Algorithm (2)

- Generate and evaluate new column \( j \), if

\[
 c_j - \sum_{i=1}^{M} a_{ij} p_i \leq 0
\]

add column \( j \) to the partitioning problem
- If enough new columns are added, solve the partitioning problem else go to step 3

Set Partitioning Algorithm (3)

- If all columns have been evaluated or solution is within tolerance, stop else go to step 2
Set Partitioning Example: Distance Data

Truck Capacity = 15

Set Partitioning Example: Demand Data

<table>
<thead>
<tr>
<th>Customer</th>
<th>Demand</th>
<th>Distance to Depot</th>
<th>Distance x Demand</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>5</td>
<td>30</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>2</td>
<td>14</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>1</td>
<td>6</td>
</tr>
</tbody>
</table>

Truck Capacity = 15

Set Partitioning Example: One-Stop Routes Excel Data

Set Partitioning Example: One-Stop Routes Excel Solver
Set Partition Example: One-Stop Routes Excel Solver Options

Excel Solver Options

Set Partition Example: One-Stop Routes Excel Solution

Set Partitioning Example: Two Customer Routes

<table>
<thead>
<tr>
<th>Route</th>
<th>Customer One</th>
<th>Customer Two</th>
<th>Length</th>
<th>Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>1</td>
<td>2</td>
<td>14</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>3</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>4</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>5</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>3</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>4</td>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>5</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>14</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>

Set Partition Example: Two Customer Routes Excel Data

Set Partition Example: Two Customer Routes Excel Solution
Set Partition Example: Two Customer Routes Excel Solver

Set Partition Example: Two Customer Routes Excel Solution

Set Partitioning Example: Two Customer Row Prices

\[ j = 5 \quad I_5 = \{5\} \quad c_5 = p_5 = 2 \]

\[ j = 8 \quad I_8 = \{1,4\} \quad j = 10 \quad I_{10} = \{2,3\} \]

\[ c_j = 8 \quad p_1 + p_4 = 8 \quad c_j = 10 \quad p_2 + p_3 = 10 \]

\[ p_1 = \frac{12}{26} \cdot 8 = 0.46 \cdot 8 = 3.69 \quad p_2 = \frac{30}{38} \cdot 10 = 0.79 \cdot 10 = 7.89 \]

\[ p_4 = \frac{14}{26} \cdot 8 = 0.54 \cdot 8 = 4.31 \quad p_3 = \frac{8}{38} \cdot 10 = 0.21 \cdot 10 = 2.11 \]

Set Partitioning Example: Three Customer Routes

<table>
<thead>
<tr>
<th>Route</th>
<th>Customer One</th>
<th>Customer Two</th>
<th>Customer Three</th>
<th>Route Demand</th>
<th>Route Length</th>
<th>Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>13</td>
<td>14</td>
<td>-0.31</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>13</td>
<td>16</td>
<td>-2.31</td>
</tr>
<tr>
<td>18</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>13</td>
<td>16</td>
<td>-2.31</td>
</tr>
<tr>
<td>19</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>14</td>
<td>16</td>
<td>-2.42</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>14</td>
<td>18</td>
<td>-4.42</td>
</tr>
<tr>
<td>21</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>14</td>
<td>14</td>
<td>-0.42</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>14</td>
<td>12</td>
<td>-1.89</td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>14</td>
<td>10</td>
<td>0.11</td>
</tr>
<tr>
<td>24</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>14</td>
<td>10</td>
<td>0.11</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>13</td>
<td>12</td>
<td>-4.20</td>
</tr>
<tr>
<td>26</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>13</td>
<td>12</td>
<td>-4.20</td>
</tr>
<tr>
<td>27</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>13</td>
<td>10</td>
<td>-2.20</td>
</tr>
</tbody>
</table>

Copyright © 1994-2003, Marc Goetschalckx, All rights reserved.
Set Partition Example: Three-Stop Routes Excel Data

Set Partition Example: Three-Stop Routes Excel Solver

Set Partition Example: Three-Stop Routes Excel Solution

Set Partitioning References


Linehaul-Backhaul Problem - VRPB

- #1 savings technique in routing
- Problem definition
- Mathematical formulation
- Heuristics algorithms
- Software
- Conclusions

VRPB Problem Definition

- Single depot
- $N_c$ customers $(x_i, y_i, \text{dem})$ and $N_s$ suppliers $(x_i, y_i, \text{sup})$
- $M$ equal size vehicles (cap)
- Rear loaded vehicles
  - all customers before any supplier
- Minimize total travel distance
- Travel distance norm

VRPB Route Illustration

Lineback Illustration: 100 x 100 x 10
**Vehicle Routing with Time Windows VRPTW**

- Time window is a restriction on the time of visiting the customer
- Types of time windows
  - Exclusion versus mandatory
  - Single versus multiple
  - Hard versus soft

**VRPTW Characteristics**

- Much more difficult problem than classical VRP
  - No simple algorithms
  - No easy feasible test
  - No easy human intuition
  - No intuitive route structure
- Solution procedures use mathematical programming or stochastic search
Vehicle Routing References