Precept 4: Traveling Salesman Problem, Hierarchical Clustering

Qian Zhu
2/23/2011
Agenda

• Assignment: Traveling salesman problem
• Hierarchical clustering
  – Example
  – Comparisons with K-means
TSP

- TSP: Given the coordinates for a set of vertices,
- Find an order of connecting the vertices, such that 1) each vertex is visited once (except for the start vertex which is visited twice), and 2) the total distance is minimum. Start vertex and end vertex must be the same.
Can you spot a TSP tour among these possible answers?
TSP

• Smaller problems (n=50): humans can do very well.
• Larger problems: not so much...
• Machine: there is no general way to solve TSP. Computationally hard.
  – Challenge: find a polynomial time algorithm to solve this problem, and you will win a million bucks!
A naive algorithm to find a TSP tour

- Exhaustive search among all possible tours, and pick the minimal tour.
- How many possibilities?
  - $N$ cities
  - $N \times (N-1) \times (N-2) \times \ldots \times 1$
  - $N!$
- Obviously not very efficient.
TSP

• Let’s say no polynomial time algorithm exists to solve this problem.
• We can use *heuristic* methods to derive a reasonably good answer in reasonable time.
  – Heuristics: no guarantee that answer is optimal
  – Modern methods can solve much larger TSP (involving millions of nodes) in reasonable time.
Assignment goals

• Use heuristic methods to estimate answer for a computationally hard problem
• Visualize the answer to see how well the heuristics perform.
• Implement a linked list structure to represent a tour in Java.
  – You cannot use LinkedList from Java standard library.
• We will use two heuristics:
  • Nearest neighbor
  • Minimize total tour distance
  – Extra credit: implement your own heuristic
Tour representation

- **Circular** linked list

```java
private class Node{
    Point P;
    Node next;
}
```

- Inner class: better encapsulation
Tour traversal

• Start from the head node
• Iterate through the list until the head node is reached again.
Adding a node to the list

Add: Node C

To: Node A

1. Insert here

2.
Nearest neighbor (NN)

- Read in the next point, and add it to the current tour after the point to which it is closest.

\[
\begin{align*}
t_{\text{list}} &= \text{list of points in current tour} \\
p_{\text{list}} &= \text{list of points} \\
&\text{for each } p \text{ in } p_{\text{list}}:
\quad x &= \text{closest point in } t_{\text{list}} \\
&\text{add } p \text{ to } t_{\text{list}}, \text{ after } x
\end{align*}
\]

Compare E with each node in the tour. Pick the node that is closest to E. Connect E after this node.
Nearest neighbor (NN)

Current tour

Let’s say B is closest to E. Add E after E:
Smallest increase heuristic

• Read in the next point, add it to the current tour after the point where it results in the least possible increase in tour length.

\[
\begin{align*}
t\text{\_list} &= \text{list of points in current tour} \\
p\text{\_list} &= \text{list of points} \\
\text{for each } p \text{ in } p\text{\_list:} \\
&\quad x = \text{point where if } p \text{ is inserted after } x, \text{ smallest increase in tour length} \\
&\quad \text{add } p \text{ to } t\text{\_list, after } x
\end{align*}
\]

Need to calculate every possible increase in order to know which is smallest.
How to calculate increase in tour length?

Increase in tour length = (4+4) – 5 = 3
Assume...

• The order of inserting nodes is given. (use the order of nodes in the input file)

• Does order matter?
  – Try randomize the order and note whether you get a better answer.
Other problems similar to TSP

• Easy problems:
  – Shortest path problem:
    • Assume: vertices and edges are known
    • Given two vertices. Find the shortest connected path between them.
    • Algorithm: Dijkstra (O(|V|^2))
    • Application: route planning

  – Chinese postman problem:
    • A mailman wants to deliver mail to a certain neighborhood. The mailman is lazy, and he wants to find the shortest route through neighborhood, that meets these criteria:
      – It is a closed circuit
      – He needs to go through every street at least once
    • Algorithm: Edmond (O(|V|^3))
    • Application: UPS delivery route planning
Other problems similar to TSP

- Easy problems:
  - Minimum spanning tree problem
    - Given: a connected graph G with weighted edges.
    - Find: a connected subgraph that touches every vertex in the graph. The sum of weight of edges must be minimal.
    - Algorithm: Prim and Kruskal (O(|E| \log |E|))
    - Application: A cable TV company is laying cable to a new neighborhood. It is constrained to bury cable only along certain paths. Find a spanning tree that is a subset of those paths that connects every house and incur minimum laying cost.
Other problems similar to TSP

• Easy problems (continued):
  – Euler Trail problem:
    Seven bridges of Konigsberg -

Take a walk through the city that would cross each bridge once and only once.

Is there a solution?
Other problems similar to TSP

• Euler Trail problem (continued):
  – Let’s convert it to a graph problem:
    • Define a trail: a path that visits every edge exactly once.
    • **Open** trail: a trail where start \(!=\) end.
    • **Closed** trail: a trail where start = end.
    • Find an open trail or a closed trail in the graph.

Can you find an Euler trail in Konigsberg?
Other problems similar to TSP

• Euler Trail problem (continued):
  – Modern day Konigsberg (after bombing during WWII destroyed two bridges):
Other problems similar to TSP

• Hard problems:
  – Find a Hamiltonian circuit (a path that visits every vertex in the graph exactly once and return to starting vertex)
Hierarchical clustering

![Hierarchical clustering plot]

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.62</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.42</td>
<td>0.19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1.32</td>
<td>0.81</td>
<td>0.93</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1.02</td>
<td>0.48</td>
<td>0.61</td>
<td>0.33</td>
</tr>
</tbody>
</table>
Hierarchical clustering (single-linkage)

### Iteration 1

<table>
<thead>
<tr>
<th>node</th>
<th>node_min</th>
<th>dist_min</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>0.42</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0.19</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0.19</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>0.33</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>0.33</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>node</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.62</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.42</td>
<td>0.19</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1.32</td>
<td>0.81</td>
<td>0.93</td>
</tr>
<tr>
<td>5</td>
<td>1.02</td>
<td>0.48</td>
<td>0.61</td>
</tr>
</tbody>
</table>

Join 2 and 3

<table>
<thead>
<tr>
<th>node</th>
<th>2,3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>2,3</td>
<td>0.42</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1.32</td>
<td>0.81</td>
</tr>
<tr>
<td>5</td>
<td>1.02</td>
<td>0.48</td>
</tr>
</tbody>
</table>
### Iteration 2

#### Node Distances

<table>
<thead>
<tr>
<th>Node</th>
<th>1</th>
<th>2,3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>2,3</td>
<td>0.42</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1.32</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1.02</td>
<td>0.48</td>
<td>0.33</td>
</tr>
</tbody>
</table>

#### Node Min Distances

<table>
<thead>
<tr>
<th>Node</th>
<th>Node$_{min}$</th>
<th>Dist$_{min}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2,3</td>
<td>0.42</td>
</tr>
<tr>
<td>2,3</td>
<td>1</td>
<td>0.42</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>0.33</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>0.33</td>
</tr>
</tbody>
</table>

#### Join 4 and 5

<table>
<thead>
<tr>
<th>1</th>
<th>2,3</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2,3</td>
<td>0.42</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1.32 0.81</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1.02 0.48 0.33</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Join 1 and 2,3
Join 1,2,3 and 4,5

Final Tree:

\[
\{ \{2, 3\}, 1\}, \{4, 5\} \}
\]

Note that the height corresponds to the minimum distance at each iteration.
Cluster these points using single-linkage method, using complete-linkage method.
Answers

Cluster Dendrogram

Complete linkage

Cluster Dendrogram

Single linkage
Hierarchical vs K-means

• Run-time analysis. Which clustering algorithm runs faster?
• What’s the bottleneck step in hierarchical clustering?
• What’s the disadvantage of K-means?
• What’s the disadvantage of hierarchical clustering?