DM872

Mathematical Optimization at Work

TSP practice

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Outline

Traveling Salesman Problem

https://www.math.uwaterloo.ca/tsp/

Outline

Dynamic Programming

- Dynamic Programming (DP) is a technique to solve combinatorial optimization problems with applications, for example, in mathematical programming, optimal control, and economics

- DP is somehow related to branch-and-bound as it performs an intelligent enumeration of the feasible solutions of the problem considered

- Principle of Optimality (known as Bellman Optimality Conditions): Suppose that the solution of a problem is the result of a sequence of n decisions $D_1, D_2, \dots, D_n$; if a given sequence is optimal, then the first k decisions must be optimal, but also the last $n - k$ decisions must be optimal

- DP breaks down the problem into stages, at which decisions take place, and find a recurrence relation that relates each stage with the previous one

Principle of Optimality

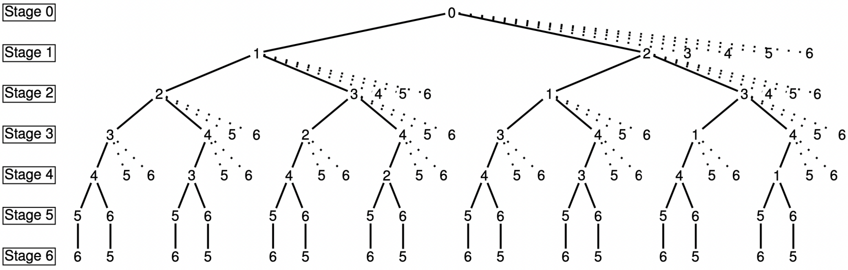
The TSP asks for the shortest tour that starts from 0, visits all cities of the set $C = \{1, 2, ..., n\}$ exactly once, and returns to 0, where the cost to travel from i to j is c_{ij} (with $(i, j) \in A$)
If the optimal solution of a TSP with six cities is $(0, 1, 3, 2, 4, 6, 5, 0)$, then...

- the optimal solution to visit $\{1, 2, 3, 4, 5, 6\}$ starting from 0 and ending at 5 is $(0, 1, 3, 2, 4, 6, 5)$
- the optimal solution to visit $\{1, 2, 3, 4, 6\}$ starting from 0 and ending at 6 is $(0, 1, 3, 2, 4, 6)$
- the optimal solution to visit $\{1, 2, 3, 4\}$ starting from 0 and ending at 4 is $(0, 1, 3, 2, 4)$
- the optimal solution to visit $\{1, 2, 3\}$ starting from 0 and ending at 2 is $(0, 1, 3, 2)$
- the optimal solution to visit $\{1, 3\}$ starting from 0 and ending at 3 is $(0, 1, 3)$
- the optimal solution to visit 1 starting from 0 is $(0, 1)$

$\rightsquigarrow$ The optimal solution is made up of a number of optimal solutions of smaller subproblems

Enumerate All Solutions of the TSP

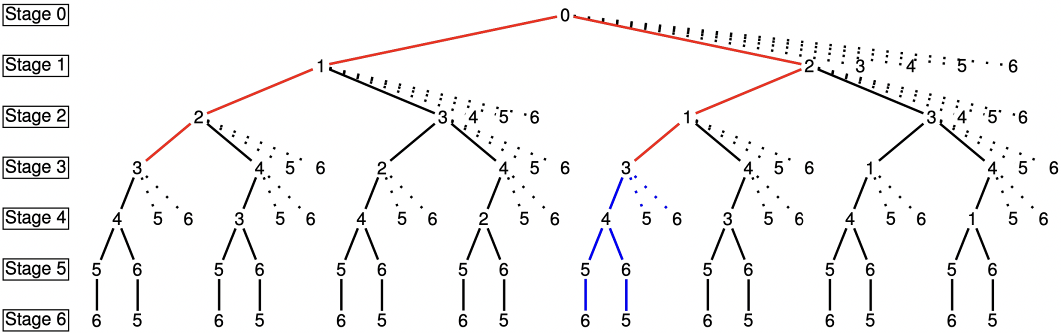
- A solution of a TSP with n cities derives from a sequence of n decisions, where the kth decision consists of choosing the kth city to visit in the tour



- The number of nodes (or states) grows exponentially with n
- At stage k, the number of states is $\binom{n}{k}k!$
- With $n = 6$, at stage $k = 6$, 720 states are necessary

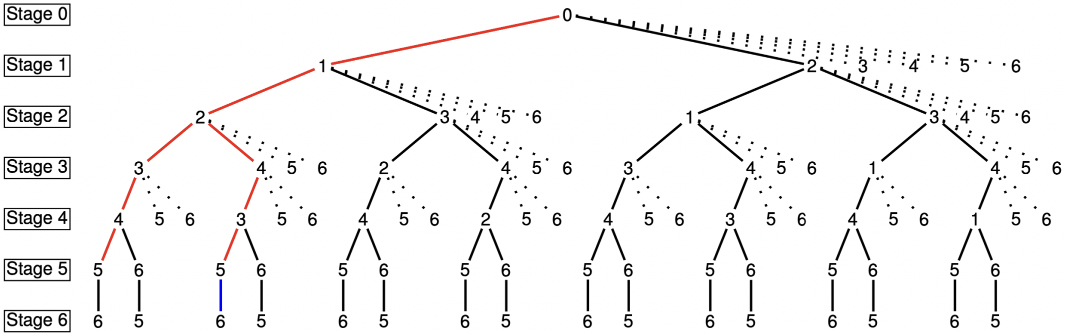
⇝ DP finds the optimal solution by implicitly enumerating all states but actually generating only some of them

Are All States Necessary?



If path $(0, 1, 2, 3)$ costs less than $(0, 2, 1, 3)$, the optimal solution cannot be found in the blue part of the tree

Are All States Necessary?



If path $(0, 1, 2, 3, 4, 5)$ costs less than $(0, 1, 2, 4, 3, 5)$, the optimal solution cannot be found in the blue part of the tree

Are All States Necessary?

- At stage k $(1 \leq k \leq n)$, for each subset of cities $S \subseteq C$ of cardinality k, it is necessary to have only k states (one for each of the cities of the set S)

- At state $k = 3$, given the subset of cities $S = \{1, 2, 3\}$, three states are needed:
  - the shortest-path to visit S by starting from 0 and ending at 1
  - the shortest-path to visit S by starting from 0 and ending at 2
  - the shortest-path to visit S by starting from 0 and ending at 3

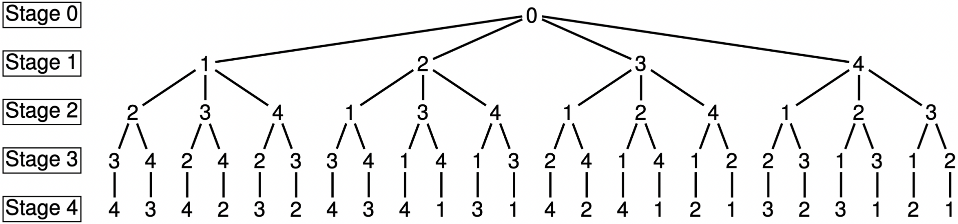
- At stage k, $\binom{n}{k} k$ states are required to compute the optimal solution (not $\binom{n}{k} k!$)

#States n = 6

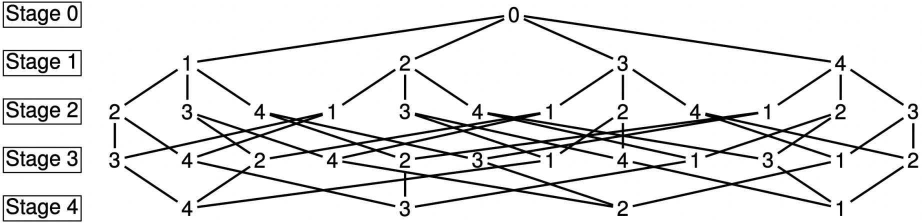
Stage	$\binom{n}{k} k!$	$\binom{n}{k} k$
1	6	6
2	30	30
3	120	60
4	360	60
5	720	30
6	720	6

Complete Trees with n=4

Enumeration of all paths



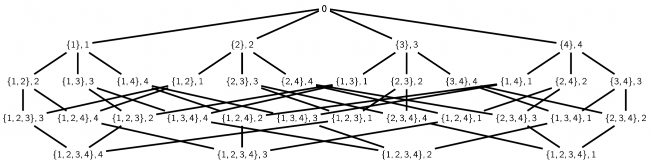
Implicit enumeration of all paths



Dynamic Programming Recursion for the TSP I

- Given a subset $S \subseteq C$ of cities and $k \in S$, let $f(S, k)$ be the optimal cost of starting from 0, visiting all cities in S, and ending at k
- Begin by finding $f(S, k)$ for $|S| = 1$, which is $f(\{k\}, k) = c_{0k}, \forall k \in C$
- To compute $f(S, k)$ for $|S| > 1$, the best way to visit all cities of S by starting from 0 and ending at k is to consider all $j \in S \setminus \{k\}$ immediately before k, and look up $f(S \setminus \{k\}, j)$, namely

$$f(S, k) = \min_{j \in S \setminus \{k\}} \{f(S \setminus \{k\}, j) + c_{jk}\}$$



- The optimal solution cost z^* of the TSP is $z^* = \min_{k \in C}\{f(C, k) + c_{k0}\}$

Dynamic Programming Recursion for the TSP II

DP Recursion from [Held and Karp (1962)]

1. Initialization. Set $f(\{k\}, k) = c_{0k}$ for each $k \in C$

2. RecursiveStep. For each stage $r = 2, 3, ..., n$, compute

$$f(S, k) = \min_{j \in S \setminus \{k\}} \{f(S \setminus \{k\}, j) + c_{jk}\} \forall S \subseteq C : |S| = r \text{ and } \forall k \in S$$

3. Optimal Solution. Find the optimal solution cost z^* as

$$z^* = \min_{k \in C} \{f(C, k) + c_{k0}\}$$

- With the DP recursion, TSP instances with up to 25 - 30 customers can be solved to optimality; other solution techniques (i.e., branch-and-cut) are able to solve TSP instances with up to... 85900 customers
- Nonetheless, DP recursions represents the state-of-the-art solution techniques to solve a wide variety of PDPs

Outline

Dantzig, Fulkerson and Johnson (DFJ) Formulation

- Find the cheapest movement for a drilling, welding, drawing, soldering arm as, for example, in a printed circuit board manufacturing process or car manufacturing process

- n locations, asymmetric c_{ij} cost of travel,

Variables:

$$x_{ij} \in \{0,1\} \qquad \forall i, j \in V, i \neq j$$

Objective:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

Constraints:

- visit all vertices

$$\sum_{j:j\neq i} x_{ij} = 1 \qquad \forall i = 1, \ldots, n$$

$$\sum_{i:i\neq j} x_{ij} = 1 \qquad \forall j = 1, \ldots, n$$

- cut set constraints

$$\sum_{i\in S} \sum_{j\notin S} x_{ij} \geq 1 \qquad \forall S \subset N, S \neq \emptyset$$

- subtour elimination constraints

$$\sum_{i\in S} \sum_{j\in S} x_{ij} \leq |S| - 1 \qquad \forall S \subset N, 2 \leq |S| \leq n - 1$$

Miller, Tucker, Zemling (MTZ) Formulation

$$\min \sum_{(ij)\in A} c_{ij} x_{ij} \tag{1}$$

$$\sum_{i:i\neq j} x_{ij} = 1 \qquad \forall j = 1, \ldots, n \tag{2}$$

$$\sum_{j:i\neq j} x_{ij} = 1 \qquad \forall i = 1, \ldots, n \tag{3}$$

$$u_i - u_j + n x_{ij} \leq n - 1, \qquad \forall i, j = 2, 3, \ldots, n, i \neq j \tag{4}$$

$$x_{ij} \in \mathbb{B} \qquad \forall ij \in A \tag{5}$$

$$u_i \in \mathbb{R} \qquad \forall i = 1, \ldots, n \tag{6}$$

Gavish-Graves (GG) Formulation

Single commodity flow. $g_{ij} \in \mathbb{R}^+$ sequence variables (is 0 if $x_{ij} = 0$ otherwise it indicates the number of arcs included on the path from vertex 1 up to arc (i, j))

$$\min \sum_{(ij) \in A} c_{ij} x_{ij} \tag{7}$$

$$\sum_{i: i \neq j} x_{ij} = 1 \qquad \forall j = 1, \ldots, n \tag{8}$$

$$\sum_{j: i \neq j} x_{ij} = 1 \qquad \forall i = 1, \ldots, n \tag{9}$$

$$\sum_{j=1}^{n} g_{ji} - \sum_{j=2}^{n} g_{ij} = 1 \qquad \forall i = 2..n \tag{10}$$

$$g_{ij} \leq (n-1) x_{ij} \qquad \forall ij \in A \tag{11}$$

$$x_{ij} \in \mathbb{B} \qquad \forall ij \in A \tag{12}$$

$$g_{ij} \in \mathbb{R}^+ \qquad \forall ij \in A \tag{13}$$

Svestka (S) Formulation

- similar to precedent, also a single commodity flow formulation
- y_{ij}: flow from city i to city j
- f: gain in flow from city i to city j

$$\min \sum_{ij \in A} c_{ij} x_{ij} \tag{14}$$

$$\sum_{j:ji \in A} y_{ji} \geq 1 \qquad \forall i = 2, \ldots, n \tag{15}$$

$$\sum_{j:ij \in A} y_{ij} - \sum_{j:ji \in A} y_{ji} = f \qquad \forall i = 1, \ldots, n \tag{16}$$

$$\sum_{ij \in A} x_{ij} \leq n \tag{17}$$

$$y_{ij} \leq (1 + n\,f) x_{ij} \qquad \forall ij \in A \tag{18}$$

$$x_{ij} \in \mathbb{B} \qquad \forall ij \in A \tag{19}$$

$$y_{ij} \in \mathbb{R}^+ \qquad \forall ij \in A \tag{20}$$

Dantzig (D) Formulation

- Indices: i, jk for cities, t for step
- $x_{ijt} = 1$ if we drive from city i to city j at step t, else 0.

$$\min \sum_{ij \in A} \sum_{t} c_{ij} x_{ijt} \tag{21}$$

$$\sum_{i} x_{ijt} - \sum_{k} x_{j,k,t+1} = 0 \qquad \forall j \text{ and } t = 1, \ldots, n \tag{22}$$

$$\sum_{j} \sum_{t} x_{ijt} = 1 \qquad \forall i = 1, \ldots, n \tag{23}$$

$$x_{ijt} \in \mathbb{B} \qquad \forall ij \in A, t \tag{24}$$

Comparison

Dual bounds

Instance	DFJ	MTZ	Svestka	Dantzig
ran20points	3182.2	2538.8	1087.7	2504.1
dantzig42.dat		2538.8	1032.8	2504.2
berlin52.dat				
bier127.dat				

Comparing LP relaxations

Source: Oncan, Altinel, Laporte, A comparative analysis of several asymmetric traveling salesman problem formulations (2009)

By the way... What does this mean?

Best algorithms are based on this model.

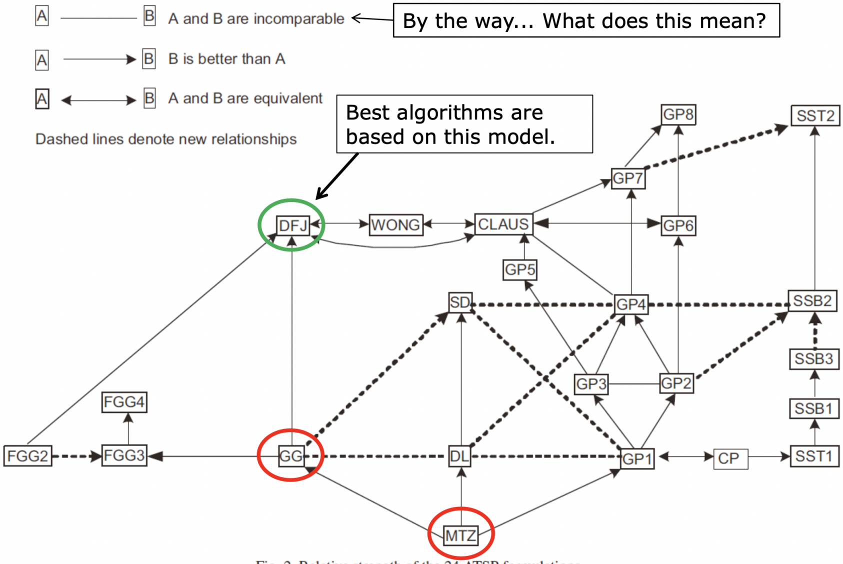


Fig. 2. Relative strength of the 24 ATSP formulations.

Symmetric DFJ

- $E = \{i, j \mid i \in V, j \in V, i < j\}$

$$
\begin{aligned}
\text{(TSPIP)} \quad & \min \sum c_{ij} x_{ij} \\
& \text{s.t.} \sum_{ij \in \delta(i)} x_{ij} + \sum_{ji \in \delta(i)} x_{ji} = 2 \text{ for all } i \in V \\
& \sum_{ij \in E(S)} x_{ij} \leq |S| - 1 \text{ for all } \emptyset \subset S \subset V, 2 \leq |S| \leq n - 1 \\
& x_{ij} \in \{0, 1\} \text{ for all } ij \in E
\end{aligned}
$$

Outline

1. Dynamic Programming

2. MILP Formulations

3. Solving the DFJ Formulation

Lazy Constraint Approach to DFJ

- relax the set of sub-tour elimination constraints
- $\mathcal{S} = \{\emptyset \subset S \subset V\}$
- $\mathcal{S}' \subset \mathcal{S}$

$$
\begin{aligned}
\text{(RTSPIP)} \quad \min \ & \sum c_{ij} x_{ij} \\
\text{s.t.} \ & \sum_{ij \in \delta(i)} x_{ij} + \sum_{ji \in \delta(i)} x_{ji} = 2 \text{ for all } i \in V \\
& \sum_{ij \in E(S)} x_{ij} \leq |S| - 1 \text{ for all } S \in \mathcal{S}' \\
& x_{ij} \in \{0, 1\} \text{ for all } ij \in E
\end{aligned}
$$

- relax the integrality constraint

$$
\begin{aligned}
\text{(RTSPLP)} \quad \min \ & \sum c_{ij} x_{ij} \\
\text{s.t.} \ & \sum_{ij \in \delta(i)} x_{ij} + \sum_{ji \in \delta(i)} x_{ji} = 2 \text{ for all } i \in V \\
& \sum_{ij \in E(S)} x_{ij} \leq |S| - 1 \text{ for all } S \in \mathcal{S}' \\
& x_{ij} \in \mathbb{R}^+ \text{ for all } ij \in E
\end{aligned}
$$

Implementation V1

set $\mathcal{S}' = \emptyset$

1. $x^* \longleftarrow$ Solve $\texttt{RTSPIP}(\mathcal{S}')$
2. $\mu_k, S \longleftarrow$ Solve $\texttt{SEP}(x^*)$
   if $\mu_k < 2$ then set $\mathcal{S}' = \mathcal{S}' \cup S$ and go to 1
   else return optimal solution x^*

$\texttt{SEP}$: connected components or number of cycles

In gurobi and cplex implementation via Lazy constraints (Model.cbLazy) and call back function called when MIPSOL. See script: tsp_gurobi_lazy

Implementation V2

set $\mathcal{S} = \emptyset$

1. $x^* \longleftarrow$ Solve $\texttt{RLP}(\mathcal{S}')$
2. $\mu_k, S \longleftarrow$ Solve $\texttt{SEPLP}(x^*)$
   if $\mu_k < 2$ then set $\mathcal{S}' = \mathcal{S}' \cup S$ and go to 1
   else go to 3
3. branch and bound and repeat 1. and 2. at every node.

$\texttt{SEPLP}$: LP formulation or Max Flow

In gurobi and cplex implementation via Lazy constraints (Model.cbLazy) and call back functions when LP solution at node.

- Is the Asymmetric formulation TUM when all sub-tour elimination constraints are removed?
- Is the Symmetric formulation TUM when all sub-tour elimination constraints are removed?
- Does the DFJ formulation describe the convex hull of the problem?

Traveling Salesman Problem

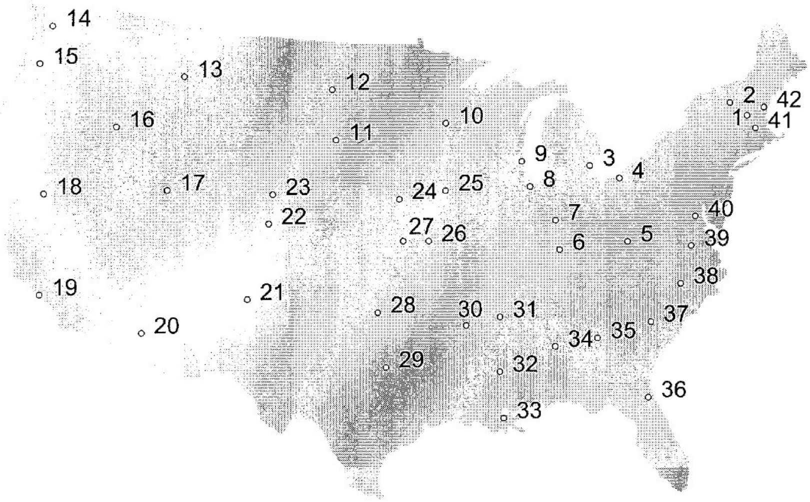


Figure 3.1 Locations of the 42 cities.

Traveling Salesman Problem

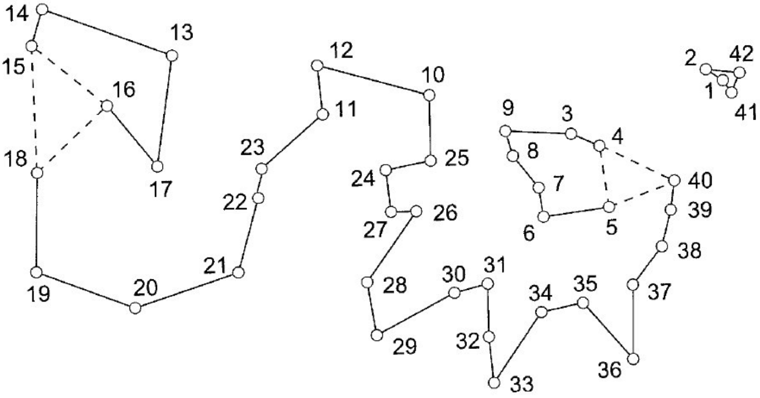


Figure 3.2 Solution of the initial LP relaxation.

Traveling Salesman Problem

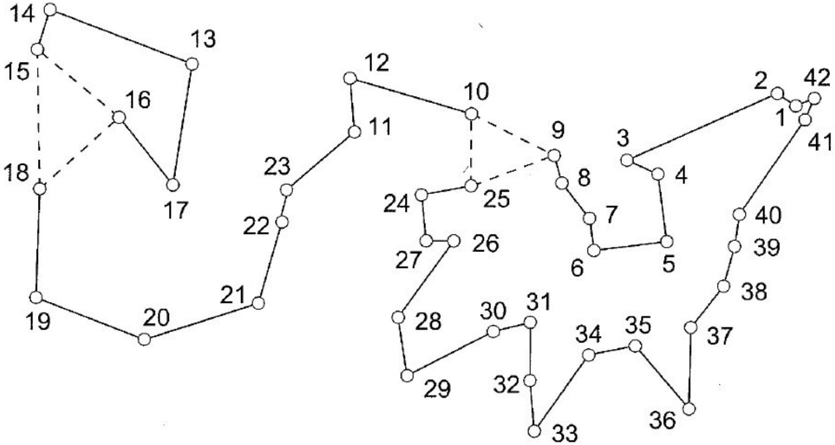


Figure 3.3 LP solution after three subtour constraints.

Traveling Salesman Problem

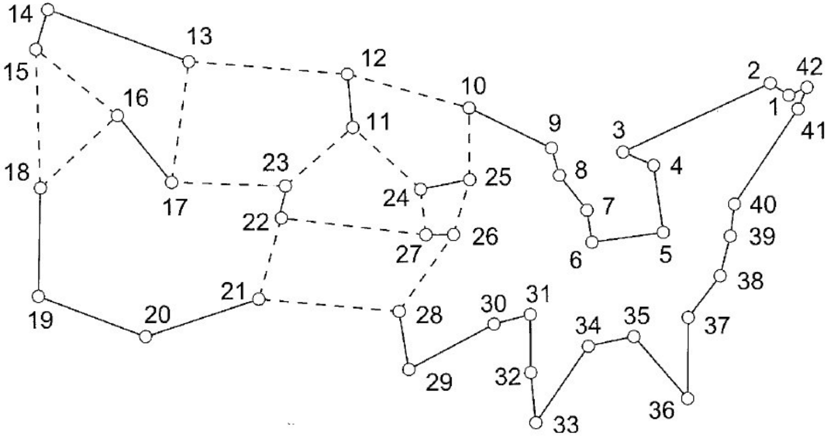


Figure 3.4 LP solution satisfying all subtour constraints.

Traveling Salesman Problem

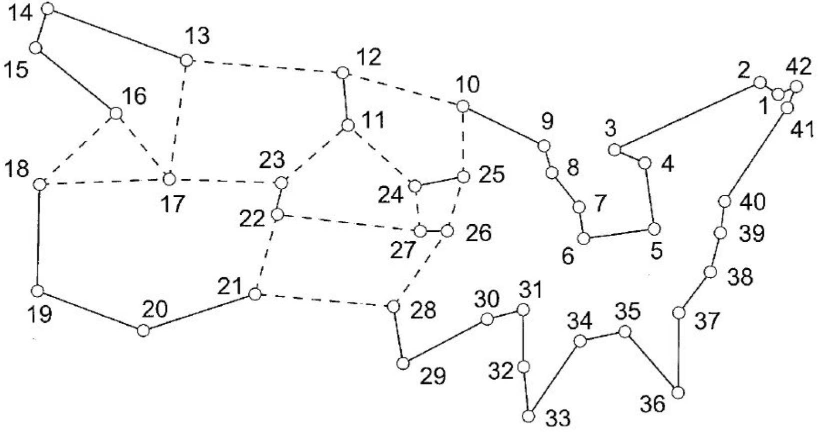


Figure 3.7 What is wrong with this vector?

Traveling Salesman Problem

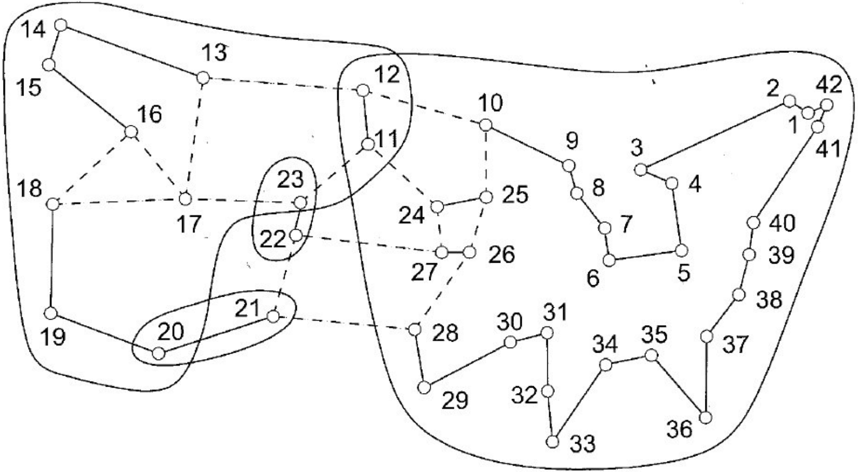


Figure 3.8 A violated comb.

Traveling Salesman Problem

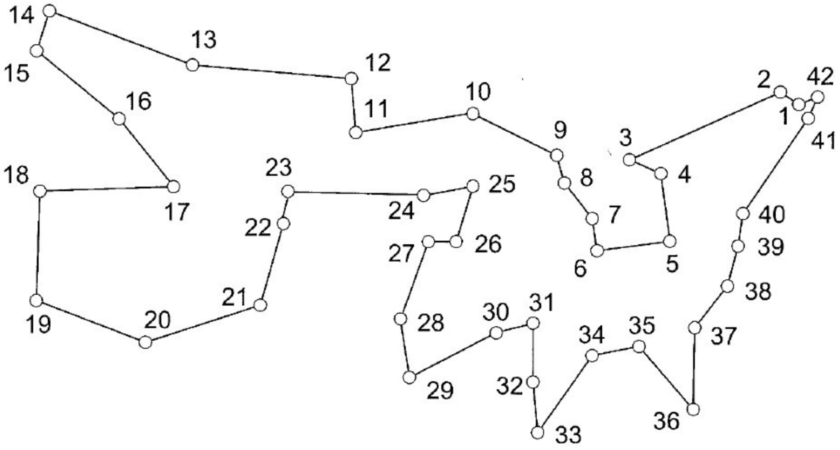


Figure 3.9 An optimal tour through 42 cities.

An Improved DFJ Formulation. (why?)

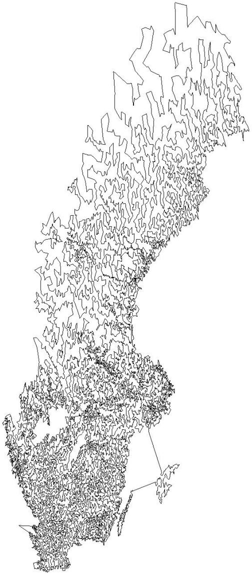
minimize $c^T x$ subject to

$0 \leq x_e \leq 1$ for all edges e,

$\sum(x_e \ : \ v$ is an end of $e) = 2$ for all cities v,

$\sum(x_e \ : \ e$ has one end in S and one end not in $S) \geq 2$
for all nonempty proper subsets S of cities,

$\sum_{i=0}^{i=3}(\sum(x_e \ : \ e$ has one end in S_i and one end not in $S_i) \geq 10$,
for any comb

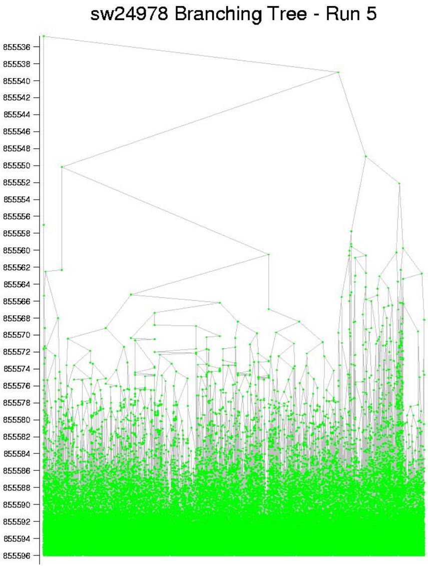


24,978 Cities

solved by LK-heuristic and prooved optimal by branch and cut

10 months of computation on a cluster of 96 dual processor Intel Xeon 2.8 GHz workstations

http://www.tsp.gatech.edu



24,978 Cities

solved by LK-heuristic and prooved optimal by branch and cut

10 months of computation on a cluster of 96 dual processor Intel Xeon 2.8 GHz workstations

http://www.tsp.gatech.edu