


# The Traveling Salesperson Problem


## Algorithms and Networks



Universiteit Utrecht

# Contents

- TSP and its applications
- Heuristics and approximation algorithms
  - Construction heuristics, a.o.: Christofides, insertion heuristics
  - Improvement heuristics, a.o.: 2-opt, 3-opt, Lin-Kernighan



Universiteit Utrecht


A&N: TSP

2

# 1

## Problem definition

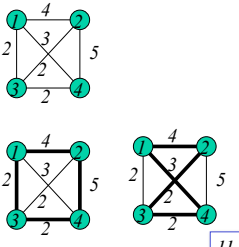

## Applications



Universiteit Utrecht

# Problem

- *Instance:*  $n$  vertices (cities), distance between every pair of vertices
- *Question:* Find shortest (simple) cycle that visits every city

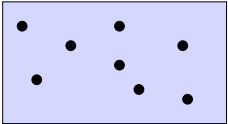

Universiteit Utrecht

A&N: TSP

4

# Applications

- Collection and delivery problems
- Robotics
- Board drilling


Universiteit Utrecht

A&N: TSP

5

# NP-complete

- *Instance:* cities, distances,  $K$
- *Question:* is there a TSP-tour of length at most  $K$ ?
  - Is an NP-complete problem
  - Relation with Hamiltonian Circuit problem



Universiteit Utrecht

A&N: TSP

6

## Assumptions

- Lengths are non-negative (or positive)
- Symmetric:  $w(u,v) = w(v,u)$ 
  - Not always: painting machine application
- Triangle inequality: for all  $x, y, z$ :  
 $w(x,y) + w(y,z) \geq w(x,z)$
- Always valid?



## If triangle inequality does not hold

**Theorem:** If  $P \neq NP$ , then there is no polynomial time algorithm for TSP without triangle inequality that approximates within a ratio  $c$ , for any constant  $c$ .

**Proof:** Suppose there is such an algorithm  $A$ . We build a polynomial time algorithm for Hamiltonian Circuit (giving a contradiction):

- Take instance  $G=(V,E)$  of HC
- Build instance of TSP:
  - A city for each  $v \in V$
  - If  $(v,w) \in E$ , then  $d(v,w) = 1$ , otherwise  $d(v,w) = nc+1$
- $A$  finds a tour with distance at most  $nc$ , if and only if  $G$  has a Hamiltonian circuit



## Heuristics and approximations

- Two types
  - Construction heuristics
    - A tour is built from nothing
  - Improvement heuristics
    - Start with 'some' tour, and continue to change it into a better one as long as possible



## 2

### Construction heuristics



## 1<sup>st</sup> Construction heuristic: Nearest neighbor

- Start at some vertex  $s$ ;  $v=s$ ;
- While not all vertices visited
  - Select closest unvisited neighbor  $w$  of  $v$
  - Go from  $v$  to  $w$ ;
  - $v=w$
- Go from  $v$  to  $s$ .

Can have performance ratio  $O(\log n)$



## Heuristic with ratio 2

- Find a minimum spanning tree
- Report vertices of tree in preorder



## Christofides

- Make a Minimum Spanning Tree  $T$
- Set  $W = \{v \mid v \text{ has odd degree in tree } T\}$
- Compute a minimum weight matching  $M$  in the graph  $G[W]$ .
- Look at the graph  $T+M$ . (Note: Eulerian!)
- Compute an Euler tour  $C'$  in  $T+M$ .
- Add shortcuts to  $C'$  to get a TSP-tour



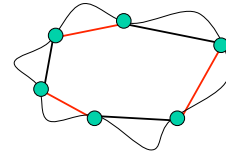
Universiteit Utrecht

A&N: TSP

13

## Ratio 1.5

- Total length edges in  $T$ : at most  $OPT$
- Total length edges in matching  $M$ : at most  $OPT/2$ .
- $T+M$  has length at most  $3/2 OPT$ .
- Use  $\Delta$ -inequality.



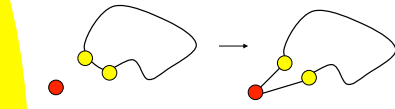
Universiteit Utrecht

A&N: TSP

14

## Closest insertion heuristic

- Build tour by starting with one vertex, and inserting vertices one by one.
- Always insert vertex that is closest to a vertex already in tour.



Universiteit Utrecht

A&N: TSP

15

## Closest insertion heuristic has performance ratio 2

- Build tree  $T$ : if  $v$  is added to tour, add to  $T$  edge from  $v$  to closest vertex on tour.
- $T$  is a Minimum Spanning Tree (Prim's algorithm)
- Total length of  $T \leq OPT$
- Length of tour  $\leq 2 * \text{length of } T$



Universiteit Utrecht

A&N: TSP

16

## Many variants

- **Closest insertion**: insert vertex closest to vertex in the tour
- **Farthest insertion**: insert vertex whose minimum distance to a node on the cycle is maximum
- **Cheapest insertion**: insert the node that can be inserted with minimum increase in cost
  - Gives also ratio 2
  - Computationally expensive
- **Random insertion**: randomly select a vertex
- *Each time: insert vertex at position that gives minimum increase of tour length*



Universiteit Utrecht

A&N: TSP

17

## Cycle merging heuristic

- Start with  $n$  cycles of length 1
- Repeat:
  - Find two cycles with minimum distance
  - Merge them into one cycle
- Until 1 cycle with  $n$  vertices
- This has ratio 2: compare with algorithm of Kruskal for MST.



Universiteit Utrecht

A&N: TSP

18

## Savings

- Cycle merging heuristic where we merge tours that provide the largest “savings”: can be merged with the smallest additional cost / largest savings



## Some test results

- In an overview paper, Junger et al report on tests on set of instances (105 – 2392 vertices; city-generated TSP benchmarks)
- Nearest neighbor: 24% away from optimal in average
- Closest insertion: 20%;
- Farthest insertion: 10%;
- Cheapest insertion: 17%;
- Random Insertion: 11%
- Preorder of min spanning tree: 38%
- Christofides: 19% with improvement 11% / 10%
- Savings method: 10% (and fast)



## 3

### Improvement heuristics



## Improvement heuristics

- Start with a tour (e.g., from heuristic) and improve it stepwise
  - 2-Opt
  - 3-Opt
  - K-Opt
  - Lin-Kernighan
  - Iterated LK
  - Simulated annealing, ...

*Iterative improvement*

*Local search*



## Scheme

- Rule that modifies solution to different solution
- While there is a Rule(sol, sol') with sol' a better solution than sol
  - Take sol' instead of sol
- Cost decrease
- Stuck in 'local minimum'
- Can use exponential time in theory...



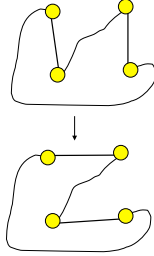
## Very simple

- Node insertion
  - Take a vertex  $v$  and put it in a different spot in the tour
- Edge insertion
  - Take two successive vertices  $v, w$  and put these as edge somewhere else in the tour



## 2-opt

- Take two edges  $(v,w)$  and  $(x,y)$  and replace them by  $(v,x)$  and  $(w,y)$  OR  $(v,y)$  and  $(w,x)$  to get a tour again.
- Costly: part of tour should be turned around



Universiteit Utrecht

A&N: TSP

25

## 2-Opt improvements

- Reversing shorter part of the tour
- Clever search to improving moves
- Look only to subset of candidate improvements
- Postpone correcting tour
- Combine with node insertion
- On  $\mathbb{R}^2$ : get rid of crossings of tour



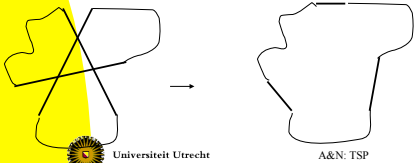
Universiteit Utrecht

A&N: TSP

26

## 3-opt

- Choose three edges from tour
- Remove them, and combine the three parts to a tour in the cheapest way to link them



Universiteit Utrecht

A&N: TSP

27

## 3-opt

- Costly to find 3-opt improvements:  $O(n^3)$  candidates
- $k$ -opt: generalizes 3-opt



Universiteit Utrecht

A&N: TSP

28

## Lin-Kernighan

- Idea: modifications that are *bad* can lead to enable something *good*
- Tour modification:
  - Collection of simple changes
  - Some increase length
  - Total set of changes decreases length



Universiteit Utrecht

A&N: TSP

29

## LK

- One LK step:
  - Make sets of edges  $X = \{x_1, \dots, x_r\}$ ,  $Y = \{y_1, \dots, y_r\}$ 
    - If we replace  $X$  by  $Y$  in tour then we have another tour
  - Sets are built stepwise
- Repeated until ...
- Variants on scheme possible



Universiteit Utrecht

A&N: TSP

30

## One LK step

- Choose vertex  $t_1$ , and edge  $x_1 = (t_1, t_2)$  from tour.
- $i=1$
- Choose edge  $y_i = (t_2, t_3)$  not in tour with  $g_i = w(x_1) - w(y_i) > 0$  (or, as large as possible)
- Repeat a number of times, or until ...
  - $i++$ ;
  - Choose edge  $x_i = (t_{2i-1}, t_{2i})$  from tour, such that
    - $x_i$  not one of the edges  $y_j$
    - $oldtour - X + (t_{2i}, t_1) + Y$  is also a tour
  - if  $oldtour - X + (t_{2i}, t_1) + Y$  has shorter length than  $oldtour$ , then take this tour: done
  - Choose edge  $y_i = (t_{2i}, t_{2i+1})$  such that
    - $g_i = w(x_i) - w(y_i) > 0$
    - $y_i$  is not one of the edges  $x_j$ .
    - $y_i$  not in the tour



## Iterated LK

*Cost much time  
Gives excellent results*

- Construct a start tour
- Repeat the following  $r$  times:
  - Improve the tour with Lin-Kernighan until not possible
  - Do a random 4-opt move that *does not increase the length with more than 10 percent*
- Report the best tour seen



## Other methods

- Simulated annealing and similar methods
- Problem specific approaches, special cases
- Iterated LK combined with treewidth/branchwidth approach:
  - Run ILK a few times (e.g., 5)
  - Take graph formed by union of the 5 tours
  - Find minimum length Hamiltonian circuit in graph with clever dynamic programming algorithm



4

A dynamic programming algorithm



## Held-Karp algorithm for TSP

- $O(n^2 2^n)$  algorithm for TSP
- Uses Dynamic programming
- Take some starting vertex  $s$
- For set of vertices  $R$  ( $s \in R$ ), vertex  $w \in R$ , let
  - $B(R, w)$  = minimum length of a path, that
    - Starts in  $s$
    - Visits all vertices in  $R$  (and no other vertices)
    - Ends in  $w$



## TSP: Recursive formulation

- $B(\{s\}, s) = 0$
- If  $|S| > 1$ , then
  - $B(S, w) = \min_{v \in S - \{w\}} B(S - \{w\}, v) + w(v, w)$
- If we have all  $B(V, v)$  then we can solve TSP.
- Gives requested algorithm using DP-techniques.



## Conclusions

- TSP has many applications
- Also many applications for variants of TSP
- Heuristics: construction and improvement
- Further reading:
  - M. Jünger, G. Reinelt, G. Rinaldi, *The Traveling Salesman Problem*, in: *Handbooks in Operations Research and Management Science, volume 7: Network Models*, North-Holland Elsevier, 1995.

