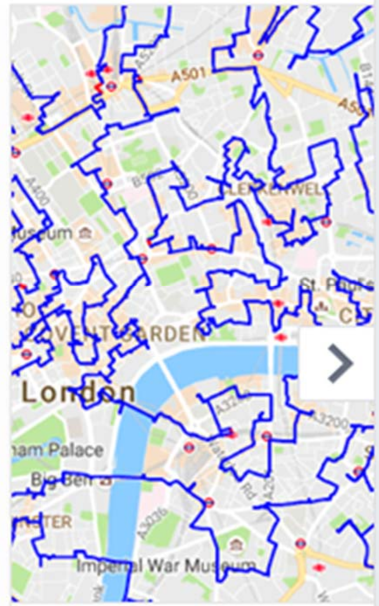
From Facebook

In Pursuit of the Traveling Salesman by William J. Cook
October 19 at 4:23pm ·

Looking for a crawl? Here is the shortest-possible walking tour through 24,727 pubs in the United Kingdom. This is by far the largest road-distance traveling salesman problem that has ever been solved.
http://www.math.uwaterloo.ca/tsp/pubs/index.html

UK Pubs Traveling Salesman Problem



UK Pubs Traveling Salesman I

Like · Comment · Share

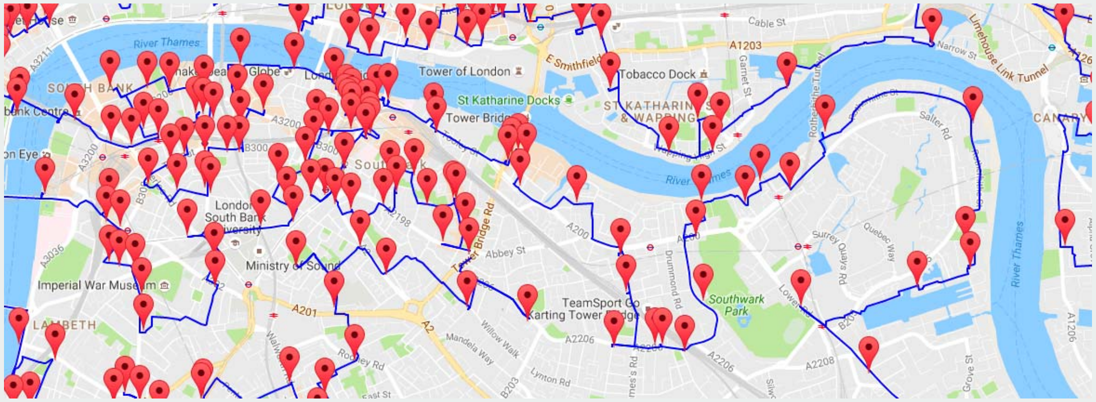
You, Paul Bouman and 80 others · Chronological

Optimal tour along 24727 pubs in the UK

- Road distance (by google maps)
- see also http://www.math.uwaterloo.ca/tsp/pubs/index.html

(part of TSP homepage http://www.math.uwaterloo.ca/tsp/)

- Applies branch-and-cut and heuristic based on Lin-Kernighan



Universiteit Utrecht

[Faculty of Science
Information and Computing Sciences]

The Landscape of Algorithms example: Travelling Salesman Problem

Algorithms for Decision Support

Solution quality / Computation time	Optimum	Bound on quality	Good solution, no quality guarantee
Polynomial	Polynomial solution algorithms	Approximation algorithms	Construction heuristics
Super polynomial and / or no guarantee	Exact algorithms: • Tree search • Dynamic programming • Integer linear programming • ……	Hybrid algorithms • Column generation without complete branch-and-price	Meta heuristics: • Local search • Genetic algorithms

cience
ences]

Contents

- TSP and its applications
- Construction heuristics and approximation algorithms
- Local search
- Exact algorithms

- *TSP is the most frequently used benchmark for new algorithms*

Universiteit Utrecht

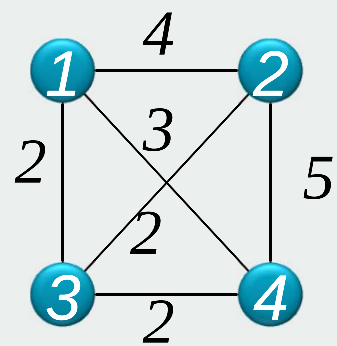
[Faculty of **Science**
Information and Computing Sciences]

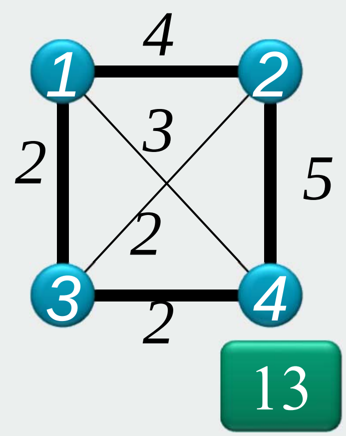
Travelling Salesman Problem

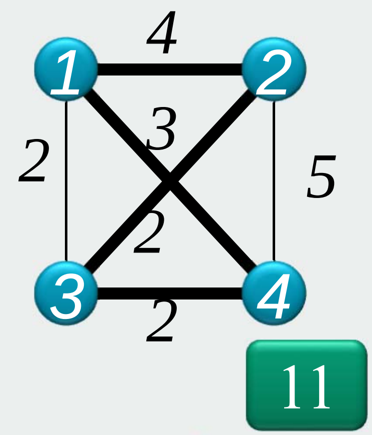
PROBLEM DEFINITION APPLICATIONS

Travelling Salesman Problem

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

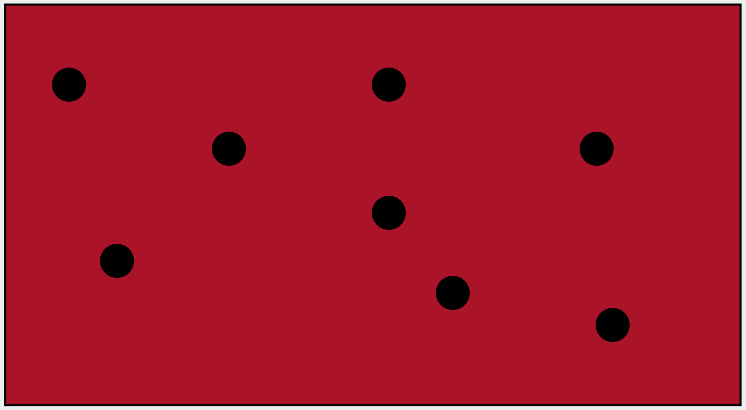






Applications

- Vehicle routing
- Routing school buses
- Pickup and delivery problems
- Robotics
- Scheduling of a machine to drill holes in a circuit board or other object (chip manufacturing)



Universiteit Utrecht

Assumptions / Problem Variants

- Complete graph vs. underlying graph structure.
  - Complete graph: a given distance between all pairs of cities.
- Directed edges vs. undirected arcs.
  - Undirected: symmetric: $w(u,v) = w(v,u)$.
  - Directed: not symmetric.
  - Asymmetric examples: one-way streets, prices of flight tickets.
- Lengths:
  - Lengths are non-negative (or positive).
  - Triangle inequality: for all x, y, z:

    $$\mathbf{w}(x,y) + \mathbf{w}(y,z) \geq \mathbf{w}(x,z)$$
- Different assumptions lead to different problems.

NP-complete

- **Instance**: cities, distances, *k*.
- **Question**: is there a TSP-tour of length at most *k*?
  - Is an NP-complete problem.
  - Has been shown by reduction from Hamiltonian Circuit problem.

Solution quality / Computation time	Optimum	Bound on quality	Good solution, no quality guarantee
Polynomial	Polynomial solution algorithms	Approximation algorithms	Construction heuristics
Super polynomial and / or no guarantee	Exact algorithms: • Tree search • Dynamic programming • Integer linear programming • ……	Hybrid algorithms • Column generation without complete branch-and-price	Meta heuristics: • Local search • Genetic algorithms

cience
ences]

Approximation algorithms

- Have performance guarantee, also called approximation ratio.

- We consider a *minimization* problem P.
  - Let $Z_A\,(x)$ be the value found by algorithm *A* for instance *x*
  - Let $Zopt\ (x)$be the optimal value for instance *x*
  - Clearly $Z_A(x) \geq Z_{opt}(x)$

- Then *A* is an *approximation algorithm with worst case performance guarantee c* (*c*>1) if for each instance *x* of P we have

$$\boldsymbol{Z_A(x) \leq cZ_{opt}(x) \quad i.e. \quad \frac{Z_A(x)}{Z_{opt}(x)} \leq c}$$

Approximation algorithms: now maximization

- We consider a *maximization* problem P.
  - Let $Z_A\ (x)$ be the value found by algorithm *A* for instance *x*
  - Let $Zopt\ (x)$be the optimal value for instance *x*
  - Clearly $Z_A(x) \leq Z_{opt}(x)$

- Then *A* is an *approximation algorithm with worst case performance guarantee c* (*c*<1) if for each instance *x* of P we have

$$\boldsymbol{Z_A(x) \geq cZ_{opt}(x) \quad i.e. \quad \frac{Z_A(x)}{Z_{opt}(x)} \geq c}$$

1st 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*.

Minimum spanning tree

- Algorithm of Kruskal:
  - Builds a tree *T* step by step
  - In each step it adds the edge with minimal cost in the graph which does not cause *T* to be a cycle

- Algorithm of Prim/Dijkstra
  - Builds a tree *T* step by step
  - In each step it adds the edge minimal cost connecting a vertex from *T* with a vertex outside *T*

2nd Construction Heuristic: Double tree heuristic with ratio 2

- Assume *symmetric* TSP
- Find a minimum spanning tree
- Make a tour as follows:
  - Walk along all vertices of the MST (you visits every edge twice)
  - Apply shortcuts

- If *triangle inequality*, we have approximation ratio 2:
  - OPT ≥ MST
  - 2 MST ≥ Result
  - Result ≤ 2MST ≤ 2OPT
  - Result/OPT ≤ 2

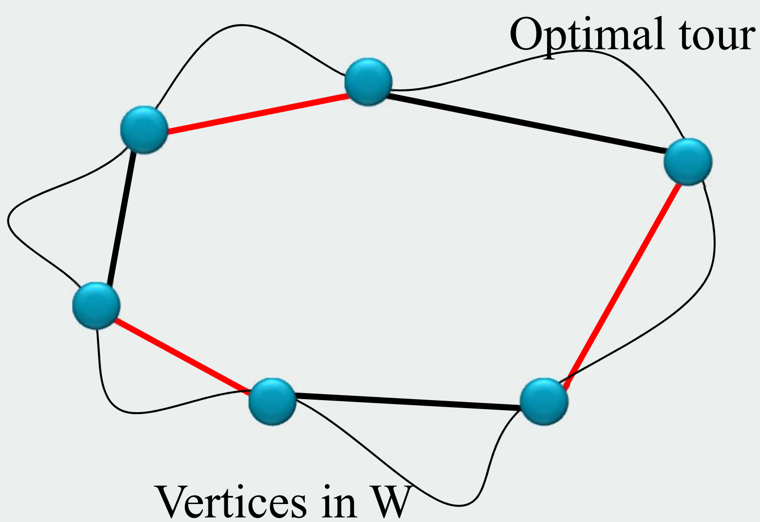
3rd Construction Heuristic: Christofides

Assume symmetric TSP

1. Make a Minimum Spanning Tree T.
2. Set W = $\{v \mid v$ has odd degree in tree T$\}$.
   |W| must be even, since for any graph $\sum_{v \in V} degree(v) = 2|E|$
3. Compute a minimum weight matching M in the graph G[W].
4. Look at the graph T+M.
   - Note that T+M is Eulerian (all vertices haven even degree)!
5. Compute a Euler tour (tour that visits every edge exactly once) C' in T+M.
6. Add shortcuts to C' to get a TSP-tour.

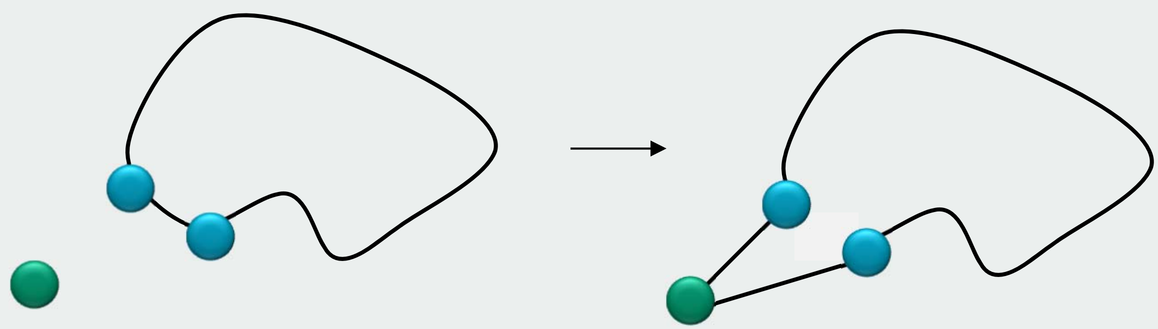
Triangle inequality: Ratio 1.5

- Total length edges in MST T: at most OPT
- Red matching+ black matching ≤ OPT
- Total length edges in min weight matching M: at most OPT/2.
- Euler circuit C′ in (T+M) has length at most 3/2 OPT.
- By Δ-inequality, result after shortcut at most 3/2 OPT .



4^{th} Construction Heuristic: 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.



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.
  - Computationally expensive.
- **Random insertion:** randomly select a vertex.

- Each time: insert vertex at position that gives minimum increase of tour length.

5th Construction Heuristic: Cycle merging heuristic

- Start with *n* cycles of length 0.
- Repeat:
  - Find two cycles with minimum distance.
  - Merge them into one cycle.
- Until 1 cycle with *n* vertices.

6^{th} Construction Heuristic: Savings heuristic

- Cycle merging heuristic where we merge tours that provide the largest “savings”:
  - Saving for a merge: merge with the smallest additional cost / largest savings.

- Quite similar to Clark and Wright savings heuristic for vehicle routing

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:
  - Closest insertion:
  - Farthest insertion:
  - Cheapest insertion:
  - Random Insertion:
  - Min spanning trees:
  - Christofides
  - Savings method:
- What is the average distance to the optimum

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%;
  - Min spanning trees: 38%;
  - Christofides: 19% with improvement 11% / 10%;
  - Savings method: 10% (and fast).

If triangle inequality does not hold: negative result

Theorem: If P≠NP, then there is no polynomial time algorithm for TSP without triangle inequality that approximates within a ratio $c>0$, for any constant c.

Proof:

- Suppose there is a polynomial time approximation algorithm *A* with ratio c.
- We build a polynomial time algorithm for Hamiltonian Circuit (giving a contradiction with P≠NP):
  - Take instance G=(V,E) of Hamiltonian Circuit.
  - Build instance of TSP:
    - A city for each $v \in$ V.
    - If $(v,w) \in$ E, then $d(v,w) = 1$, otherwise $d(v,w) = nc+1$.
  - Now run *A* on this instance
  - *A* finds a TSP-tour with distance at most nc, if and only if, G has a Hamiltonian circuit.

Solution quality Computation time	Optimum	Bound on quality	Good solution, no quality guarantee
Polynomial	Polynomial solution algorithms	Approximation algorithms	Construction heuristics
Super polynomial and / or no guarantee	Exact algorithms: • Tree search • Dynamic programming • Integer linear programming • ……	Hybrid algorithms • Column generation without complete branch-and-price	Meta heuristics: • Local search • Genetic algorithms

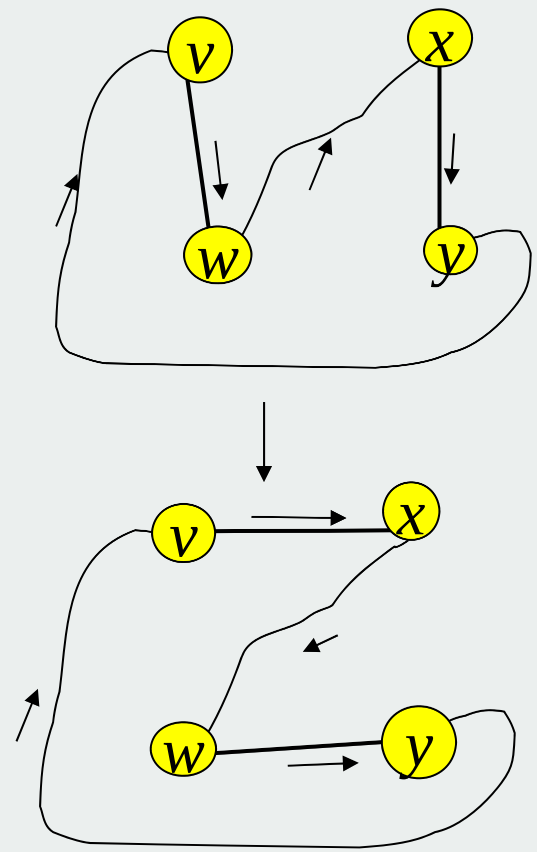
cience
ences]

Improvement heuristics

- Start with a tour (e.g., from construction heuristic) and improve it stepwise
- Improvement heuristics can be used in different local search methods.
  - Iterated local search
  - Variable neighborhood search
  - Simulated annealing
  - Tabu search
  - Genetic algorithms

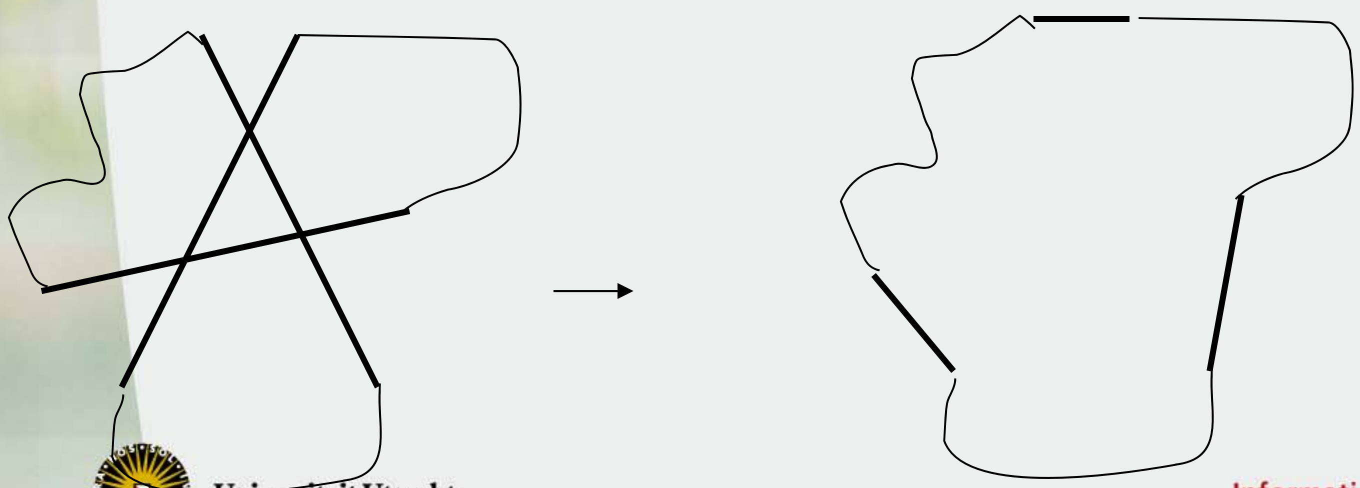
2-opt

- Take two edges (v,w) and (x,y) and replace them by (v,x) and (w,y) if this improves the tour.
- Costly: part of tour should be turned around.
- In $\mathbb{R}^2$: get rid of crossings of tour.



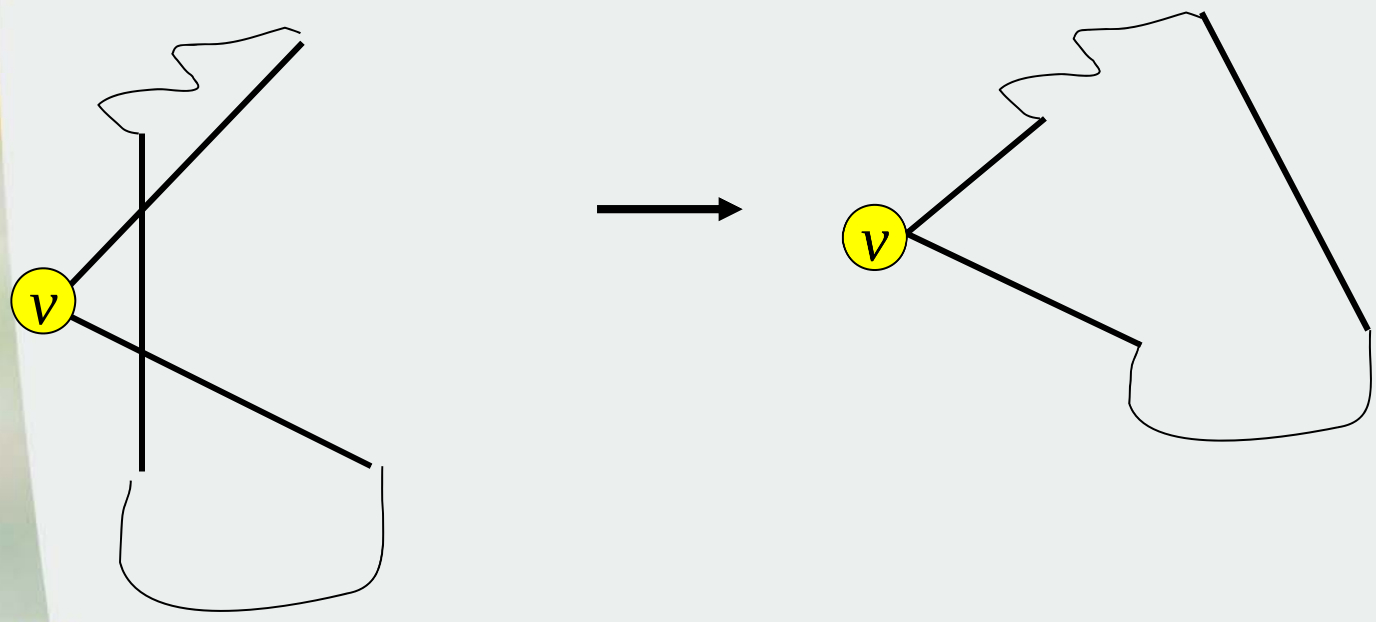
3-opt

- Choose three edges from tour
- Remove them, and combine the three parts to a tour in the cheapest way to link them
- *k*-opt: generalizes 3-opt



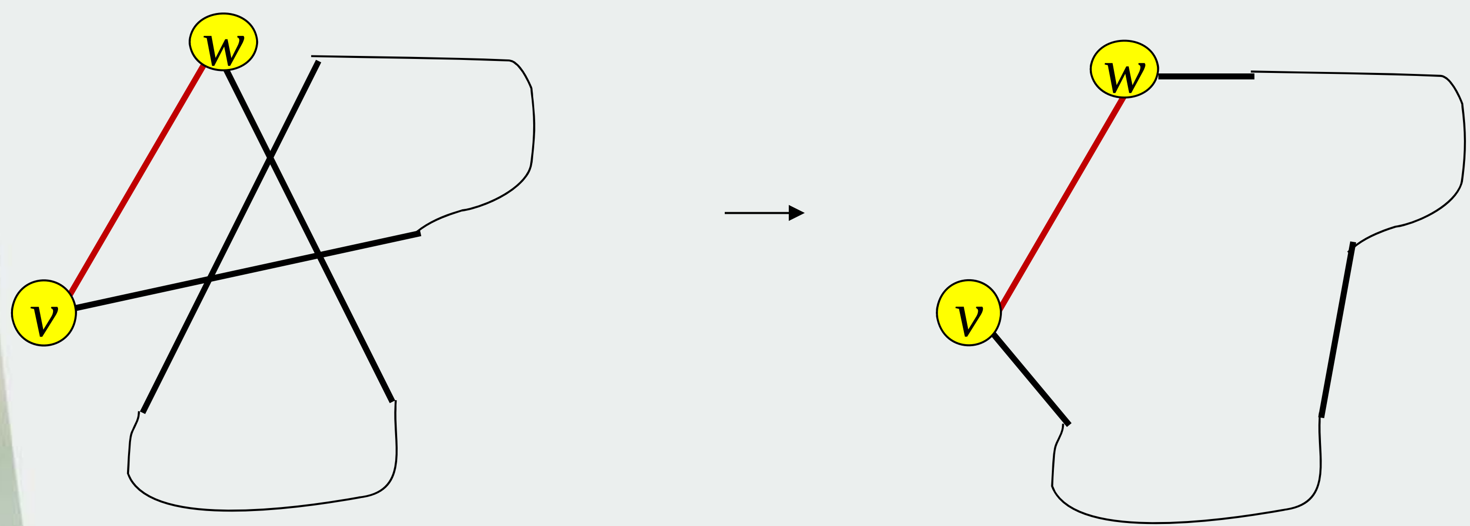
Node insertion: 2.5-opt

- Node insertion:
  - Take a vertex *v* and put it in a different spot in the tour.
- This is a special case of 3-opt, called 2.5-opt:
  - two of the three removed edges are consecutive edges in the tour, i.e. they are connected to the same vertex



Edge insertion

- Edge insertion:
  - Take two successive vertices *v*, *w* and put these as edge somewhere else in the tour.
- This is also a special case of 3-opt:
  - two of the three removed edges are almost consecutive edges in the tour, i.e. they are connected to the same edge

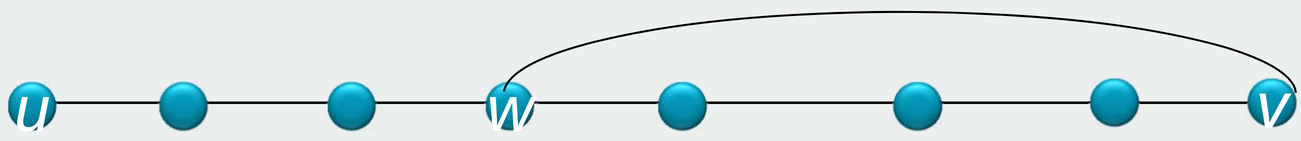


Lin-Kernighan

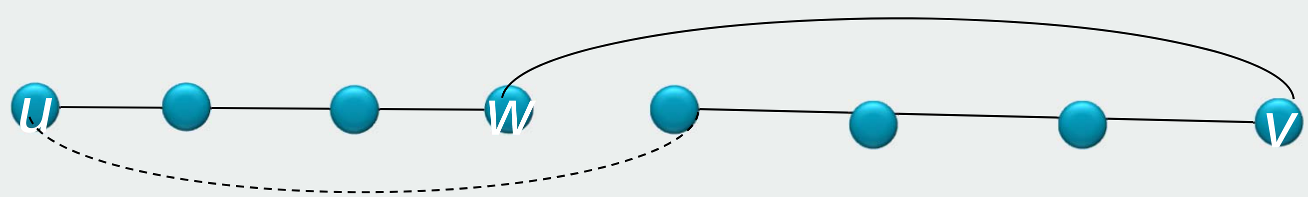
- Idea: modifications that are *bad* can lead to something *good*
- Tour modification:
  - Collection of simple changes
  - Some increase length
  - Total set of changes decreases length

Lin-Kernighan

1. Break the tour
2. Add a new edge



3. Break the subtour
4. If connecting to a complete tour gives an improvement, stop. Otherwise repeat (go to Step 1).



Universiteit Utrecht

[Faculty of Science
Information and Computing Sciences]

Iterated Lin-Kernighan

- 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.

Cost much time. Gives excellent results!

Solution quality / Computation time	Optimum	Bound on quality	Good solution, no quality guarantee
Polynomial	Polynomial solution algorithms	Approximation algorithms	Construction heuristics
Super polynomial and/or no guarantee	Exact algorithms: • Tree search • Dynamic programming • Integer linear programming • ……	Hybrid algorithms • Column generation without complete branch-and-price	Meta heuristics: • Local search • Genetic algorithms

cience
ences]

Exact algorithms

- Dynamic Programming: Held Karp
- Branch-and cut:
  - World-record exact TSP solving
  - CONCORDE: http://www.math.uwaterloo.ca/tsp/concorde/index.html

The slides for this course were written by:

Marjan van den Akker

Peter de Waal

Han Hoogeveen

Hans Bodlaender

Johan van Rooij

Alison Liu

Universiteit Utrecht

[Faculty of **Science**
Information and Computing Sciences]