5.3. HEURISTICS FOR THE TSP

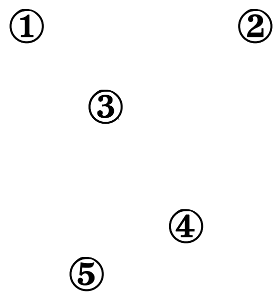
Notation: f_a = length of the tour given by the algortihm
f_{min} = length of the optimal tour

Most of the following heuristics are designed to solve symmetric instances of TSP, although they can be modified for asymmetric cases. Let us assume, that the distances
• are nonnegative and symmetric
• satisfy the triangle inequality.
Without these assumptions, the bound given for the objective function value may not be valid.

Many of these methods, e.g. the construction heuristics, start with an arbitrary vertex. To increase reliability, the procedure can be repeated from several (or all) vertices in turn. Then the computation time multiplies correspondingly.

Example 5.5. The heuristics are demonstrated with this example of n=5 vertices and distances in the following matrix:

c_{ij}	***1***	***2***	***3***	***4***	***5***
1	-	8	4	9	9
2	8	-	6	7	10
3	4	6	-	5	6
4	9	7	5	-	4
5	9	10	6	4	-



5.3.1. CONSTRUCTION METHODS

In construction methods, the tour is built up by adding new vertices to a partial tour or path.

A. NEAREST NEIGHBOR METHOD
1. Choose an arbitrary starting node for the tour. This is the initial path of one node.
2. Find an unconnected node that is closest to the last node of the path. Join this node to the last node.
3. Repeat step 2 until all nodes are joined to the path. Then, join the last node to the first one.

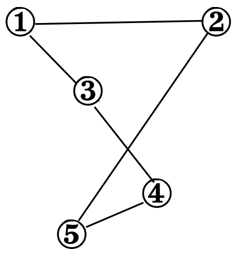
Evaluation of the solution:	$f_a / f_{min} \leq ½\lceil\log(n)\rceil + ½$
Worst case complexity:	$T(n) = O(n^2)$

($\lceil x \rceil$ = smallest integer that is $\geq x$)

Experimental testing on the TSPLIB problems gives an average $f_a / f_{min} \approx 1.26$.

Example 5.5.
Starting with node 1, the solution is 1-3-4-5-2-1, f = 31

Different solutions result from different starting points.



B. METHOD OF CLARKE AND WRIGHT = THE SAVINGS ALGORITHM

1. Choose an arbitrary node k (as a center node) for the tour. The starting closed walk visits each other node one by one and returns to node k after each of them.
2. For all pairs of nodes i,j = 1,...,n, such that $i \neq k$, $j \neq k$, $i \neq j$, calculate the *savings* for replacing cycles k-i-k and k-j-k by a combined cycle k-i-j-k:

$$s_{ij} = c_{ki} + c_{kj} - c_{ij}.$$

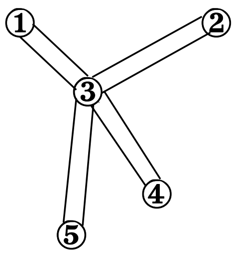
3. Sort the savings in decreasing order.
4. In the order of the savings s_{ij}, modify the current walk by replacing two edges incident to the center, i-k and k-j, by edge i-j. The replacement is not made if i and j already belong to the same subtour. Repeat until a Hamiltonian cycle is formed.

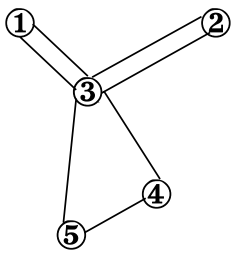
Evaluation of the solution:	$f_a / f_{min} = O(\log n)$
Worst case complexity:	$T(n) = O(n^2 \log n)$

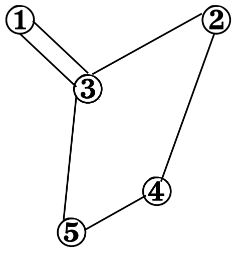
Example 5.5.
Savings matrix, with center k=3:

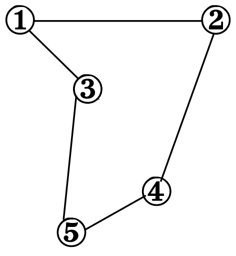
s_{ij}	***1***	***2***	***4***	***5***
1	-	2	0	1
2	2	-	4	2
4	0	4	-	7
5	1	2	7	-

Symmetry: $s_{ij} = s_{ji}$ (only upper half of the matrix needed)
Decreasing order: s_{45}, s_{24}, s_{12}, s_{25}, s_{15}, s_{14}

1)


2)


3)


4)


Solution: 1-3-5-4-2-1, f = 29

C. INSERTION METHODS

In insertion methods, the tour is consrtructed by inserting new nodes to subtours, i.e. partial tours.

Notation:
T = subtour, partial tour
$\triangle f = c_{ik} + c_{kj} - c_{ij}$ = increase in tour length, when node k is inserted between nodes i and j.
Define the distance from node k to subtour T as $d(k,T) = \min\limits_{j \in T} c_{kj}$

METHOD OF NEAREST INSERTION
1. Choose an arbitrary node i as a starting node.
2. Find a node j closest to i. Form a subtour T = i-j-i.
3. Find a node k not in the subtour that is closest to any of the subtour nodes (that is, for which the distance to the subtour is smallest),

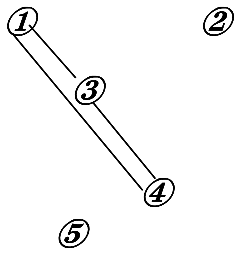
$$d(k,T) = \min_{i \notin T} d(i,T)$$

4. Find an edge [i,j] of the subtour to insert k, such that the increase of length $\triangle f = c_{ik} + c_{kj} - c_{ij}$ is minimized. Modify the subtour by inserting k between i and j.
5. Go to 3 until a Hamiltonian cycle is formed.

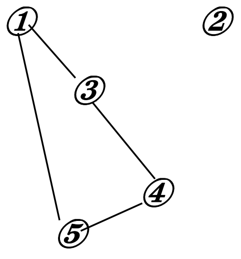
Evaluation of the solution:	$f_a / f_{min} \le 2$
Worst case complexity:	$T(n) = O(n^2)$

Example 5.5.

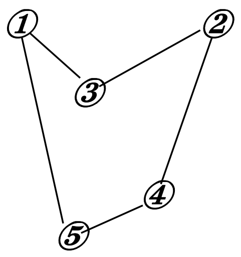
Starting node: i=1.
Closest node: j=3: T= 1-3-1.
Closest node to the subtour:
k=4: T = 1-3-4-1
k=5: T = 1-3-4-5-1
k=2: insertion to 1-3: $\triangle f = c_{12} + c_{23} - c_{13} = 8+6-4 = 10$
insertion to 3-4: $\triangle f = c_{32} + c_{24} - c_{34} = 6+7-5 = 8 \leftarrow$ smallest
T = 1-3-2-4-5-1, f = 30



T = 1-3-4-1



T = 1-3-4-5-1



Solution: T = 1-3-2-4-5-1, f =30

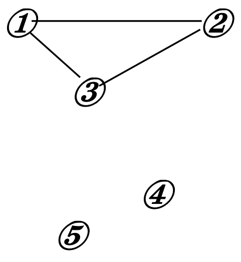
METHOD OF CHEAPEST INSERTION

1. Choose an arbitrary node i as a starting node.
2. Find a node j closest to i. Form a subtour T = i-j-i.
3. Find an edge [i,j] of the subtour and a node k not in the subtour, such that the increase of length $\triangle f = c_{ik} + c_{kj} - c_{ij}$ is minimized. Modify the subtour by inserting k between i and j.
4. Go to 3 until a Hamiltonian cycle is formed.

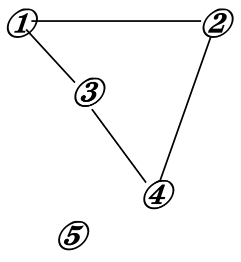
Evaluation of the solution:	$f_a / f_{min} \leq 2$
Worst case complexity:	$T(n) = O(n^2 \log n)$

Example 5.5.

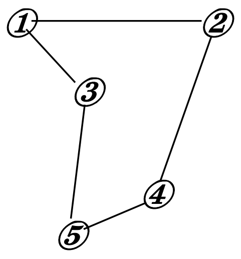
Starting node i=1.
Closest node j=3: T= 1-3-1.
Edge 1-3, insert node
2: $\triangle f = c_{12} + c_{23} - c_{13} = 8+6-4 = 10$ ← smallest
4: $\triangle f = c_{14} + c_{43} - c_{13} = 9+5-3 = 11$
5: $\triangle f = c_{15} + c_{53} - c_{13} = 9+6-3 = 12$
k=2: T = 1-2-3-1
1-2, insert node 4: $\triangle f = c_{14} + c_{42} - c_{12} = 9+7-8 = 8$
1-2, insert node 5: $\triangle f = c_{15} + c_{52} - c_{12} = 9+10-8 = 11$
2-3, insert node 4: $\triangle f = c_{24} + c_{43} - c_{23} = 7+5-6 = 6$ ← smallest
2-3, insert node 5: $\triangle f = c_{25} + c_{53} - c_{23} = 10+6-6 = 10$
3-1, insert node 4: $\triangle f = c_{34} + c_{41} - c_{31} = 5+9-4 = 10$
3-1, insert node 5: $\triangle f = c_{35} + c_{51} - c_{31} = 6+9-4 = 11$
k=4: T = 1-2-4-3-1
Last node 5: optimal insertion to 4-3.
k=5: T = 1-2-4-5-3-1, f = 29



T = 1-2-3-1



T = 1-2-4-3-1



Solution T = 1-2-4-3-1, f = 29

METHOD OF FARTHEST INSERTION

1. Choose an arbitrary node i as a starting node.
2. Find the farthest node from i, i.e. the node j for which distance c_{ij} is longest. Form a subtour T = i-j-i.
3. Find a node k not in the subtour that is farthest from any of the subtour nodes (that is, for which the distance to the subtour is longest),

$$d(k,T) = \max_{i \notin T} d(i,T)$$

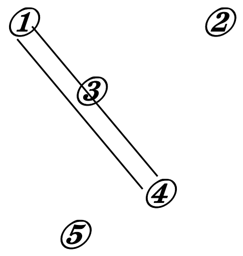
4. Find an edge [i,j] of the subtour to which the insertion of k gives the smallest increase of length, i.e. for which $\triangle f = c_{ik} + c_{kj} - c_{ij}$ is smallest. Modify the subtour by inserting k between i and j.
5. Go to 3 until a Hamiltonian cycle is formed.

Evaluation of the solution:	$f_a / f_{min} \leq \lceil \log(n) \rceil + 1$
Worst case complexity:	$T(n) = O(n^2)$

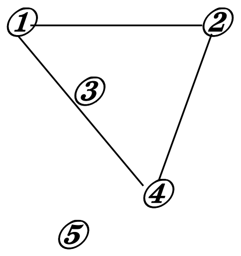
According to experimental results, the Farthest Insertion Method usually gives a better average function value than nearest and cheapest insertion, although it is unknown whether the f-ratio has a constant upper bound.

Example 5.5.

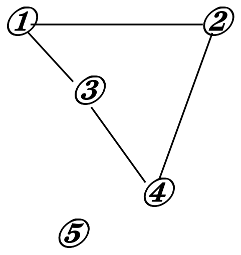
Starting node i=1.
Farthest node j=4: T= 1-4-1.
Smallest distance to T from node
2: $c_{24} = 7$ ← largest
3: $c_{31} = 4$
5: $c_{54} = 4$
k=2: T = 1-2-4-1
Smallest distance to T from node
3: $c_{31} = 4$
5: $c_{54} = 4$
k=3: T = 1-2-4-3-1, because $\triangle$f smallest when 3 inserted to 4-1.
Last node 5: optimal insertion to 4-3.
k=5: T = 1-2-4-5-3-1, f=29

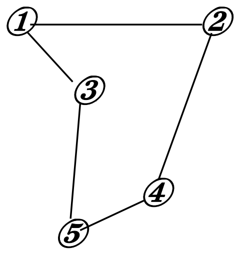


T = 1-4-1



T = 1-2-4-1





T = 1-2-4-3-1

Solution: T = 1-2-4-5-3-1, f = 29

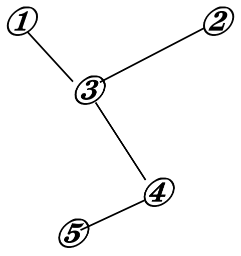
OTHER INSERTION METHODS:
Quick insertion or Nearest addition, Arbitrary insertion, Convex hull insertion, Greatest angle insertion

D. MINIMUM SPANNING TREE METHOD
In this algorithm an Eulerian cycle is formed in a multigraph where the edges of a minimum spanning tree are doubled.
1. Construct the minimum spannng tree for the weighted graph G.
2. Choose a leaf node i_0 (i.e. node of degree 1). Set $i:=i_o$.
3. If an untraversed edge [i,j] incident to i exists, go from i to j (edge [i,j] is added to the path). Set i:=j and repeat step 3.
If all edges from i are traversed, return to the first predecessor of i, be it node k. If $k=i_o$ (the starting node), an Eulerian cycle is formed: go to 4. Else, set i:=k and repeat step 3..
4. Form a Hamiltonian cycle using *shortcuts*: traverse the Eulerian cycle by skipping every node already visited.

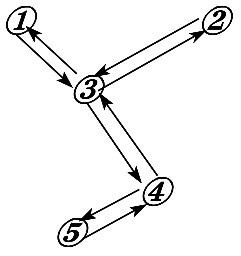
Evaluation of the solution:	$f_a / f_{min} \leq 2$
Worst case complexity:	$T(n) = O(n^2)$

Example 5.5.

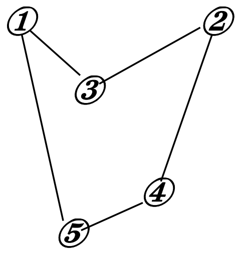
Minimum spanning tree:



Eulerian cycle:



Construction of the Eulerian cycle:
Choose $i_o = 1$.
1-3, i=3
3-2, i=2
Return to k=3 using 2-3, i=3
3-4, i=4
4-5, i=5
Return to k=4 using 5-4, i=4
Return to k=3 using 4-3, i=3
Return to k=1 using 3-1, $i=1=i_o$.
Eulerian cycle: 1-3-2-3-4-5-4-3-1.
Hamiltonian cycle with shortcuts: 1-3-2-4-5-1, f = 30



E. METHOD OF CHRISTOFIDES

1. Construct the minimum spannng tree T for the weighted graph G.
2. Identify the nodes with odd degree. Construct a minimum weight perfect matching for these nodes. Append the edges of the matching to the minimum spanning tree. Every node has an even degree. Generate an Eulerian cycle in this graph.
3. Form a Hamiltonian cyle using shortcuts as in the previous method.

Evaluation of the solution:	$f_a / f_{min} \leq 1.5$	This is the best worst case bound among nonexact methods!
Worst case complexity:	$T(n) = O(n^3)$	

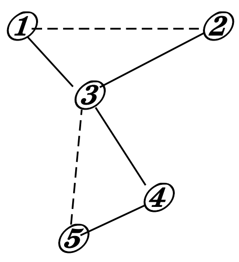
Experimental testing on the TSPLIB problems gives an average $f_a / f_{min} \approx 1.14$.

A minimum weight perfect matching in a complete graph of n vertices can be found in time $O(n^3)$. Some edges may double when appending the matching edges to the tree.

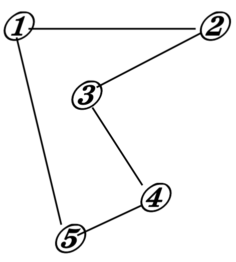
Example 5.5.

Start with the minimum spanning tree given in the previous example.

Nodes with odd degree: 1,2,3,5
Perfect matchings in the set {1,2,3,5}::
1-2, 3-5, weight 14 $\leftarrow$ smallest
1-3, 2-5, weight 14 $\leftarrow$ smallest
1-5, 2-3, weight 15
Add edges 1-2, 3-5 of the minimum weight matching to the spanning tree.
$\rightarrow$ Eulerian cycle: 1-2-3-4-5-3-1
Hamiltonian cycle using shortcuts: 1-2-3-4-5-1



Minimum spanning tree + matching edges



Solution: 1-2-3-4-5-1, f = 32

F. MERGING METHODS

METHOD OF NEAREST MERGER

Start with n one-node subtours. Find two cycles with minimum distance and merge them into one cycle in an optimal way. Continue until you get one cycle.

Evaluation of the solution: $f_a / f_{min} \leq 2$
Worst case complexity: $T(n) = O(n^2)$

ASSIGNMENT-BASED MERGING starts with a solution of the assignment relaxation. If separate subtours exist, two largest subtours are merged like in the previous method. This method is suitable mainly for asymmetric TSP.

5.3.2. IMPROVEMENT METHODS

Improvement methods start with a complete tour and try to improve it gradually. Many improvement methods use local search or neighborhood search.

LOCAL SEARCH = NEIGHBORHOOD SEARCH, a generic algorithm

1. Generate a feasible solution x.
2. Generate an unexamined neighbor y of x: $y \in N(x) \cap S$.
3. If $f(y) < f(x)$, set x:=y.
4. Stop, if all neighbors of x are examined without improvement: x is a locally optimal solution. Else, go to 2.

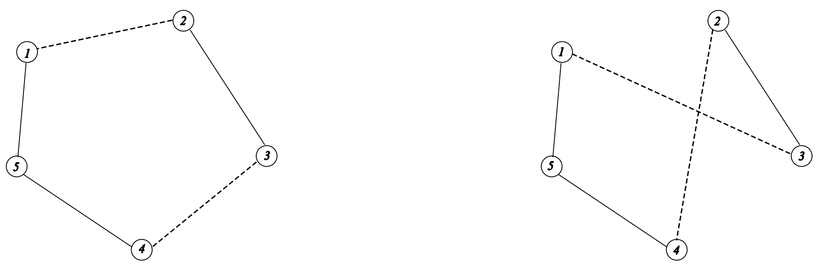
S = set of feasible solutions
N(x) = *neighborhood* of x, a set of neighboring solutions of x.

The definition of a neighborhood depends on the problem and it has often a strong effect on the quality of the solution.

Definition 5.1.
a) Operation *k-change* means deleting k edges from a tour and replacing them with k edges such that the result is a Hamiltonian cycle.
b) A *k-neighborhood* of a TSP-solution (Hamiltonian cycle) x is the set of Hamiltonian cycles that result from x with a k-change.
c) TSP tour (Hamiltonian cycle) is *k-optimal*, if it cannot be improved with any k-change.

We can say that any k-optimal solution is a locally optimal solution of the TSP with respect to the k-neighborhood.

Example of a 2-change: The edges marked with broken line to be changed:



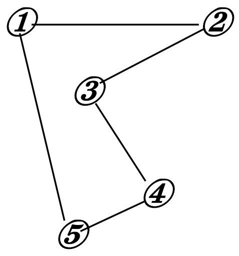
Notice that in the directed Hamiltonian cycle, some of the old edges change direction after the k-change.

ALGORITHM *k*-OPT:

1. Construct a Hamiltonian cycle x.
2. Execute a k-change, resulting in cycle y.
3. If f(y) < f(x), set x:=y.
4. Go to 2 until a local optimum is obtained.

This is a typical local search for TSP. In step 1, a construction method or random choice can be applied.

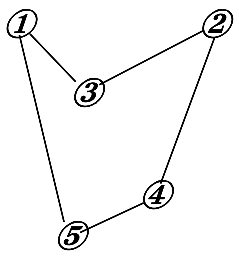
Example 5.5. Alogirthm 2-Opt
Starting cycle: T = 1-2-3-4-5-1, f = 32



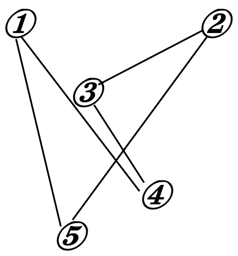
Possibilities for a 2-change:

Edges to be changed	Result
a) 1-2, 3-4	1-3-2-4-5-1 f=30
b) 1-2, 4-5	1-4-3-2-5-1 f=39
c) 2-3, 4-5	1-2-4-3-5-1 f=35
d) 2-3, 5-1	1-2-5-4-3-1 f=31
e) 3-4, 5-1	1-2-3-5-4-1 f=33

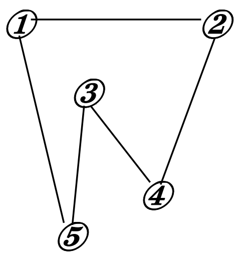
a)



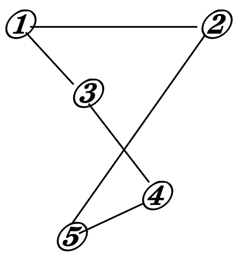
b)



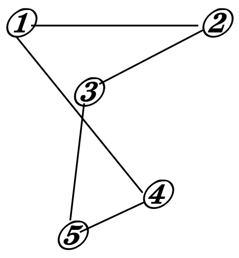
c)



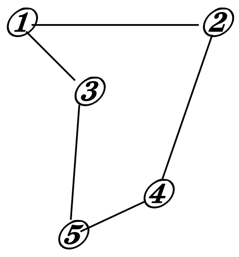
d)



e)



The first change improves the solution: T = 1-3-2-4-5-1
Changing edges 3-2 and 5-1 result in optimum: T = 1-2-4-5-3-1, f = 29



Typically small values of parameter k are used: k=2 or k=3. Then the change is relatively easy to perform and the neighborhood is small. Prosedure 3-Opt takes about n times as long as the 2-Opt prosedure. Also this algorithm can be repeated from different randomly generated starting solutions to improve the solution.

When k is icreased, the number of iterations increases but the solution improves. It has been shown experimentally that values of k>3 don't give further advantage to the quality of the solution. Best results are gained with methods where the value of k is dynamically changed.

COMPOSITE k-OPT HEURISTIC
1. Construct a Hamiltonian cycle x.
2. Apply 2-Opt algorithm to x, resulting solution y.
3. Apply 3-Opt algorithm to y.

Modifications:
• Repeat the composite procedure with varying construction methods. Random generation is recommended because it tends to cover the solution space more evenly.
• Repeat the construction step 1 from several starting vertex. Apply steps 2-3 to the best initial solution.

LIN-KERNIGHAN HEURISTIC
This is a very powerful method, based on k-Opt. Different versions of the Lin-Kernighan-methods exist, and they have given good, "near-optimal" results on empirical tests. The method is more complicated to code than the simple k-Opt.
(http://www.research.att.com/~dsj/papers/TSPchapter.pdf)

Several other heuristics are available for the TSP. Some special methods are introduced in the last chapter:

- Genetic Algorithms
- Simulated Annealing
- Tabu Search