March 25, 2018

## 0.1 IE 517 - Homework #1: Solving the TSP using Construction Heuristics and 2-Opt Improvement Heuristic

**Instructor:** Prof. Necati Aras

**Students:** Gönül Aycı - 2016800003, ÇağlaYaman - 2017800018

# Construction heuristics

Problem type : Shortest path problem
cities : $c_i$ = ($s_i$, $x_i$, $y_i$) i=0,1,2,…,n
n = number of cities
s = indis of a city s=[1,n]
x,y = place of a city in 2D coordinates
p = final path (n+1 cities)

objective function = $\min \sum_{i=0}^{n} d_{i(i+1)}$ d = distance between cities i and j

decision variables : $p_i = c_j$ if city $c_j$ is the $i^{th}$ element in the path p

constraints : $p_i \neq p_j$ where $i = 0$ and $j = n + 1$ (Every city should be visited only once except for the first city is also the last city in the tour)

Cities are represented as a class.

```
class CustomerLocation:
    #number of the city on the file
    indis = 0
    #coordianates of the cities
    x = 0
    y = 0

    def __init__(self, indis=None, x=None,
y=None):
        self.indis = indis
        self.x = x
        self.y = y
```

Distances between cities are calculated at initialization and kept in the distance matrix.

```
def calculateDistance(path):
    totalDistance = 0
    for i in range(len(path) -1):
        #use euclidean distance as the distance measure
        totalDistance += euclideanDistance(path[i], path[i + 1])
    return totalDistance
```

# Improvement heuristics

Solution representation : Permutation representation ($s_0$, $s_1$, ….. , $s_n$, $s_{n+1}$ )

Neighborhood structure : 2-opt neighborhood structure

# 1. One-sided insertion

Solve each instance using the one-sided nearest neighbor heuristic starting at cities 10, 20, and 30.

```python
# Creates route according to one-sided NN
# a: list of cities, index: first city
def oneSidedNN(index, a, distanceMatrix):
    #the final route that will be created
    path = []
    # start with a node which is given from instructor
    # current is always the last added city to the path
    current = a[index - 1]
    path.append(current)

    #while all cities are not visited
    while len(path) < len(a):
        # the minimum distance found so far
        candidateDistance = float('inf')
        # city wiht the min distance from the end of the path
        candidateNode = CustomerLocation()

        # loop all the cities and find the closest one to last added city
        for can in a:
            if can not in path:
                distance = distanceMatrix[current.indis - 1][can.indis - 1]
                if distance < candidateDistance:
                    candidateDistance = distance
                    candidateNode = can
        # add the closest city to the path
        path.append(candidateNode)
        current = candidateNode
    # return to the first city
    path.append(path[0])
    # calculate the total lenght of the path
    totalLength = round(calculateDistance(path), 2)
    return totalLength, path
```

# 2. Two-sided insertion

Solve each instance using the two-sided nearest neighbor heuristic starting at cities 10, 20, and 30.

```python
# Creates route according to two-sided NN
# a: list of cities, index: first city
def twoSidedNN(index, a, distanceMatrix):
    # initialize final route
    path = []
    # set the beginning of the route
    first_node = a[index - 1]
    path.append(first_node)
    # at the initial step first and last node is the same
    last_node = first_node

    # while every city is not visited
    while len(path) < len(a):
        #initialize candidate distance
        candidateDistance = float('inf')
        # candidate node is the closest node to beginning or the end of the route
        candidateNode = CustomerLocation()
        # distence from the first and last node are kept separately
        dist_begin = candidateDistance
        dist_end = candidateDistance

        # for every city not already in the path find the closest one
        for can in a:
            if can not in path:
                #distance from first node
                first_distance = distanceMatrix[first_node.indis - 1][can.indis - 1]
                #distance from last node
                last_distance = distanceMatrix[last_node.indis - 1][can.indis - 1]
                # set the smallest distance to the end
                if last_distance < candidateDistance:
                    candidateDistance = last_distance
                    candidateNode = can
                    dist_end = last_distance
                # set the smallest distance to the beginning
                if first_distance < candidateDistance and first_distance<last_distance:
                    candidateDistance = first_distance
                    candidateNode = can
                    dist_begin = first_distance
        # the priority to add is to the end of the route in case the distances are equal
        if (dist_begin < dist_end):
            # if the node is to be added to the beginning of the route
            # reverse the route
            path.reverse()
        # add the new node to the end
        path.append(candidateNode)
        #set the first and last node
        first_node = path[0]
        last_node = path[path.__len__()-1]
    totalLength = round(calculateDistance(path), 2)

    return totalLength, path
```

# 3. Nearest insertion

Solve each instance using the nearest insertion heuristic starting at cities 10, 20, and 30.

```
# Creates route according to nearest insertion
# a: list of cities, index: first city
def nearestInsertion(index, a, distanceMatrix):
    # initialize the final path
    path = []
    # start the tour from the given city
    current = a[index - 1]
    path.append(current)

    #keep a list of unvisited cities
    q = copy.copy(a)
    q.remove(current)

    candidateDistance = float("inf")
    candidateNode = CustomerLocation()

    # from unvisited cities, find the closest one to the initial city and add it to the path
    for candidate in q:
        if candidate not in path:
            distance = distanceMatrix[current.indis - 1][candidate.indis - 1]
            if distance < candidateDistance:
                candidateDistance = distance
                candidateNode = candidate
    path.append(candidateNode)
    #remove the visited cities
    q.remove(candidateNode)
    # create the edges list of th einitial tour
    edges = [(current, candidateNode), (candidateNode, current)]

    # while there is still unvisited cities, do the insertion process
    while len(q) > 0:
        # select a possible candidate to insert
        k = q[0]
        # initialize the distance
        crj = distanceMatrix[path[0].indis - 1][k.indis - 1]

        # find the node to be added to the path among unvisited cities
        for candidate in q:
            if candidate not in path:
                # check the distance of the candidate city to each of the visited cities
                for c in path:
                    distance = distanceMatrix[candidate.indis - 1][c.indis - 1]
                    if distance < crj:
                        # k is the nearest city
                        k = candidate

        # we will add k to path. Find place
        cost = float('inf')
        # the edge that will be eliminated
        edge_to_part = edges[0]
        # for each edge, calculate the cost added when the pat is inserted in this edge
        for edge in edges:
            # cost added is
            # total of distances of the candidate node to the begin and end of the edge is added to
the cost
            # length of the edge is reduced from the cost
            distance = distanceMatrix[edge[0].indis - 1][k.indis - 1] + distanceMatrix[k.indis - 1][
                edge[1].indis - 1] - distanceMatrix[edge[0].indis - 1][edge[1].indis - 1]
            # k will be added instead of the removed edge
            if distance < cost:
                cost = distance
                edge_to_part = edge

        # remove the edge
        edges.remove(edge_to_part)
        # add the edge between i and k (i is the beginning of the removed edge)
        edges.append((edge_to_part[0], k))
        # add the edge between j and k (j  is the end of the removed edge)
        edges.append((k, edge_to_part[1]))
        # the location in the path k will be added
        inst_loc = path.index(edge_to_part[1])
        # insert k to the path
        path.insert(inst_loc, k)
        # remove k from the unvisited city list
        q.remove(k)
    # go back to first city
    path.append(path[0])
    totalLength = calculateDistance(path)
    return totalLength, path
```

# 4. farthest insertion

Solve each instance using the farthest insertion heuristic starting at cities 10, 20, and 30.

```
# Creates route according to nearest insertion
# a: list of cities, index: first city
def farthestInsertion(index, a, distanceMatrix):
    # initialization part is the same with the nearest insertion
    path = []
    current = a[index - 1]
    path.append(current)

    q = copy.copy(a)
    q.remove(current)

    candidateDistance = float("-inf")
    candidateNode = CustomerLocation()

    # find the farthest node to the start node and add it to the path
    for candidate in q:
        if candidate not in path:
            distance = distanceMatrix[current.indis - 1][candidate.indis - 1]
            if (distance - candidateDistance) > 0.01:
                candidateDistance = distance
                candidateNode = candidate
    path.append(candidateNode)
    q.remove(candidateNode)
    # add the edges of the initial route
    edges = [(current, candidateNode), (candidateNode, current)]

    while len(q) > 0:
        k = q[0]
        crj = distanceMatrix[path[0].indis - 1][k.indis - 1]

        # find the fartest node to be added to the path
        for candidate in q:
            if candidate not in path:
                for c in path:
                    distance = distanceMatrix[candidate.indis - 1][c.indis - 1]
                    if (distance - crj) > 0.01:
                        k = candidate

        # we will add k to path. Find the place
        cost = float('inf')
        edge_to_part = edges[0]
        # for each edge, calculate the cost added when the pat is inserted in this edge
        for edge in edges:
            distance = distanceMatrix[edge[0].indis - 1][k.indis - 1] + distanceMatrix[k.indis - 1][
                edge[1].indis - 1] - distanceMatrix[edge[0].indis - 1][edge[1].indis - 1]
            # k with the smallest cost will be added instead of the removed edge
            if (distance - cost) < 0.01:
                cost = distance
                edge_to_part = edge

        # remove the edge
        edges.remove(edge_to_part)
        # add the edge between i and k (i is the beginning of the removed edge)
        edges.append((edge_to_part[0], k))
        # add the edge between j and k (j  is the end of the removed edge)
        edges.append((k, edge_to_part[1]))
        # the location in the path k will be added
        inst_loc = path.index(edge_to_part[1])
        # insert k to the path
        path.insert(inst_loc, k)
        # remove k from the unvisited city list
        q.remove(k)

    totalLength = round(calculateDistance(path), 2)
    return totalLength, path
```

# 5. 2-opt improvement

For each tour obtained so far, apply the 2-opt improvement heuristic.

```
def twoOptSolution(existingPath):
    improvement = True

    existingDistance = calculateDistance(existingPath)
    # while there is improvement execute 2-opt edge swap
    while improvement:
        improvement = False
        candidate_path = existingPath
        # keep start and end node unchanged.
        for i in range(1, len(existingPath) - 2):
            for k in range(i + 1, len(existingPath)-1):
                # swap two edges and create new path
                newPath = twoOptSwap(existingPath, i, k)
                # calculate the distance of the new path
                newDistance = calculateDistance(newPath)
                # find the shortest path
                if existingDistance - newDistance > 0.01:
                    existingDistance = newDistance
                    candidate_path = newPath
                    improvement = True
        # set the new shortest path as the solution and keep improving
        existingPath = candidate_path
    return existingDistance, existingPath
```

# Experiments

Experiments are run by executing test_tsp.py

---test with 51 nodes---

one sided insertion------10-------

initial

558.84

[10, 49, 9, 50, 16, 2, 29, 21, 34, 30, 39, 33, 45, 15, 44, 37, 17, 4, 18, 47, 12, 46, 51, 27, 1, 32, 11, 38, 5, 6, 48, 8, 26, 31, 28, 3, 20, 35, 36, 22, 7, 23, 24, 14, 25, 13, 41, 19, 42, 40, 43, 10]

2-opt improvement

436.85

[10, 30, 34, 21, 29, 2, 16, 50, 9, 49, 38, 5, 37, 17, 4, 18, 47, 12, 46, 51, 27, 11, 32, 1, 22, 20, 35, 36, 3, 28, 31, 26, 8, 48, 6, 23, 7, 43, 24, 14, 25, 13, 41, 40, 19, 42, 44, 15, 45, 33, 39, 10]

two sided insertion------10-------

initial

558.47

[30, 34, 21, 29, 2, 16, 50, 9, 49, 10, 39, 33, 45, 15, 44, 37, 17, 4, 18, 47, 12, 46, 51, 27, 1, 32, 11, 38, 5, 6, 48, 8, 26, 31, 28, 3, 20, 35, 36, 22, 7, 23, 24, 14, 25, 13, 41, 19, 42, 40, 43, 30]

2-opt improvement

440.52

[30, 34, 21, 29, 2, 22, 20, 35, 36, 3, 28, 31, 26, 8, 48, 6, 23, 7, 43, 24, 14, 25, 13, 41, 40, 19, 42, 44, 37, 17, 4, 18, 47, 12, 46, 51, 27, 1, 32, 11, 38, 5, 15, 45, 33, 39, 10, 49, 9, 16, 50, 30]

nearest insertion------10-------

initial

455.39

[10, 39, 30, 34, 50, 9, 38, 11, 2, 16, 21, 29, 20, 35, 36, 3, 28, 31, 26, 8, 22, 1, 32, 27, 6, 48, 23, 7, 43, 24, 14, 25, 18, 51, 46, 12, 47, 17, 4, 13, 41, 40, 19, 42, 44, 45, 33, 15, 37, 5, 49, 10]

2-opt improvement

```
455.39

[10, 39, 30, 34, 50, 9, 38, 11, 2, 16, 21, 29, 20, 35, 36, 3, 28, 31, 26, 8, 22, 1, 32,
27, 6, 48, 23, 7, 43, 24, 14, 25, 18, 51, 46, 12, 47, 17, 4, 13, 41, 40, 19, 42, 44, 45,
33, 15, 37, 5, 49, 10]

farthest insertion------10-------

initial

455.39

[24, 14, 25, 18, 51, 46, 12, 47, 17, 4, 13, 41, 40, 19, 42, 44, 45, 33, 15, 37, 5, 49,
10, 39, 30, 34, 50, 9, 38, 11, 2, 16, 21, 29, 20, 35, 36, 3, 28, 31, 26, 8, 22, 1, 32,
27, 6, 48, 23, 7, 43, 24]

2-opt improvement

455.39

[24, 14, 25, 18, 51, 46, 12, 47, 17, 4, 13, 41, 40, 19, 42, 44, 45, 33, 15, 37, 5, 49,
10, 39, 30, 34, 50, 9, 38, 11, 2, 16, 21, 29, 20, 35, 36, 3, 28, 31, 26, 8, 22, 1, 32,
27, 6, 48, 23, 7, 43, 24]

---test with 51 nodes---

one sided insertion------20-------

initial

567.29

[20, 35, 36, 3, 28, 31, 8, 26, 7, 23, 48, 27, 51, 46, 12, 47, 4, 17, 37, 15, 44, 42, 19,
41, 13, 25, 14, 6, 24, 43, 18, 5, 38, 11, 32, 1, 22, 2, 16, 50, 9, 49, 10, 30, 34, 21,
29, 39, 33, 45, 40, 20]

2-opt improvement

436.73

[20, 35, 36, 3, 28, 31, 8, 26, 7, 23, 43, 24, 14, 25, 13, 41, 40, 19, 42, 4, 18, 6, 48,
27, 51, 46, 12, 47, 17, 37, 44, 15, 45, 33, 39, 10, 49, 5, 38, 11, 32, 1, 22, 2, 16, 50,
9, 30, 34, 21, 29, 20]

two sided insertion------20-------
```

```
initial

545.56

[39, 33, 45, 5, 38, 22, 1, 32, 11, 2, 16, 34, 30, 10, 49, 9, 50, 21, 29, 36, 35, 20, 3,
28, 31, 8, 26, 7, 23, 48, 27, 51, 46, 12, 47, 4, 17, 37, 15, 44, 42, 19, 41, 13, 25, 14,
6, 24, 43, 18, 40, 39]

2-opt improvement

437.97

[39, 33, 45, 15, 44, 37, 17, 47, 12, 46, 51, 27, 48, 6, 18, 4, 42, 19, 40, 41, 13, 25,
14, 24, 43, 23, 7, 26, 8, 31, 28, 3, 36, 35, 20, 29, 21, 16, 2, 22, 1, 32, 11, 38, 5, 49,
9, 50, 34, 30, 10, 39]

nearest insertion------20-------

initial

460.63

[29, 21, 34, 50, 16, 2, 11, 5, 38, 9, 49, 10, 30, 39, 33, 45, 15, 37, 17, 44, 42, 19,
40, 41, 13, 25, 24, 43, 7, 23, 6, 14, 18, 4, 47, 12, 46, 51, 27, 48, 32, 1, 22, 8, 26,
31, 28, 3, 20, 36, 35, 29]

2-opt improvement

444.79

[29, 21, 34, 50, 16, 2, 11, 5, 38, 9, 49, 10, 30, 39, 33, 45, 15, 37, 17, 44, 42, 19,
40, 41, 13, 25, 24, 43, 7, 23, 48, 6, 14, 18, 4, 47, 12, 46, 51, 27, 32, 1, 22, 8, 26,
31, 28, 3, 36, 35, 20, 29]

farthest insertion------20-------

initial

451.25

[41, 13, 4, 17, 47, 12, 46, 51, 18, 25, 14, 24, 43, 7, 23, 48, 6, 27, 32, 1, 22, 8, 26,
31, 28, 3, 36, 35, 20, 29, 21, 34, 50, 16, 2, 11, 5, 38, 9, 49, 10, 30, 39, 33, 45, 15,
37, 44, 42, 19, 40, 41]

2-opt improvement
```

```
451.25

[41, 13, 4, 17, 47, 12, 46, 51, 18, 25, 14, 24, 43, 7, 23, 48, 6, 27, 32, 1, 22, 8, 26,
31, 28, 3, 36, 35, 20, 29, 21, 34, 50, 16, 2, 11, 5, 38, 9, 49, 10, 30, 39, 33, 45, 15,
37, 44, 42, 19, 40, 41]

---test with 51 nodes---

one sided insertion------30-------

initial

520.0

[30, 34, 50, 9, 49, 10, 39, 33, 45, 15, 44, 37, 17, 4, 18, 47, 12, 46, 51, 27, 1, 32,
11, 38, 5, 16, 2, 29, 21, 20, 35, 36, 3, 28, 31, 8, 26, 7, 23, 48, 6, 14, 25, 13, 41, 19,
42, 40, 24, 43, 22, 30]

2-opt improvement

444.12

[30, 34, 50, 21, 29, 20, 35, 36, 3, 28, 31, 8, 26, 7, 23, 43, 24, 6, 48, 27, 1, 22, 2,
16, 38, 5, 11, 32, 51, 46, 12, 47, 18, 14, 25, 13, 41, 40, 19, 42, 4, 17, 37, 44, 15,
45, 33, 39, 10, 49, 9, 30]

two sided insertion------30-------

initial

520.63

[40, 42, 19, 41, 13, 25, 14, 6, 48, 23, 7, 26, 8, 31, 28, 3, 36, 35, 20, 21, 29, 2, 16,
30, 34, 50, 9, 49, 10, 39, 33, 45, 15, 44, 37, 17, 4, 18, 47, 12, 46, 51, 27, 1, 32, 11,
38, 5, 22, 24, 43, 40]

2-opt improvement

444.21

[40, 41, 13, 25, 14, 18, 47, 12, 46, 51, 27, 48, 6, 24, 43, 23, 7, 26, 8, 31, 28, 3, 36,
35, 20, 29, 21, 2, 22, 1, 32, 11, 5, 38, 16, 50, 34, 30, 9, 49, 10, 39, 33, 45, 15, 44,
37, 17, 4, 42, 19, 40]

nearest insertion------30-------
```

```
initial

451.34

[30, 49, 10, 39, 33, 45, 15, 37, 44, 42, 19, 40, 41, 13, 4, 17, 47, 12, 46, 51, 18, 25,
14, 24, 43, 7, 23, 48, 6, 27, 32, 1, 22, 8, 26, 31, 28, 3, 36, 35, 20, 29, 21, 16, 2, 11,
5, 38, 9, 50, 34, 30]

2-opt improvement

451.34

[30, 49, 10, 39, 33, 45, 15, 37, 44, 42, 19, 40, 41, 13, 4, 17, 47, 12, 46, 51, 18, 25,
14, 24, 43, 7, 23, 48, 6, 27, 32, 1, 22, 8, 26, 31, 28, 3, 36, 35, 20, 29, 21, 16, 2, 11,
5, 38, 9, 50, 34, 30]

farthest insertion------30-------

initial

451.34

[24, 14, 25, 18, 51, 46, 12, 47, 17, 4, 13, 41, 40, 19, 42, 44, 37, 15, 45, 33, 39, 10,
49, 30, 34, 50, 9, 38, 5, 11, 2, 16, 21, 29, 20, 35, 36, 3, 28, 31, 26, 8, 22, 1, 32, 27,
6, 48, 23, 7, 43, 24]

2-opt improvement

451.34

[24, 14, 25, 18, 51, 46, 12, 47, 17, 4, 13, 41, 40, 19, 42, 44, 37, 15, 45, 33, 39, 10,
49, 30, 34, 50, 9, 38, 5, 11, 2, 16, 21, 29, 20, 35, 36, 3, 28, 31, 26, 8, 22, 1, 32, 27,
6, 48, 23, 7, 43, 24]

---test with 76 nodes---

one sided insertion------10-------

initial

640.48

[10, 58, 72, 39, 9, 32, 44, 3, 16, 63, 33, 73, 62, 28, 74, 30, 2, 68, 75, 76, 67, 34,
46, 8, 35, 7, 53, 14, 19, 54, 13, 27, 52, 45, 29, 48, 47, 21, 36, 71, 60, 70, 20, 37, 5,
15, 57, 4, 26, 12, 40, 17, 51, 6, 1, 43, 41, 42, 64, 22, 61, 69, 23, 56, 49, 24, 18, 50,
25, 55, 31, 38, 65, 11, 66, 59, 10]
```

2-opt improvement

571.23

[10, 58, 72, 39, 9, 32, 44, 3, 16, 63, 33, 73, 62, 2, 30, 74, 28, 21, 47, 48, 29, 45, 27, 52, 4, 76, 75, 68, 6, 51, 17, 40, 12, 26, 67, 34, 46, 8, 35, 7, 53, 14, 19, 54, 13, 57, 15, 5, 37, 20, 70, 60, 71, 36, 69, 61, 22, 64, 42, 41, 43, 1, 56, 23, 49, 24, 18, 50, 25, 55, 31, 66, 59, 11, 65, 38, 10]

two sided insertion------10-------

initial

699.02

[69, 61, 22, 64, 42, 41, 43, 1, 6, 51, 17, 40, 12, 26, 4, 57, 15, 5, 37, 20, 70, 60, 71, 36, 21, 47, 48, 29, 45, 52, 27, 13, 54, 19, 14, 53, 7, 35, 8, 46, 34, 67, 76, 75, 68, 2, 30, 74, 28, 62, 73, 33, 63, 16, 3, 44, 32, 9, 39, 72, 58, 10, 38, 65, 11, 66, 59, 31, 25, 50, 18, 24, 49, 23, 56, 55, 69]

2-opt improvement

586.75

[69, 61, 22, 64, 42, 41, 43, 1, 56, 23, 49, 24, 18, 50, 25, 55, 31, 66, 59, 11, 65, 38, 10, 58, 72, 39, 9, 32, 44, 3, 40, 12, 26, 17, 6, 51, 16, 63, 33, 73, 62, 28, 74, 30, 2, 68, 4, 75, 76, 67, 34, 46, 8, 35, 7, 53, 14, 19, 54, 13, 52, 27, 45, 29, 48, 21, 47, 36, 37, 5, 15, 57, 20, 70, 60, 71, 69]

nearest insertion------10-------

initial

591.4

[7, 35, 53, 14, 59, 11, 66, 65, 38, 10, 72, 39, 9, 31, 55, 25, 18, 50, 32, 44, 3, 24, 49, 16, 51, 6, 63, 33, 73, 1, 43, 23, 56, 41, 42, 64, 22, 62, 28, 74, 48, 47, 21, 61, 69, 36, 71, 60, 70, 20, 37, 5, 15, 57, 13, 54, 19, 8, 52, 46, 34, 27, 45, 29, 30, 2, 4, 68, 75, 76, 67, 26, 17, 40, 12, 58, 7]

2-opt improvement

579.03

```
[7, 35, 53, 14, 59, 11, 66, 65, 38, 10, 72, 39, 9, 31, 25, 55, 18, 50, 32, 44, 3, 24,
49, 16, 51, 6, 33, 63, 73, 1, 43, 23, 56, 41, 42, 64, 22, 62, 28, 74, 48, 47, 21, 61,
69, 36, 71, 60, 70, 20, 37, 5, 15, 57, 13, 54, 19, 8, 46, 34, 52, 27, 45, 29, 30, 2, 68,
4, 75, 76, 67, 26, 17, 40, 12, 58, 7]

farthest insertion------10-------

initial

572.27

[71, 36, 69, 61, 21, 47, 48, 29, 45, 30, 74, 28, 62, 22, 64, 42, 43, 41, 56, 23, 24,
49, 16, 63, 33, 1, 73, 2, 6, 68, 4, 75, 76, 51, 17, 12, 40, 3, 44, 32, 50, 18, 25, 55,
31, 9, 39, 72, 58, 10, 38, 65, 66, 11, 59, 14, 53, 35, 7, 26, 67, 34, 46, 52, 27, 8, 19,
54, 13, 57, 15, 5, 37, 20, 70, 60, 71]

2-opt improvement

571.17

[71, 36, 69, 61, 21, 47, 48, 29, 45, 30, 74, 28, 62, 22, 64, 42, 43, 41, 56, 23, 24,
49, 16, 63, 33, 1, 73, 2, 6, 68, 4, 75, 76, 51, 17, 12, 40, 3, 44, 32, 50, 18, 55, 25,
31, 9, 39, 72, 58, 10, 38, 65, 66, 11, 59, 14, 53, 35, 7, 26, 67, 34, 27, 52, 46, 8, 19,
54, 13, 57, 15, 5, 37, 20, 70, 60, 71]

---test with 76 nodes---

one sided insertion------20-------

initial

735.93

[20, 70, 60, 71, 36, 47, 21, 74, 28, 62, 73, 33, 63, 16, 51, 6, 68, 75, 76, 67, 34, 46,
8, 35, 7, 53, 14, 19, 54, 13, 27, 52, 45, 29, 48, 5, 37, 15, 57, 4, 30, 2, 1, 43, 41, 42,
64, 22, 61, 69, 26, 12, 40, 17, 3, 44, 32, 9, 39, 72, 58, 10, 38, 65, 11, 66, 59, 31,
25, 50, 18, 24, 49, 23, 56, 55, 20]

2-opt improvement

583.99

[20, 70, 60, 71, 69, 61, 22, 64, 42, 41, 43, 1, 73, 33, 16, 63, 23, 56, 49, 24, 18, 50,
25, 55, 31, 66, 59, 11, 65, 38, 10, 58, 72, 39, 9, 32, 44, 3, 40, 12, 26, 17, 51, 6, 68,
75, 76, 67, 34, 46, 8, 35, 7, 53, 14, 19, 54, 13, 57, 15, 37, 5, 48, 29, 45, 27, 52, 4,
30, 2, 62, 28, 74, 21, 47, 36, 20]

two sided insertion------20-------

initial

699.95

[69, 61, 22, 64, 42, 41, 43, 1, 4, 2, 30, 48, 15, 57, 13, 54, 19, 14, 53, 7, 35, 8, 46,
34, 52, 27, 45, 29, 5, 37, 20, 70, 60, 71, 36, 47, 21, 74, 28, 62, 73, 33, 63, 16, 51,
6, 68, 75, 76, 67, 26, 12, 40, 17, 3, 44, 32, 9, 39, 72, 58, 10, 38, 65, 11, 66, 59, 31,
25, 50, 18, 24, 49, 23, 56, 55, 69]

2-opt improvement

578.95

[69, 61, 22, 64, 42, 41, 43, 1, 56, 23, 49, 24, 18, 50, 25, 55, 31, 66, 59, 11, 65, 38,
10, 58, 72, 39, 9, 32, 44, 3, 17, 40, 12, 26, 67, 76, 75, 68, 6, 51, 16, 63, 33, 73, 62,
28, 74, 2, 4, 30, 48, 29, 45, 27, 52, 34, 46, 8, 35, 7, 53, 14, 19, 54, 13, 57, 15, 5,
37, 20, 70, 60, 71, 36, 47, 21, 69]

nearest insertion------20-------

initial

572.27

[20, 37, 5, 15, 57, 13, 54, 19, 8, 27, 52, 46, 34, 67, 26, 7, 35, 53, 14, 59, 11, 66,
65, 38, 10, 58, 72, 39, 9, 31, 55, 25, 18, 50, 32, 44, 3, 40, 12, 17, 51, 76, 75, 4, 68,
6, 2, 73, 1, 33, 63, 16, 49, 24, 23, 56, 41, 43, 42, 64, 22, 62, 28, 74, 30, 45, 29, 48,
47, 21, 61, 69, 36, 71, 60, 70, 20]

2-opt improvement

571.17

[20, 37, 5, 15, 57, 13, 54, 19, 8, 46, 52, 27, 34, 67, 26, 7, 35, 53, 14, 59, 11, 66,
65, 38, 10, 58, 72, 39, 9, 31, 25, 55, 18, 50, 32, 44, 3, 40, 12, 17, 51, 76, 75, 4, 68,
6, 2, 73, 1, 33, 63, 16, 49, 24, 23, 56, 41, 43, 42, 64, 22, 62, 28, 74, 30, 45, 29, 48,
47, 21, 61, 69, 36, 71, 60, 70, 20]

farthest insertion------20-------
```

```
initial

582.88

[25, 18, 50, 44, 3, 24, 49, 56, 23, 63, 16, 33, 73, 1, 43, 41, 42, 64, 22, 62, 2, 74,
28, 61, 21, 69, 71, 60, 70, 20, 57, 15, 5, 37, 36, 47, 48, 29, 45, 30, 4, 68, 6, 51, 17,
75, 76, 26, 67, 34, 46, 52, 27, 13, 54, 19, 8, 35, 7, 53, 14, 59, 11, 66, 65, 38, 10,
58, 72, 39, 12, 40, 32, 9, 31, 55, 25]

2-opt improvement

581.97

[25, 31, 9, 32, 40, 12, 39, 72, 58, 10, 38, 65, 66, 11, 59, 14, 53, 7, 35, 8, 19, 54,
13, 27, 52, 46, 34, 67, 75, 76, 26, 17, 51, 6, 68, 4, 30, 45, 29, 48, 47, 36, 37, 5, 15,
57, 20, 70, 60, 71, 69, 21, 61, 28, 74, 2, 62, 22, 64, 42, 41, 43, 1, 73, 33, 16, 63,
23, 56, 49, 24, 3, 44, 50, 18, 55, 25]

---test with 76 nodes---

one sided insertion------30-------

initial

730.23

[30, 74, 28, 62, 73, 33, 63, 16, 51, 6, 68, 75, 76, 67, 34, 46, 8, 35, 7, 53, 14, 19,
54, 13, 27, 52, 45, 29, 48, 47, 21, 36, 71, 60, 70, 20, 37, 5, 15, 57, 4, 26, 12, 40,
17, 3, 44, 32, 9, 39, 72, 58, 10, 38, 65, 11, 66, 59, 31, 25, 50, 18, 24, 49, 23, 56,
41, 42, 43, 1, 22, 61, 69, 2, 64, 55, 30]

2-opt improvement

569.96

[30, 74, 28, 21, 47, 48, 29, 45, 27, 52, 4, 76, 75, 68, 6, 51, 17, 40, 12, 26, 67, 34,
46, 8, 35, 7, 53, 14, 19, 54, 13, 57, 15, 5, 37, 20, 70, 60, 71, 36, 69, 61, 22, 1, 43,
42, 64, 41, 56, 23, 49, 24, 18, 50, 25, 55, 31, 66, 59, 11, 65, 38, 10, 58, 72, 39, 9,
32, 44, 3, 16, 63, 33, 73, 62, 2, 30]

two sided insertion------30-------

initial

662.86
```

[59, 66, 11, 65, 38, 10, 31, 55, 25, 50, 18, 24, 49, 3, 44, 32, 9, 39, 72, 58, 4, 26, 12, 40, 17, 6, 51, 16, 63, 33, 73, 62, 28, 74, 30, 2, 68, 75, 76, 67, 34, 46, 8, 35, 7, 53, 14, 19, 54, 13, 27, 52, 45, 29, 48, 47, 21, 36, 71, 60, 70, 20, 37, 5, 15, 57, 69, 61, 22, 1, 43, 41, 42, 64, 56, 23, 59]

2-opt improvement

567.08

[59, 11, 66, 65, 38, 10, 31, 55, 25, 50, 18, 24, 49, 23, 56, 41, 64, 42, 43, 1, 22, 61, 69, 71, 60, 70, 20, 57, 15, 5, 37, 36, 47, 21, 48, 29, 45, 27, 52, 13, 54, 19, 35, 8, 46, 34, 4, 75, 76, 67, 26, 12, 40, 17, 51, 6, 68, 2, 30, 74, 28, 62, 73, 33, 63, 16, 3, 44, 32, 9, 39, 72, 58, 7, 53, 14, 59]

nearest insertion------30-------

initial

576.18

[28, 61, 21, 69, 36, 71, 60, 70, 20, 37, 15, 57, 29, 5, 47, 48, 30, 45, 4, 76, 26, 17, 12, 40, 3, 44, 32, 50, 18, 25, 55, 31, 9, 39, 72, 58, 10, 38, 65, 66, 11, 59, 14, 53, 7, 35, 8, 19, 54, 13, 27, 52, 46, 34, 67, 75, 68, 6, 51, 33, 73, 1, 63, 16, 49, 24, 23, 56, 41, 43, 42, 64, 22, 62, 2, 74, 28]

2-opt improvement

571.37

[28, 61, 21, 69, 36, 71, 60, 70, 20, 37, 15, 57, 29, 5, 47, 48, 30, 45, 4, 67, 34, 46, 52, 27, 13, 54, 19, 8, 35, 7, 53, 14, 59, 11, 66, 65, 38, 10, 58, 72, 39, 9, 31, 25, 55, 18, 50, 32, 44, 3, 40, 12, 17, 26, 76, 75, 68, 6, 51, 33, 73, 1, 63, 16, 49, 24, 23, 56, 41, 43, 42, 64, 22, 62, 2, 74, 28]

farthest insertion------30-------

initial

580.24

[31, 9, 32, 40, 12, 39, 72, 58, 10, 38, 65, 66, 11, 59, 14, 53, 7, 35, 8, 19, 54, 13, 27, 52, 46, 34, 67, 26, 76, 75, 17, 51, 6, 68, 2, 30, 4, 45, 29, 48, 47, 5, 15, 57, 37, 20, 70, 60, 71, 36, 69, 21, 61, 28, 74, 62, 22, 64, 42, 41, 43, 1, 73, 33, 16, 63, 23, 56, 49, 24, 3, 44, 50, 18, 25, 55, 31]

2-opt improvement

579.33

[31, 9, 32, 40, 12, 39, 72, 58, 10, 38, 65, 66, 11, 59, 14, 53, 7, 35, 8, 19, 54, 13, 27, 52, 46, 34, 67, 75, 76, 26, 17, 51, 6, 68, 2, 30, 4, 45, 29, 48, 47, 5, 15, 57, 37, 20, 70, 60, 71, 36, 69, 21, 61, 28, 74, 62, 22, 64, 42, 41, 43, 1, 73, 33, 16, 63, 23, 56, 49, 24, 3, 44, 50, 18, 55, 25, 31]

---test with 101 nodes---

one sided insertion------10-------

initial

795.98

[10, 62, 88, 31, 70, 30, 20, 51, 9, 81, 33, 79, 3, 77, 76, 50, 1, 69, 27, 101, 53, 58, 40, 21, 73, 72, 74, 22, 75, 56, 39, 23, 67, 25, 55, 54, 80, 68, 12, 26, 28, 89, 6, 94, 95, 97, 92, 59, 99, 96, 93, 85, 91, 100, 37, 98, 61, 16, 44, 14, 42, 87, 2, 57, 15, 43, 38, 86, 17, 84, 5, 60, 83, 18, 52, 7, 82, 48, 47, 36, 49, 19, 11, 63, 90, 32, 66, 71, 35, 34, 78, 29, 24, 4, 41, 13, 8, 45, 46, 64, 65, 10]

2-opt improvement

666.79

[10, 62, 88, 31, 70, 30, 20, 51, 9, 81, 33, 79, 3, 77, 76, 50, 1, 69, 27, 101, 53, 28, 26, 12, 68, 80, 54, 4, 72, 74, 73, 21, 40, 58, 13, 87, 42, 14, 44, 16, 61, 85, 91, 100, 37, 98, 93, 99, 96, 59, 92, 97, 95, 94, 6, 89, 18, 52, 7, 82, 48, 19, 11, 64, 49, 36, 47, 46, 8, 45, 83, 60, 5, 84, 17, 86, 38, 43, 15, 57, 2, 41, 22, 75, 56, 23, 67, 39, 25, 55, 24, 29, 78, 34, 35, 71, 65, 66, 32, 90, 63, 10]

two sided insertion------10-------

initial

816.6

[65, 66, 71, 35, 34, 78, 29, 24, 4, 28, 26, 12, 68, 80, 54, 55, 25, 67, 23, 39, 56, 75, 22, 74, 72, 73, 21, 40, 58, 53, 101, 27, 69, 1, 50, 76, 77, 3, 79, 33, 81, 9, 51, 20, 30, 70, 31, 88, 62, 10, 90, 32, 63, 11, 19, 47, 48, 82, 7, 52, 18, 60, 83, 84, 5, 99, 59, 92, 37, 98, 85, 93, 96, 94, 6, 89, 13, 95, 97, 87, 2, 57, 15, 43, 42, 100, 91, 16, 61, 17, 45, 8, 46, 36, 49, 64, 86, 44, 14, 38, 41, 65]

2-opt improvement

674.17

[65, 66, 32, 90, 10, 62, 88, 31, 70, 30, 20, 51, 9, 81, 33, 79, 3, 77, 76, 50, 1, 69, 27, 89, 6, 94, 96, 99, 5, 84, 83, 60, 18, 52, 7, 82, 48, 47, 19, 11, 63, 64, 49, 36, 46, 8, 45, 17, 61, 16, 86, 38, 44, 14, 43, 42, 100, 91, 85, 98, 37, 93, 59, 92, 97, 95, 13, 87, 2, 57, 15, 41, 22, 75, 56, 23, 67, 39, 25, 55, 4, 72, 74, 73, 21, 40, 58, 53, 101, 28, 26, 12, 68, 80, 54, 24, 29, 78, 34, 35, 71, 65]

nearest insertion------10-------

initial

710.07

[7, 48, 82, 8, 45, 83, 60, 18, 89, 6, 96, 99, 59, 93, 98, 85, 61, 5, 84, 17, 86, 38, 14, 44, 16, 91, 100, 37, 92, 97, 87, 42, 43, 15, 57, 2, 41, 22, 75, 23, 39, 67, 25, 55, 54, 4, 56, 74, 72, 73, 21, 26, 40, 58, 13, 95, 94, 53, 101, 28, 50, 76, 12, 80, 68, 77, 3, 24, 29, 79, 78, 34, 81, 33, 51, 9, 71, 35, 65, 66, 20, 32, 30, 70, 1, 69, 27, 52, 88, 31, 10, 90, 63, 64, 49, 36, 47, 46, 19, 11, 62, 7]

2-opt improvement

696.94

[7, 48, 82, 8, 45, 83, 60, 18, 89, 6, 96, 99, 59, 93, 98, 85, 61, 5, 84, 17, 86, 38, 14, 44, 16, 91, 100, 37, 92, 97, 87, 42, 43, 15, 57, 2, 41, 22, 75, 23, 67, 39, 25, 55, 54, 4, 56, 74, 72, 73, 21, 26, 40, 58, 13, 95, 94, 53, 101, 28, 12, 76, 50, 3, 77, 68, 80, 24, 29, 79, 78, 34, 81, 33, 51, 9, 35, 71, 65, 66, 20, 32, 30, 70, 1, 69, 27, 52, 88, 31, 10, 90, 63, 64, 49, 36, 46, 47, 19, 11, 62, 7]

farthest insertion------10-------

initial

710.53

[25, 55, 54, 4, 56, 72, 21, 26, 40, 58, 53, 101, 28, 27, 69, 1, 50, 76, 12, 80, 68, 77, 3, 24, 29, 79, 78, 34, 81, 33, 51, 9, 71, 35, 65, 66, 20, 30, 70, 32, 90, 63, 64, 49, 36, 47, 46, 19, 11, 31, 10, 62, 88, 7, 48, 82, 18, 52, 89, 6, 94, 96, 99, 59, 93, 98, 85, 91, 44, 14, 38, 86, 17, 84, 45, 8, 83, 60, 5, 61, 16, 100, 37, 92, 97, 95, 13, 87, 42, 43, 15, 57, 2, 41, 22, 73, 74, 75, 23, 39, 67, 25]

2-opt improvement

```
675.88

[25, 55, 54, 4, 56, 75, 74, 72, 73, 21, 26, 40, 58, 53, 101, 28, 27, 69, 1, 50, 76, 12,
77, 3, 68, 80, 24, 29, 79, 78, 34, 81, 33, 51, 9, 35, 71, 65, 66, 20, 30, 70, 32, 90,
63, 64, 49, 36, 46, 47, 19, 11, 62, 10, 31, 88, 7, 48, 82, 8, 45, 84, 17, 86, 38, 14,
44, 16, 61, 5, 60, 83, 18, 52, 89, 6, 94, 96, 99, 59, 93, 98, 85, 91, 100, 37, 92, 97,
95, 13, 87, 42, 43, 15, 57, 2, 41, 22, 23, 67, 39, 25]

---test with 101 nodes---

one sided insertion------20-------

initial

800.67

[20, 30, 70, 31, 88, 7, 82, 48, 47, 36, 49, 19, 11, 62, 10, 90, 32, 63, 64, 46, 8, 45,
17, 84, 5, 60, 83, 18, 52, 89, 6, 94, 95, 97, 92, 59, 99, 96, 93, 85, 91, 100, 37, 98,
61, 16, 44, 14, 42, 87, 2, 57, 15, 43, 38, 86, 13, 58, 40, 53, 101, 27, 28, 26, 12, 80,
68, 77, 3, 79, 33, 81, 9, 51, 50, 76, 1, 69, 78, 34, 35, 71, 66, 65, 29, 24, 54, 55, 25,
39, 56, 75, 74, 22, 41, 73, 21, 72, 4, 23, 67, 20]

2-opt improvement

659.73

[20, 66, 65, 71, 35, 34, 78, 29, 24, 54, 55, 25, 39, 67, 23, 56, 75, 41, 22, 74, 73,
21, 72, 4, 26, 28, 27, 101, 53, 40, 58, 13, 2, 87, 42, 57, 15, 43, 14, 44, 38, 86, 16,
61, 85, 91, 100, 37, 98, 93, 99, 96, 59, 92, 97, 95, 94, 6, 89, 52, 18, 83, 60, 5, 84,
17, 45, 8, 46, 47, 36, 49, 64, 63, 90, 32, 10, 62, 11, 19, 48, 82, 7, 88, 31, 70, 30,
69, 1, 50, 76, 12, 80, 68, 77, 3, 79, 33, 81, 9, 51, 20]

two sided insertion------20-------

initial

833.54

[46, 45, 8, 52, 18, 83, 60, 5, 84, 17, 86, 38, 43, 15, 57, 2, 87, 42, 14, 44, 16, 61,
98, 37, 100, 91, 85, 93, 96, 99, 59, 92, 97, 95, 94, 6, 89, 28, 26, 12, 68, 80, 54, 55,
25, 67, 23, 39, 56, 75, 22, 74, 72, 73, 21, 40, 58, 53, 101, 27, 69, 1, 50, 76, 77, 3,
79, 33, 81, 9, 51, 20, 30, 70, 31, 88, 7, 82, 48, 47, 36, 49, 19, 11, 62, 10, 90, 32,
63, 64, 66, 71, 35, 34, 78, 29, 24, 4, 41, 13, 65, 46]

2-opt improvement
```

678.34

[46, 45, 83, 60, 5, 84, 17, 86, 38, 43, 15, 57, 2, 41, 22, 75, 56, 23, 67, 39, 25, 55, 24, 29, 78, 34, 35, 71, 65, 66, 32, 90, 10, 62, 63, 64, 11, 19, 49, 36, 47, 48, 82, 7, 88, 31, 70, 30, 20, 51, 9, 81, 33, 79, 3, 77, 76, 50, 1, 69, 27, 101, 53, 28, 26, 12, 68, 80, 54, 4, 72, 74, 73, 21, 40, 58, 13, 87, 42, 14, 44, 16, 61, 85, 91, 100, 37, 98, 93, 99, 96, 59, 92, 97, 95, 94, 6, 89, 52, 18, 8, 46]

nearest insertion------20-------

initial

737.87

[51, 33, 50, 76, 77, 12, 54, 4, 25, 55, 24, 80, 68, 29, 3, 79, 78, 81, 9, 34, 35, 71, 65, 20, 66, 32, 90, 63, 64, 11, 62, 19, 47, 49, 36, 46, 48, 82, 18, 83, 8, 45, 17, 84, 5, 60, 96, 94, 95, 87, 57, 2, 73, 21, 72, 74, 56, 39, 67, 23, 75, 22, 41, 15, 43, 42, 97, 92, 37, 100, 91, 44, 14, 38, 86, 16, 61, 85, 98, 93, 59, 99, 6, 89, 13, 58, 40, 26, 53, 101, 28, 27, 1, 69, 52, 7, 88, 31, 10, 70, 30, 51]

2-opt improvement

708.32

[51, 33, 50, 76, 77, 68, 80, 12, 54, 4, 25, 55, 24, 29, 3, 79, 78, 81, 9, 34, 35, 71, 65, 66, 20, 32, 90, 63, 64, 11, 62, 19, 47, 49, 36, 46, 48, 82, 18, 83, 8, 45, 17, 84, 5, 60, 89, 6, 94, 95, 96, 99, 59, 93, 98, 85, 61, 16, 86, 38, 14, 44, 91, 100, 37, 92, 97, 42, 43, 15, 41, 22, 75, 23, 67, 39, 56, 74, 72, 21, 73, 2, 57, 87, 13, 58, 40, 26, 53, 101, 27, 28, 1, 69, 52, 7, 88, 31, 10, 70, 30, 51]

farthest insertion------20-------

initial

699.66

[14, 44, 100, 98, 37, 92, 97, 87, 42, 43, 15, 57, 2, 41, 22, 75, 23, 39, 67, 25, 55, 54, 4, 56, 74, 72, 73, 21, 40, 58, 13, 95, 94, 6, 89, 101, 53, 26, 28, 50, 76, 12, 80, 68, 77, 3, 24, 29, 79, 78, 34, 81, 33, 51, 9, 71, 35, 65, 66, 20, 30, 32, 90, 63, 64, 49, 36, 47, 46, 19, 11, 62, 10, 31, 70, 1, 69, 27, 52, 88, 7, 48, 82, 18, 83, 8, 45, 17, 84, 5, 60, 96, 99, 59, 93, 85, 91, 61, 16, 86, 38, 14]

2-opt improvement

683.53

```
[14, 44, 100, 98, 37, 92, 97, 87, 42, 43, 15, 57, 2, 41, 22, 75, 23, 67, 39, 25, 55,
54, 4, 56, 74, 72, 73, 21, 40, 58, 13, 95, 94, 6, 89, 101, 53, 28, 26, 12, 76, 50, 3,
77, 68, 80, 24, 29, 79, 78, 34, 81, 33, 51, 9, 35, 71, 65, 66, 20, 30, 32, 90, 63, 64,
49, 36, 46, 47, 19, 11, 62, 10, 31, 70, 1, 69, 27, 52, 88, 7, 48, 82, 18, 83, 8, 45, 17,
84, 5, 60, 96, 99, 59, 93, 85, 91, 61, 16, 86, 38, 14]

---test with 101 nodes---

one sided insertion------30-------

initial

776.46

[30, 70, 31, 88, 7, 82, 48, 47, 36, 49, 19, 11, 62, 10, 90, 32, 63, 64, 46, 8, 45, 17,
84, 5, 60, 83, 18, 52, 89, 6, 94, 95, 97, 92, 59, 99, 96, 93, 85, 91, 100, 37, 98, 61,
16, 44, 14, 42, 87, 2, 57, 15, 43, 38, 86, 13, 58, 40, 53, 101, 27, 28, 26, 12, 80, 68,
77, 3, 79, 33, 81, 9, 51, 20, 66, 71, 35, 34, 78, 29, 24, 54, 55, 25, 39, 56, 75, 74,
22, 41, 73, 21, 72, 4, 23, 67, 76, 50, 1, 69, 65, 30]

2-opt improvement

659.73

[30, 70, 31, 88, 7, 82, 48, 19, 11, 62, 10, 32, 90, 63, 64, 49, 36, 47, 46, 8, 45, 17,
84, 5, 60, 83, 18, 52, 89, 6, 94, 95, 97, 92, 59, 96, 99, 93, 98, 37, 100, 91, 85, 61,
16, 86, 38, 44, 14, 43, 15, 57, 42, 87, 2, 13, 58, 40, 53, 101, 27, 28, 26, 4, 72, 21,
73, 74, 22, 41, 75, 56, 23, 67, 39, 25, 55, 54, 24, 29, 78, 34, 35, 71, 65, 66, 20, 51,
9, 81, 33, 79, 3, 77, 68, 80, 12, 76, 50, 1, 69, 30]

two sided insertion------30-------

initial

833.54

[46, 45, 8, 52, 18, 83, 60, 5, 84, 17, 86, 38, 43, 15, 57, 2, 87, 42, 14, 44, 16, 61,
98, 37, 100, 91, 85, 93, 96, 99, 59, 92, 97, 95, 94, 6, 89, 28, 26, 12, 68, 80, 54, 55,
25, 67, 23, 39, 56, 75, 22, 74, 72, 73, 21, 40, 58, 53, 101, 27, 69, 1, 50, 76, 77, 3,
79, 33, 81, 9, 51, 20, 30, 70, 31, 88, 7, 82, 48, 47, 36, 49, 19, 11, 62, 10, 90, 32,
63, 64, 66, 71, 35, 34, 78, 29, 24, 4, 41, 13, 65, 46]

2-opt improvement

678.34
```

```
[46, 45, 83, 60, 5, 84, 17, 86, 38, 43, 15, 57, 2, 41, 22, 75, 56, 23, 67, 39, 25, 55,
24, 29, 78, 34, 35, 71, 65, 66, 32, 90, 10, 62, 63, 64, 11, 19, 49, 36, 47, 48, 82, 7,
88, 31, 70, 30, 20, 51, 9, 81, 33, 79, 3, 77, 76, 50, 1, 69, 27, 101, 53, 28, 26, 12,
68, 80, 54, 4, 72, 74, 73, 21, 40, 58, 13, 87, 42, 14, 44, 16, 61, 85, 91, 100, 37, 98,
93, 99, 96, 59, 92, 97, 95, 94, 6, 89, 52, 18, 8, 46]

nearest insertion------30-------

initial

716.74

[30, 51, 33, 81, 9, 20, 66, 65, 71, 35, 34, 78, 79, 3, 29, 68, 80, 24, 55, 25, 54, 12,
77, 76, 50, 1, 69, 27, 28, 101, 53, 26, 40, 21, 72, 4, 39, 67, 23, 56, 75, 41, 22, 74,
73, 58, 13, 94, 95, 87, 2, 57, 15, 43, 42, 97, 92, 37, 100, 91, 85, 98, 93, 61, 16, 44,
14, 38, 86, 17, 84, 45, 8, 83, 60, 5, 99, 59, 96, 6, 89, 52, 18, 82, 48, 46, 36, 49, 47,
19, 7, 88, 31, 10, 62, 11, 64, 63, 90, 32, 70, 30]

2-opt improvement

684.45

[30, 51, 33, 81, 9, 20, 66, 65, 71, 35, 34, 78, 79, 3, 77, 76, 50, 1, 69, 27, 28, 101,
53, 26, 12, 80, 68, 29, 24, 54, 55, 25, 4, 39, 67, 23, 56, 75, 41, 22, 74, 72, 73, 21,
40, 58, 13, 94, 95, 87, 2, 57, 15, 43, 42, 97, 92, 37, 100, 91, 85, 98, 93, 61, 16, 44,
14, 38, 86, 17, 84, 45, 8, 83, 60, 5, 99, 59, 96, 6, 89, 52, 18, 82, 48, 46, 36, 49, 47,
19, 7, 88, 31, 10, 62, 11, 64, 63, 90, 32, 70, 30]

farthest insertion------30-------

initial

694.96

[14, 44, 100, 98, 37, 92, 97, 87, 42, 43, 15, 57, 2, 41, 22, 75, 23, 39, 67, 25, 55,
54, 4, 56, 74, 72, 73, 21, 26, 40, 58, 13, 95, 94, 6, 89, 53, 101, 28, 27, 69, 1, 50,
76, 12, 80, 68, 77, 3, 24, 29, 79, 78, 34, 81, 33, 51, 9, 71, 35, 65, 66, 20, 30, 70,
10, 32, 90, 63, 64, 49, 36, 47, 46, 19, 11, 62, 88, 31, 52, 18, 7, 48, 82, 8, 83, 45,
17, 84, 5, 60, 96, 99, 59, 93, 85, 91, 61, 16, 86, 38, 14]

2-opt improvement

683.83
```

[14, 44, 100, 98, 37, 92, 97, 87, 42, 43, 15, 57, 2, 41, 22, 75, 23, 67, 39, 25, 55, 54, 4, 56, 74, 72, 73, 21, 26, 40, 58, 13, 95, 94, 6, 89, 101, 53, 28, 27, 69, 1, 50, 76, 12, 77, 3, 68, 80, 24, 29, 79, 78, 34, 81, 33, 51, 9, 35, 71, 65, 66, 20, 30, 70, 10, 32, 90, 63, 64, 49, 36, 46, 47, 19, 11, 62, 88, 31, 52, 18, 7, 48, 82, 8, 83, 45, 17, 84, 5, 60, 96, 99, 59, 93, 85, 91, 61, 16, 86, 38, 14]

| | | | Tour Lengths (% Deviation from Optimal Tour Length | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Instance | Opt. Tour Length | Initial cust. | 1-sided NN | After 2-opt | 2-sided NN | After 2-opt | Nearest Insert | After 2-opt | Farthest Insert | After 2-opt |
| eil51 | 429,97 | 10 | 558,84 | 436,85 | 558,47 | 440,52 | 455,39 | 455,39 | 455,39 | 455,39 |
| eil51 | 429,97 | 20 | 567,29 | 436,73 | 545,56 | 437,97 | 460,63 | 444,79 | 451,25 | 451,25 |
| eil51 | 429,97 | 30 | 520,00 | 444,12 | 520,63 | 444,21 | 451,34 | 451,34 | 451,34 | 451,34 |
| eil76 | 545,34 | 10 | 640,48 | 571,23 | 699,02 | 586,75 | 591,40 | 579,03 | 572,27 | 571,17 |
| eil76 | 545,34 | 20 | 735,93 | 583,99 | 699,95 | 578,95 | 572,27 | 571,17 | 582,88 | 581,97 |
| eil76 | 545,34 | 30 | 730,23 | 569,96 | 662,86 | 567,08 | 576,18 | 571,37 | 580,24 | 579,33 |
| eil101 | 642,23 | 10 | 795,98 | 666,79 | 816,60 | 674,17 | 710,07 | 696,94 | 710,53 | 675,88 |
| eil101 | 642,23 | 20 | 800,67 | 659,73 | 833,54 | 678,34 | 737,87 | 708,32 | 699,66 | 683,53 |
| eil101 | 642,23 | 30 | 776,46 | 659,73 | 833,54 | 678,34 | 716,74 | 684,45 | 694,96 | 683.83 |
| Average | 539,18 | | 680,65 | 558,79 | 685,57 | 565,15 | 585,77 | 573,64 | 577,61 | 556,23 |