Modelling, Simulation & Optimisation (H9MSO)

2. Integer Programming

Outline of Week 2

- Principles of Integer Programming
- Using PuLP
- The manufacturing problem from last week, this time in PuLP
- The n-Queens Problem
- Logical Constraints
- The Travelling Salesman Problem
- A Knapsack Problem
- Cutting Stock Problem

The travelling salesman problem (TSP)

- The TSP is a very well-known problem that has been attacked with all sorts of methods.
- We have a set of cities, or (more generally) nodes in a graph. Between each pair of nodes is an edge with a distance attached to it.
- In some cases, the distance between city A and city B may be different than the distance from city B to city A: this is the *asymmetric* TSP.
- The problem is to plan a tour that visits each city exactly once, finishing up back where we started, while minimising the total distance travelled. (Not all the edges will be used in any tour.)

The travelling salesman problem (TSP)

- If the distance matrix is C_{ij} then we must minimize

$$Z = \sum_i \sum_j C_{ij} x_{ij}$$

Which we describe in pulp as:

```
prob = pulp.LpProblem("TSP",pulp.LpMinimize)
prob += pulp.lpSum([ distance(i,j)*x[i][j]
                       for i in range(0,n)
                         for j in range(0,n)
                           if i!=j
                   ])
```

with:

```
def distance(i, j):
    return g7.loc[i][capitals[j]]
```

The travelling salesman problem (TSP)

- Constraints:
  Each node i has exactly one successor (from every city we must travel to exactly one other):

$$\sum_j x_{ij} = 1$$

Which we describe in pulp as:

```
for i in range(0,n):
   prob += pulp.lpSum([ x[i][j]
                        for j in range(0,n)
                           if i!=j
                   ]) ==1
```

The travelling salesman problem (TSP)

- Constraints:
  Each node j has exactly one successor (from every city we must travel to exactly one other):

$$\sum_i x_{ij} = 1$$

Which we describe in pulp as:

```
for j in range(0,n):
   prob += pulp.lpSum([ x[i][j]
                         for i in range(0,n)
                            if i!=j
                    ]) ==1
```

The travelling salesman problem (TSP)

- The constraint that each node (city) must be visited exactly once is not easy to represent directly.
- The standard way is to prevent *subtours*, that is visits to *some* of the nodes that form a cycle. For every possible subtour, represented by a set S of edges, we add a constraint

$$\sum_{(i,j)\in S} x_{ij} \leq |S| - 1$$

- That is, not all the edges in the subtour can be used.

- Actually, we only need to consider subtours of size up to $n/2$ (where n is the number of cities), because if there is a larger subtour then the rest of the nodes must be contained in a smaller one (this might need some thought).

DEMO

The travelling salesman problem (TSP)

- The drawback with this IP is that there are a lot of subtour elimination constraints: $2^{n-1} - n - 1$ of them. So this is not practical except for small TSPs. At least not with the usual solution methods.

- But there is in fact a way of solving large TSPs with this model: instead of putting all the subtour elimination constraints into a file before solving the IP, they are generated as needed during search.

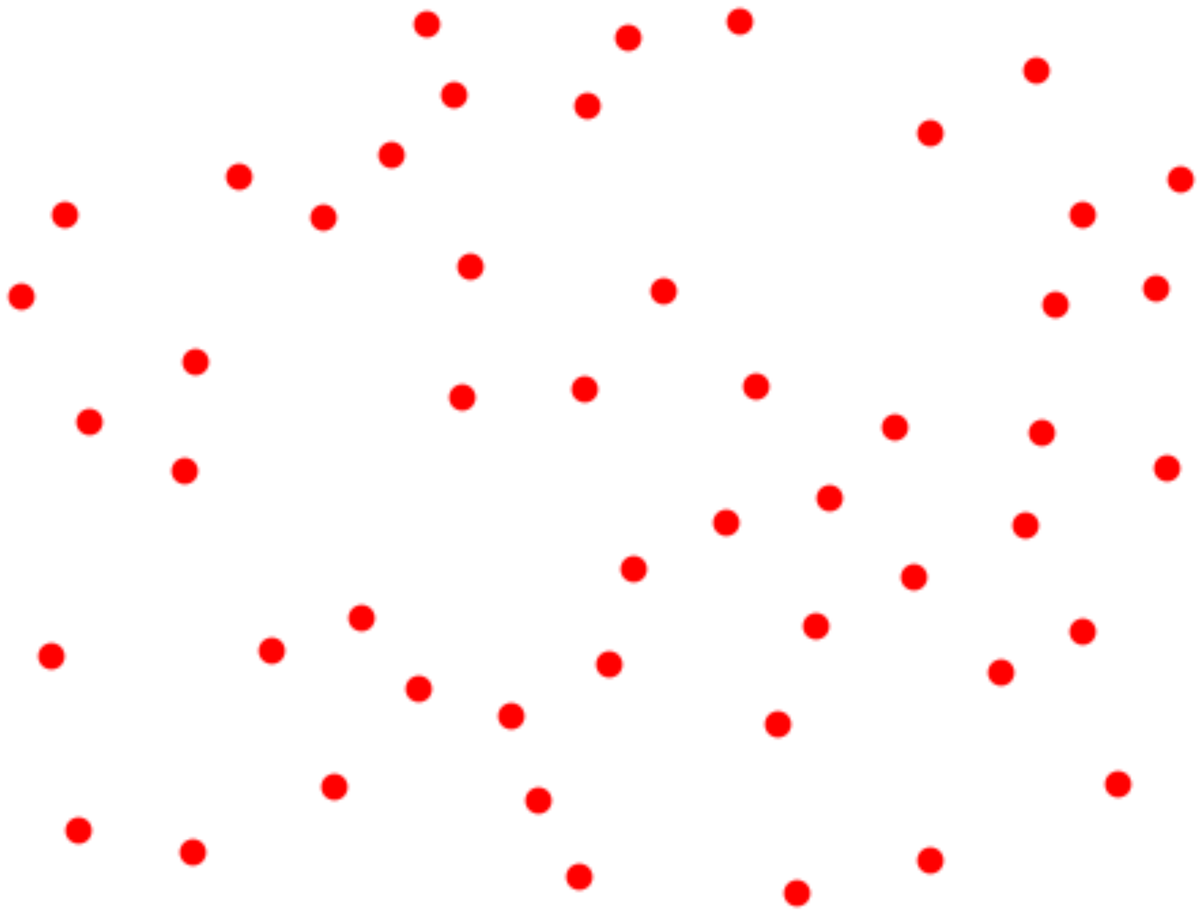
- This is just one model; there are a few different alternative IPs for the TSP.

- Some of these are easier to solve than others.

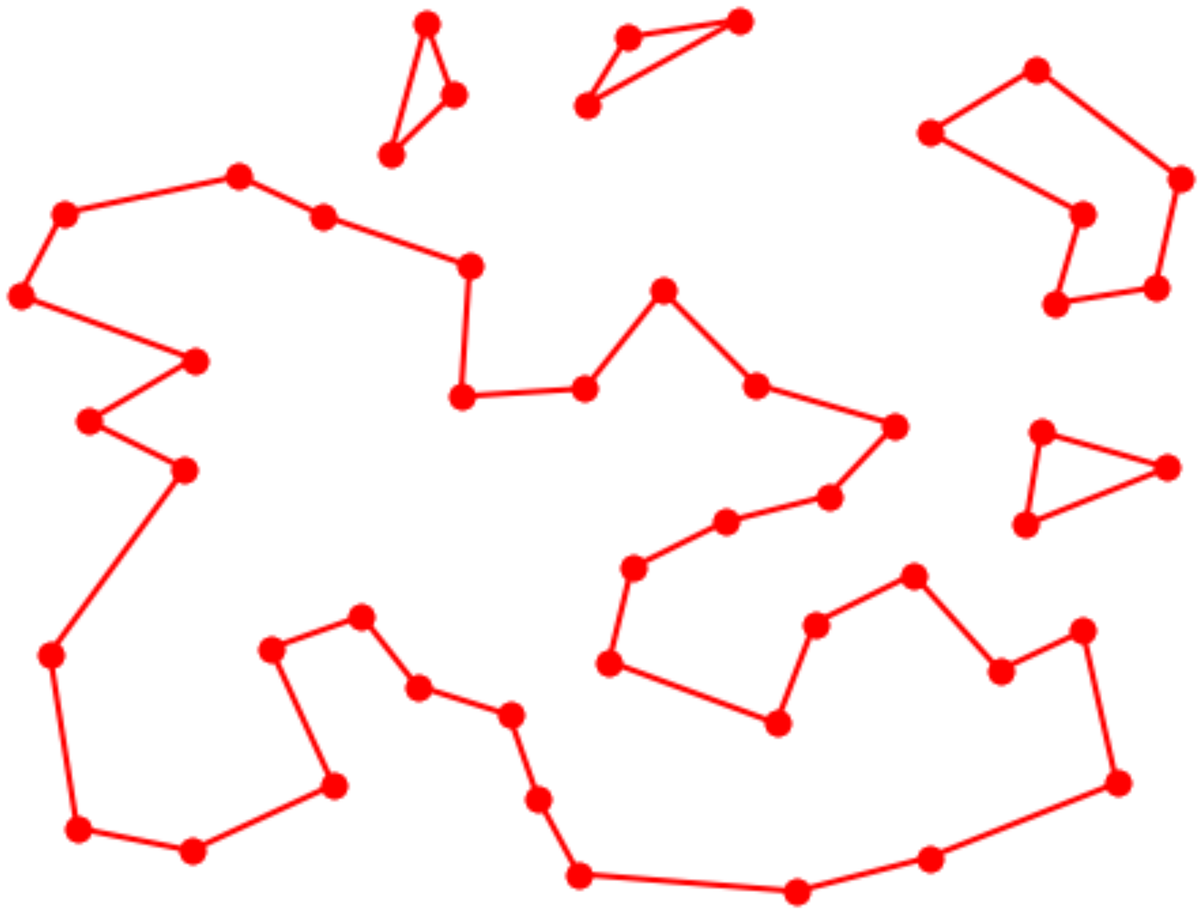
Alternate Solution

- We only add constraints for subtours of length two, i.e. forbidding short loops going forward and backward between two towns.
- When the system comes up with a "non solution" i.e., a number of combined loops, we just add new constraints forbidding these particular short loops and repeat the process until the solution found consists only of a single loop.

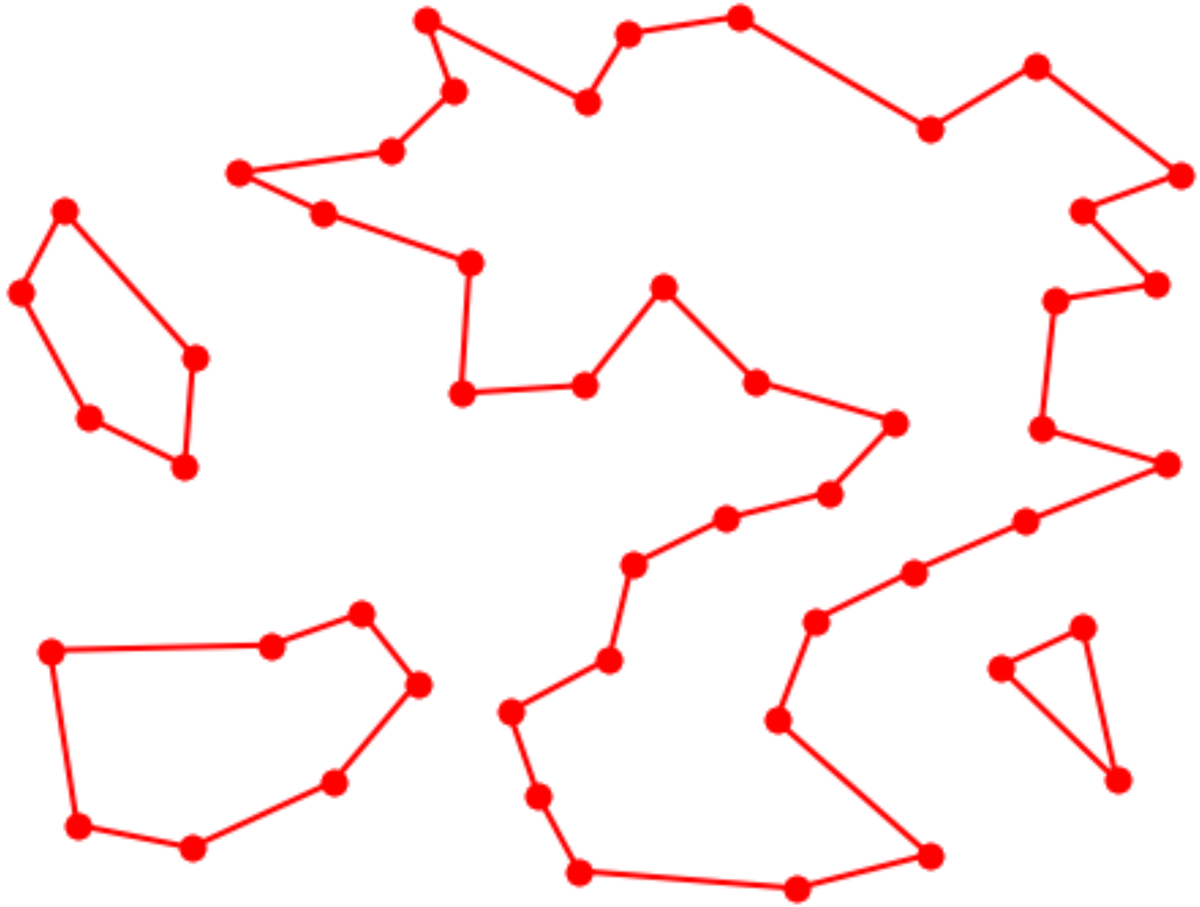
Iterative Solution for n=50



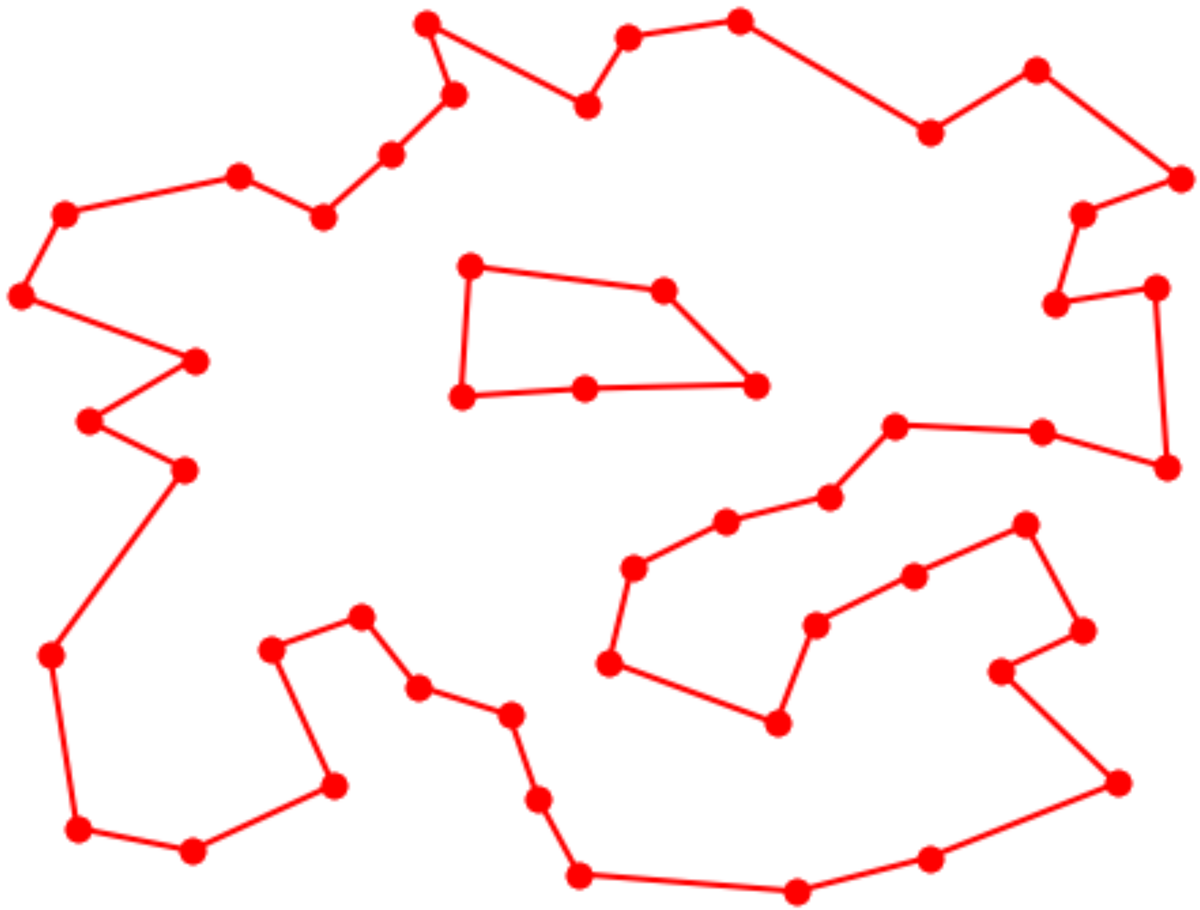
2550 Constraints Length=3785



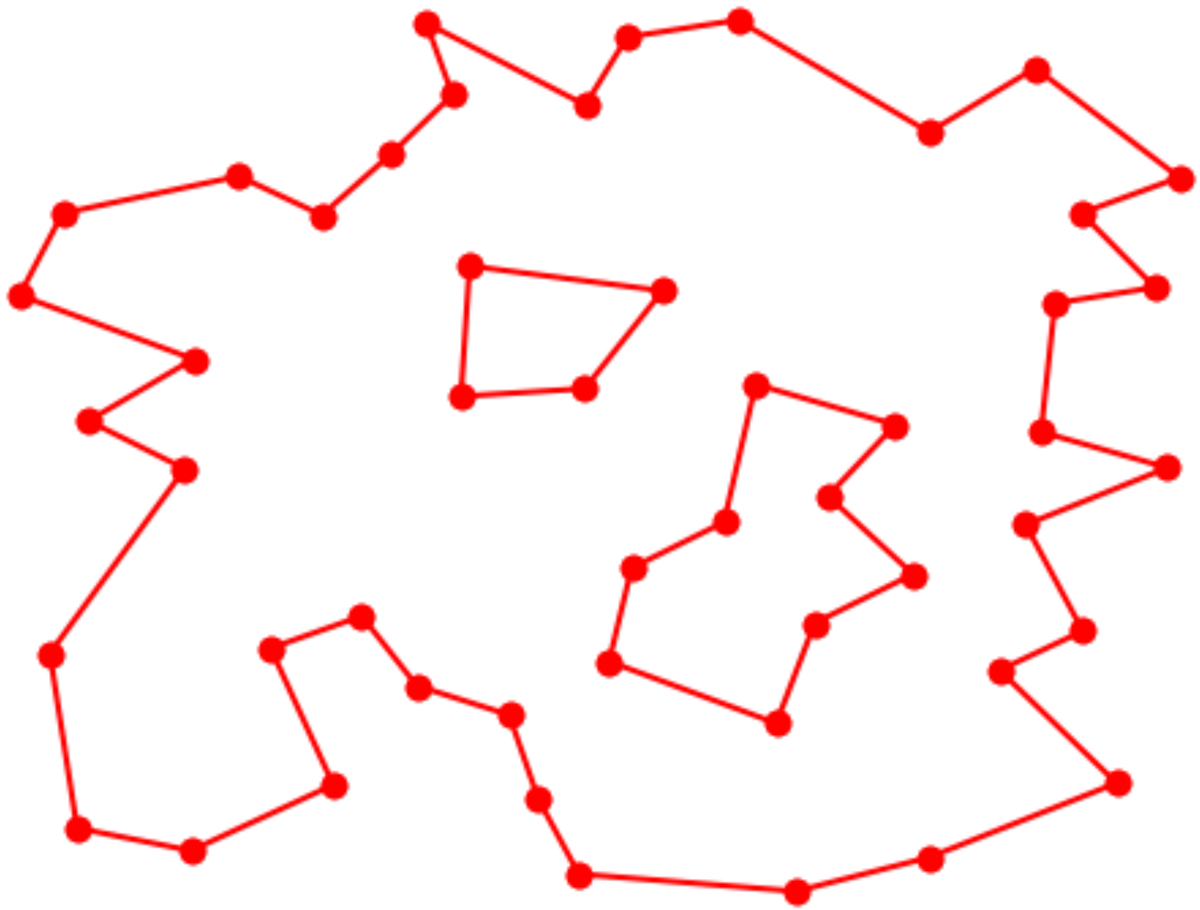
2555 Constraints Length=3796



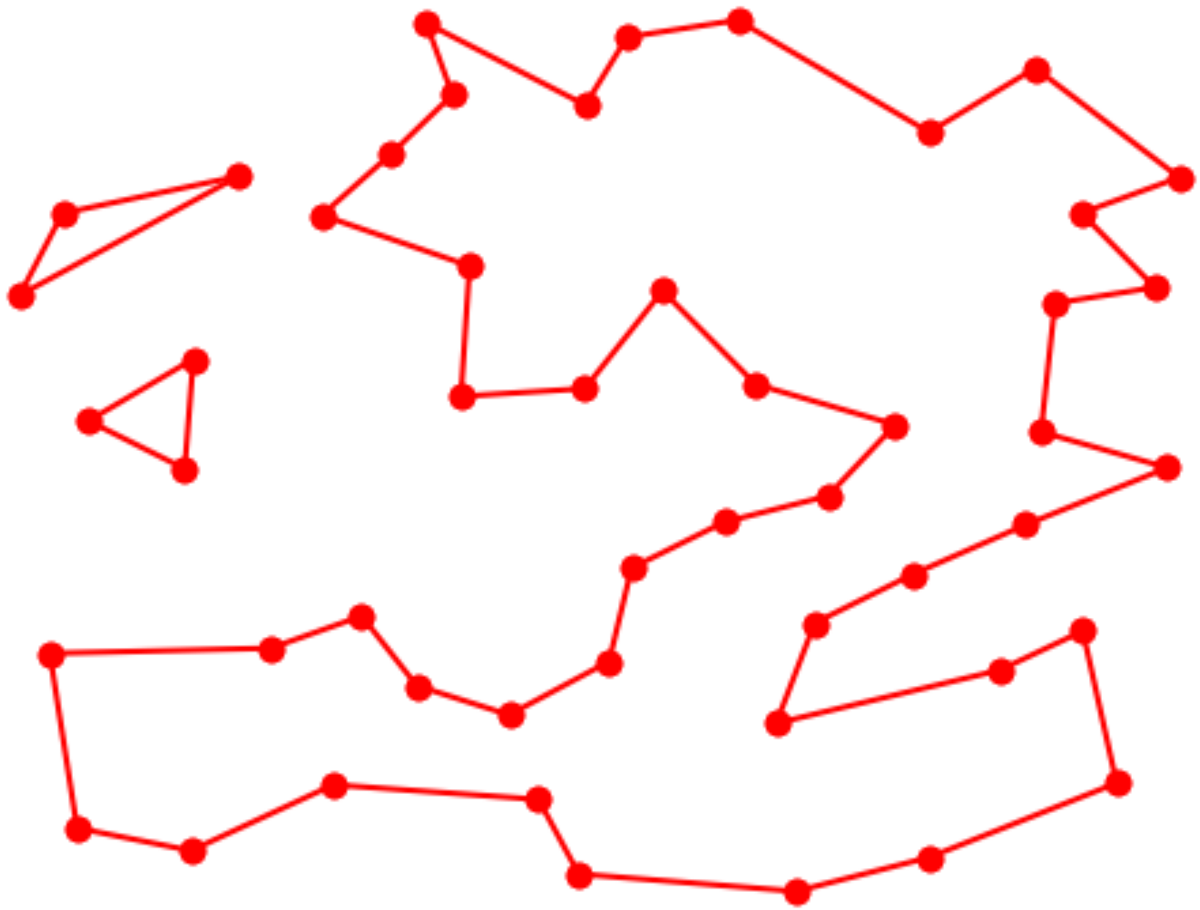
2559 Constraints Length=3817



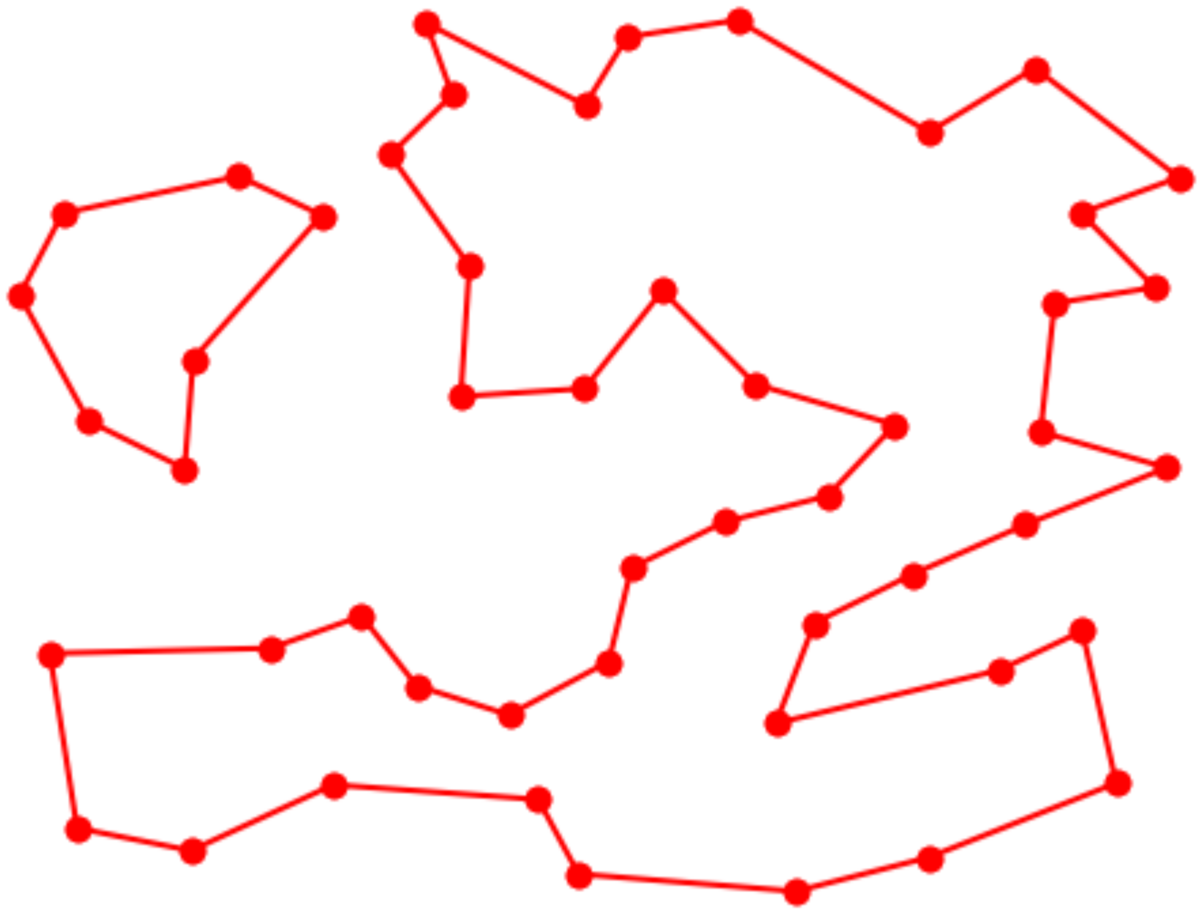
2561 Constraints Length=3820



2564 Constraints Length=3837



2567 Constraints Length=3840



2567 Constraints Length=3845

Total Runtime
15,700ms