solver_greedy_2opt_and.py

This code aims to solve TSP by using the method of greedy and 2 opt.
If the number of data is less than 10, trying all cases will always yield the best results.
If the number of data is more than 10, use greedy, followed by 2 opt, as it takes too much time to calculate the distances for all paths.
gist : https://gist.github.com/hono-mame/42a169bcdedfb296f94b5e61a504caed

Functions

・search_all (cities)
This function is used when the number of data is less than 10.
Enumerate all tours using permutations (=all) and consider only those that start at 1 (= search).
This is because by fixing the starting point, the computational complexity can be reduced to a factor of n while covering all cases.
Calculates the distance of all tours in search and returns the minimum tour.

・twp_opt (tour, dist) →http://www.nct9.ne.jp/m_hiroi/light/pyalgo64.html

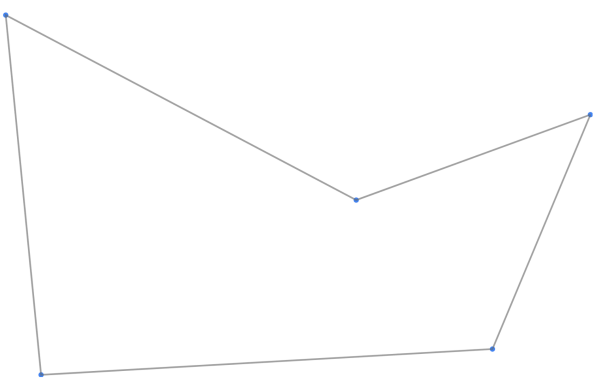
```
   < way1 >                    < way2 >
--→ tour[i]       tour[j] ←---      ---→ tour[i]  ---  tour[j] →------
                ×                  |                                  |
←--- tour[j+1]    tour[i+1] —-      ←---- tour[j+1] --- tour[i+1] ←----
```

If there is a crossing, reconnect, and if the distance is shorter, update.
The computational complexity is about O(N^2) . (N is length of the tour)

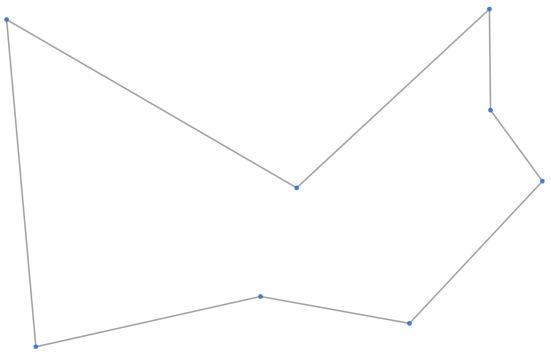
Results

challenge	Challenge 0	Challenge 1	Challenge 2	Challenge 3	Challenge 4	Challenge 5	Challenge 6
data	N = 5	N = 8	N = 16	N = 64	N = 128	N = 512	N = 2048
score	3291.62	3778.72	4494.42	8979.05	11489.79	21,363.60	42,712.37

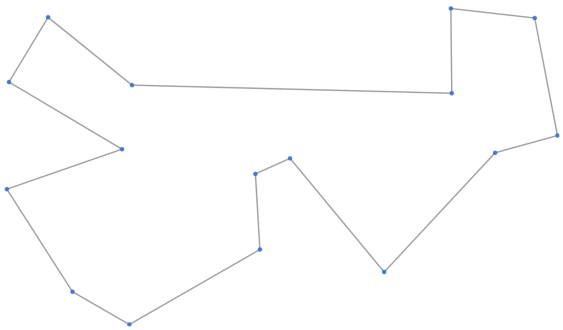
N = 5



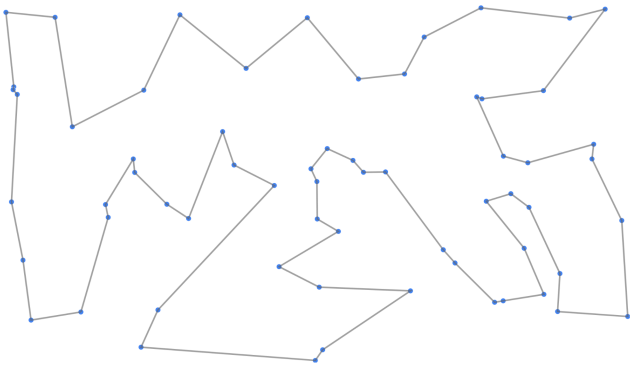
N = 8



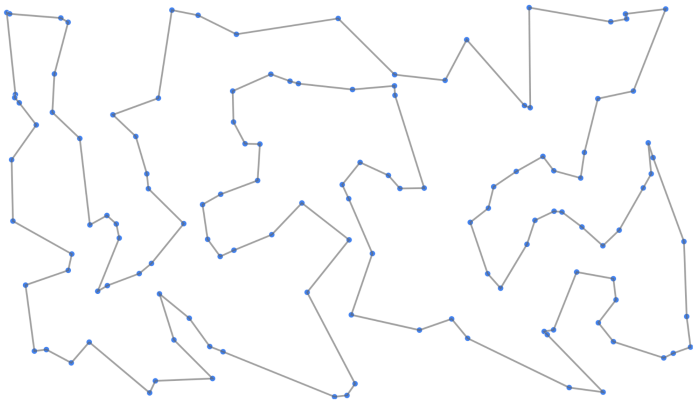
N = 16



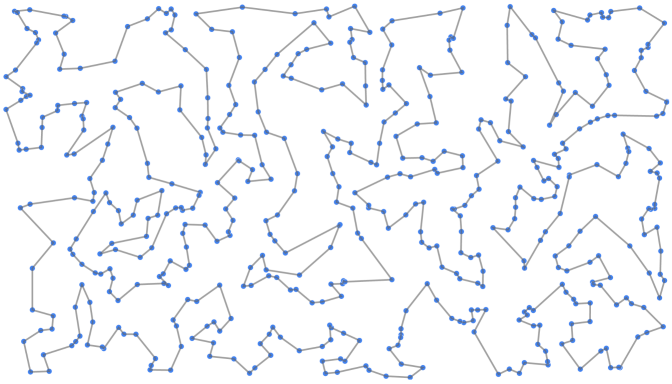
N = 64



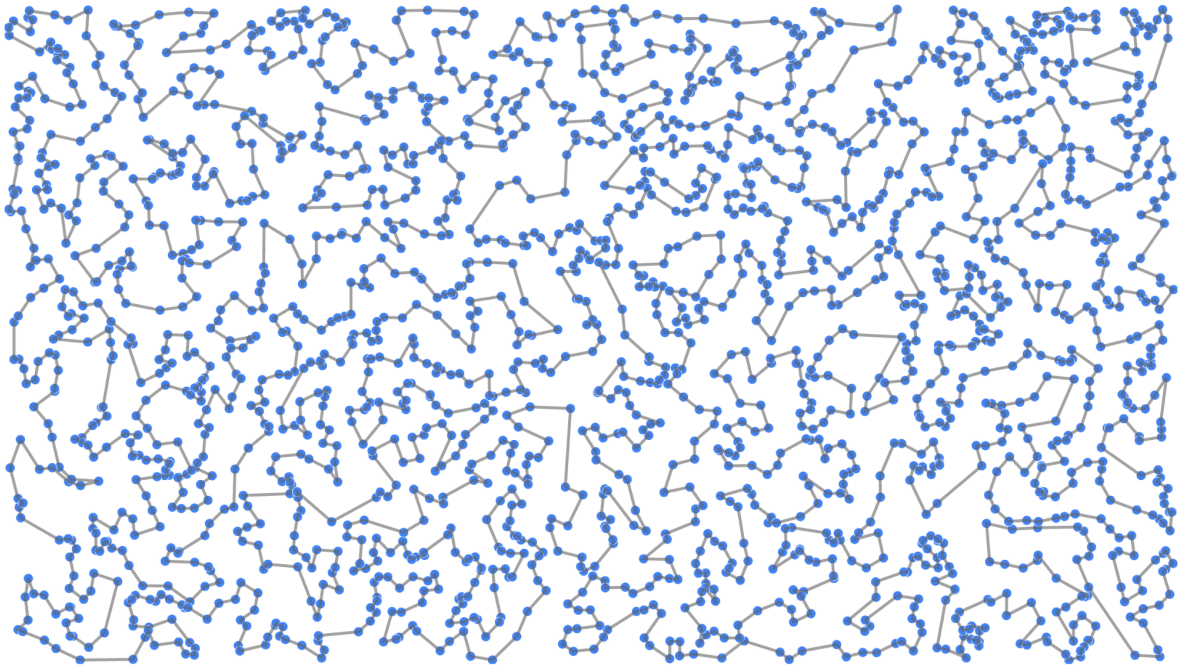
N = 128



N = 512



N = 2048



References

https://github.com/hayatoito/google-step-tsp
https://en.wikipedia.org/wiki/2-opt
http://www.nct9.ne.jp/m_hiroi/light/pyalgo64.html
https://en.wikipedia.org/wiki/3-opt