Olivier AGIER
Horace GUY 31/05/2017

Discrete optimization Task 3 : Travelling Salesman Problem

I worked in collaboration with Olivier on this task. We only used local search algorithms. We implemented Nearest neighbor heuristic and 2-opt search with good success, then 3-opt, and finally simulated annealing (with optional tabu search).

For the first test, multiple random start NN tour improved by 2-opt is sufficient to reach a path length of 429.

For the second test, the following strategy was successful : using 2-opt until a local minimum is reached, then perform 3-opt on the tour. Repeating this for various starting tours leads quite quickly to solution.

For the third test, I had to use simulated annealing to escape the local minima. Sadly, I can't remember which parameters I used to find a good solution : initial temperature, decreasing rate, tabu search or not, when to delete the tabu nodes. Another key factor is how the arc pairs are explored (the « neighborhood »). I found three interesting way of doing this :

- Sort the nodes by arc length, and loop over i and j in the sorted list. Heuristic method, hoping that long edges are likely to be suboptimal, and hopefully by large values. Downsize is the sorting in O(NlogN) at each iteration.
- Sort the nodes by arc length, loop over i and choose a random order for j. This way, large arcs can be flipped with small or average length arcs, not only large ones.
- Randomly choose i and j, and limit the number of nodes explored (or not) to, say 200. This computes fast new choices, and for large samples, avoides the computations of 1 billion value. This way, all nodes are not explored, so the initial condition to stop the search when the length doesn't improve is flawed : it doesn't mean that the algorithm has reached a local minimum.

The fourth sample was quite frustrating: I couldn't manage to break under 37655, even with various parameters for a simulated annealing search. I couldn't obtain very good scores for the fifth problem, but the simulated annealing techniques dramatically improves the 2-opt results very fast.

For the last one, I used the nearest neighbor heuristic, which returns an acceptable result after a few minutes. I then improved the solution by performing 2-opt for hours, without ever reaching a local minimum. I guess a more adequate heuristic could improve substantially the result. Or another way of selecting the arcs visited first in the neighborhood, for example by searching only through the x nearest neighbors of the node i. In order to efficiently find those, a 2-d tree seems like a good lead to explore.
Either way, I didn't manage to find the time to implement it. I tried some homemade algorithm, named « chenille », to simply link the nodes by increasing x then y values (lexicographic (x,y) order), and once a different x is reached change to decreasing y, and so on. It did not work well at all. I think that with a bit more patience and detailed algorithm, one can find a simple good solution, since some arcs are obviously to include in the path by looking at the graph.

I saw that 3-opt wasn't finishing for reasonably large graph sizes, so I didn't try to implement k-opt. Maybe that was a mistake, and some clever passes at the right moment could maybe help to escape local minima and find a very good solution.

Olivier AGIER
Horace GUY 31/05/2017

Generally speaking, this assigment was very challenging because the methods to obtain the solutions are quite blurry. A lot of choices have to be made specifically for some tests, like when to use which search move etc. This is why this was the more exciting assignment, and the most difficult to me. It fits the idea I had of « real-life » optimization task (with a very easily comprehensible problem though).