Travelling Salesperson Problem

Ádám Fischer - 22203265

06 June 2023

Introduction

The Travelling Salesperson Problem (TSP) proposes a seemingly simple question: given any number of cities, we want to travel through them by the shortest route possible, visiting each city exactly once. An equivalent, mathematically more precise formulation of the problem is that we want to find the shortest Hamilton cycle of a graph. How can we find this shortest path?

One might think that the problem can be solved simply by listing all possible routes (combinations), and then we just have to choose the shortest (optimal) path. Although this seems like a logical idea, it is almost impossible to implement this in real life (for large number of cities) since the TSP is a NP-hard problem. It is suspected (but never been proven) that NP $\neq$ P which would imply there is no algorithm that could solve a NP-hard problem, includeing the TSP in polynomial time, which means if we increase the number of cities, the problem becomes practically unsolvable, as it requires too much computational time.

The traveling agent problem has many practical applications, mainly in logistics and planning, but it also appears in apparently remote areas such as DNA sequencing. In my project work I will try to implement optimization methods which are trying to find near optimal solutions. I would like to solve the problem primarily with a simulated annealing algorithm [2], but I'm also planning to use different optimization methods,

such as ant colony optimization [3] and compare their performances.

Simulated Annealing

During numerical optimization, we want to minimize a specific cost function $E(x)$. In the case of TSP, this cost function is the distance traveled. Generally when we impelment numerical optimization algorithms we can use gradient descent based searching techniques to find the optima, but the simulated annealing uses a probabilistic approach to find the solution x_*.

The simulated annealing closely resembles the famous Metropolis–Hasting [4] algorithm. The key idea is that it generates sample states of a thermodynamic system. We start with an initial (guess) state x_0 for the solution, and with an initial temperature T_0. In every iteration we propose a new state for the solution, and if it is a better solution than the previous best guess, we accept it. If it's worse, we accept the propesed state with a given probability. As the systems "cools down" we will accept worse proposals for the solution with smaller and smaller probabilities. This way the algorithm is able to escape from local optima and can find global optima.

Here I will present a pseudo code for the Simulated Annealing based on the notes of Lennon O'Naraigh.

Algorithm 1 Simulated Annealing

Set initial state x_0 and $x = x_0$

Set initial temperature T_0

Set a cooling schedule T_k (somehow), that defines the number of iterations at each temperature

while stopping criterion is not met **do**

  propose new state x_k

  calculate $\Delta E = E(x_k) - E(x)$

  if $\Delta E < 0$ **then**

    accept the new state $x = x_k$

  else

    accept the new state $x = x_k$ with probability $e^{-\Delta E/T_k}$

  end if

end while

Ant Colony Optimization

The ant colony optimization method [1] is inspired by the behavior of real ants. In real life, ants move randomly and leave pheromones as they go. The other ants are more likely to choose a path with more pheromones. The pheromone left behind evaporates over time, so the pheromone density will be lower on paths that take more time for the ants to travel, compared to shorter paths, so longer paths will be less likely to be chosen. Initially, we have to select the number of ants/agents and place each ant on a node. Ant number k at iteration t will traverse from node i to node j with a probability of

$$p_{ij}^{k}(t) = \begin{cases} \frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}}{\Sigma_{k \epsilon allowed}[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}} & \text{if j} \in \text{allowed} \\ 0 & \text{otherwise} \end{cases}$$

where η_{ij} is the visibility of an edge. It is usually inversely proportional to the

distence between the city i and j. τ_{ij} is the pheromone concentration which is updated at the end of every iteration. α and β are hyperparameters to be set. If α is set to be 0 the algorithm will be 'greedy', if β is set to be 0 all the ant will choose a sub optimal trail. A more detailed explanaiton of the workings of the the optimization method can be seen on the flow chart below.

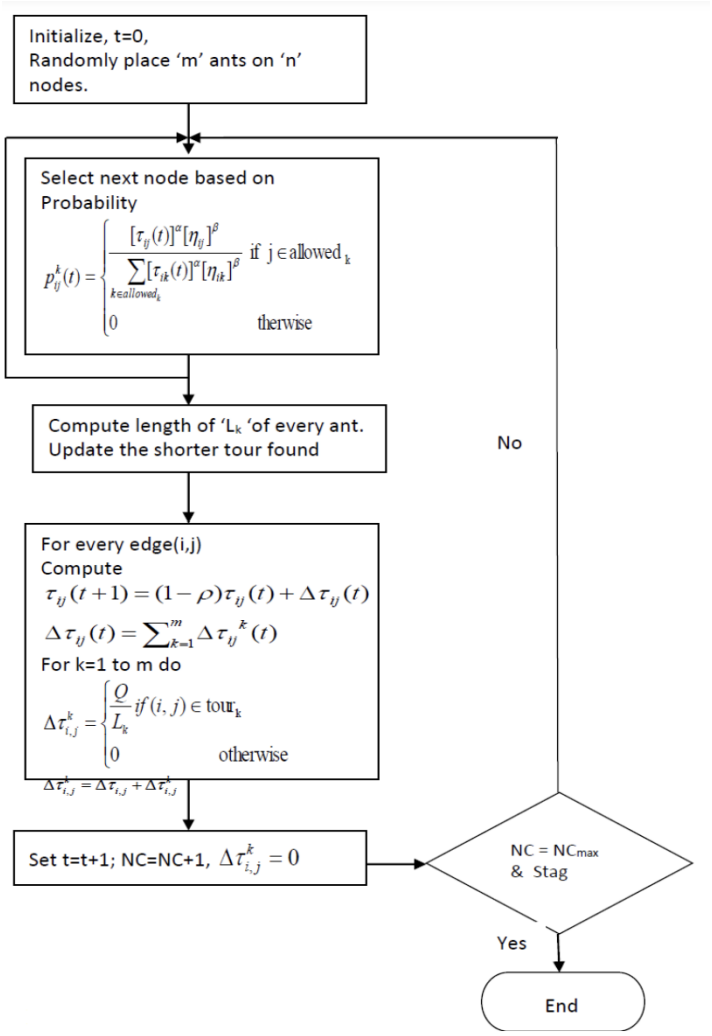

Figure 1: Variables not yet defined: L_k is the travelled distance by ant k, ρ is the evaporation rate and Q is an arbitrary constant.

References

[1] Raghavendra BV. "Solving Traveling Salesmen Problem using Ant Colony Optimization Algorithm". In: *Journal of Applied Computational Mathematics* 04 (Jan. 2015).

[2] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. "Optimization by Simulated Annealing". In: *Science* 220.4598 (1983), pp. 671–680.

[3] Thomas Stützle Marco Dorigo. "Ant Colony Optimization". In: *The MIT Press* (2004).

[4] Nicholas Metropolis et al. "Equation of state calculations by fast computing machines". In: (Mar. 1953).