# In optimization, 2-opt is a simple local search algorithm with a special swapping mechanism that suits well to solve the...

Pedram Ataee, PhD

I am sure you already heard about the traveling salesman problem or TSP. There are many applications for this problem and also many solutions with different performances. Here, I want to share my recent experience to solve the traveling salesman problem especially with the **2-opt method** that is one of the easiest, yet effective, methods to solve this problem.

> If you just want to read about the 2-opt, you can jump directly to the end of this article. You can also use the Python package that I developed below to solve a TSP.

In this article, I want to share my experience in solving a TSP with 120 cities to visit. The problem had to be solved in less than 5 minutes to be used in practice. I aimed to solve this problem with the following methods:

- dynamic programming,
- simulated annealing, and
- 2-opt.

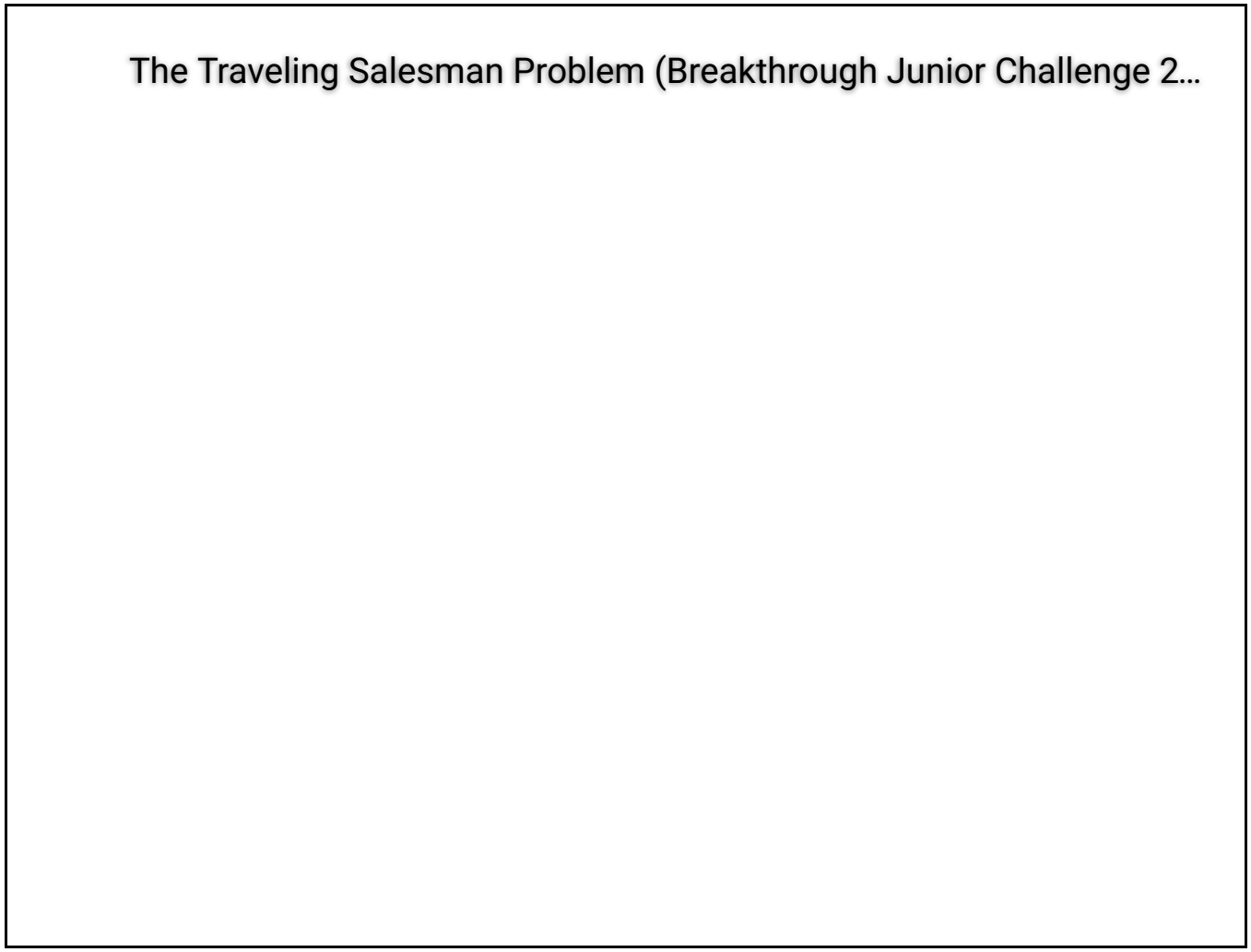
First, let me explain TSP in brief.

## Traveling Salesman Problem

The traveling salesman problem is a classic problem in combinatorial optimization. This problem is to find the **shortest path** that a salesman should take to traverse through a list of cities and return to the origin city. The list of cities and the distance between each pair are provided.

TSP is useful in various **applications** in real life such as planning or logistics. For example, a concert tour manager who wants to schedule a series of performances for the band must determine the shortest path for the tour to ensure reducing traveling costs and not making the band unnecessarily exhausted.

This is an **NP-hard** problem. In simple words, it means you can not guarantee to find the shortest path within a reasonable time limit. This is not unique to TSP though. In real-world optimization problems, you frequently encounter problems for which you must find **sub-optimal** solutions instead of optimal ones.




The traveling salesman problem

# I. Dynamic Programming

The dynamic programming or DP method guarantees to find the best answer to TSP. However, its time complexity would exponentially increase

with the number of cities. The time complexity with the DP method asymptotically equals $N^2 \times 2^N$ where N is the number of cities.

To give you a hint of how this time complexity increases, let me share my experiments. TSP with 10 cities can be solved by a DP method in almost 0.2 seconds using intel core i7. This number increases to almost 13 seconds (~60 times greater) with 15 cities. That is, the time complexity significantly increases even with a small increment in the number of cities.

To optimize the DP method, I could have used the memoization technique. However, the memoization technique with a large number of cities needs a $2^N \times 2^N$ matrix that can not be easily handled in memory.

> **Suggestion-** If you want to solve traveling salesman problem with a large number of cities the dynamic programming method is not the best choice. The DP method can guarantee the global optimum but it just needs much time and computational power that we mostly can not afford in real-world problems.

## II. Simulated Annealing

Simulated Annealing or SA is a heuristic search algorithm that is inspired by the annealing mechanism in the metallurgy industry. Annealing refers to a **controlled cooling mechanism** that leads to the desired state of the material. But, how does this map to an optimization problem?

The lesson that optimization experts learned from the annealing mechanism is to enforce more control over the search process contrary to the gradient descent algorithm with fixed rules. The SA method has two major rules explained below.

- The **moving direction** must be probabilistically determined in each step with the hope of not getting trapped in a local optimum and moving toward the global optimum.
- The **search step** must be reduced in size while the search process is moving forward and getting close to the final result. That helps to move

vigorously in early steps, and cautiously in later steps.

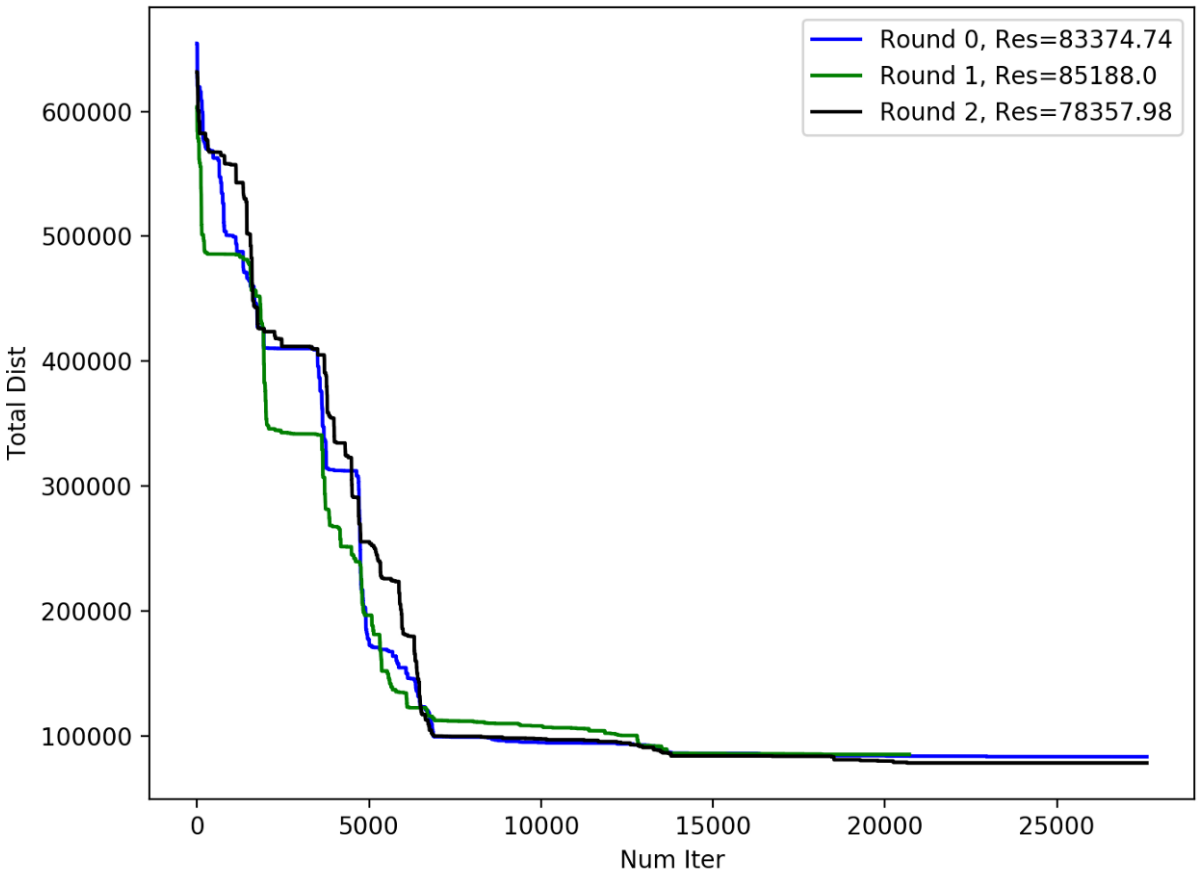
In the SA method, the search process must be continued until finding a good-enough solution or until reaching stopping criteria. Plus, this method is sensitive to how its parameters including the search step and moving direction are formulated and tuned. SA method is a heuristic search algorithm and, therefore, it is sensitive to its **initial point** in the search space.

I implemented the SA method and tested it several times. In the end, I could not obtain a better result compared to the 2-opt method given time constraints to execute. The 2-opt method was executed very fast.

> **Suggestion-** The outcome of the simulated annealing method is sensitive to its parameters and its stopping criteria. A simulated annealing method is a powerful tool. But if you want to work with it, make sure you are aware of its flaws.

## III. 2-opt

The 2-opt algorithm is a simple local search method with a **special swapping mechanism** that works as its heuristic. The main idea behind the 2-opt method is to **remove path crossing** in each neighborhood of cities. The 2-opt can be implemented easily and executed fast. For example, TSP with 120 cities can be solved in less than 5 seconds on the intel core i7 using this method. Here, “solved” means the algorithm converges to a good-enough solution that is a sub-optimal solution. The 2-opt method converges fast since it is deterministic in contrary to the SA method.




3 runs of the 2-opt method with randomized initial points. The stopping criteria are defined based on the ratio of improvement compared to the best result.

This method similar to other heuristic search algorithms does not guarantee finding the global optimum. The 2-opt method can be easily trapped in local optimums since it does not have a mechanism to jump out of them. Knowing all of these flaws, this method still works very well in TSP **since its heuristic is very relevant, so effective, in this problem**.

## pdrm83/py2opt

### In optimization, 2-opt is a simple local search algorithm with a special swapping mechanism that suits well to solve the...

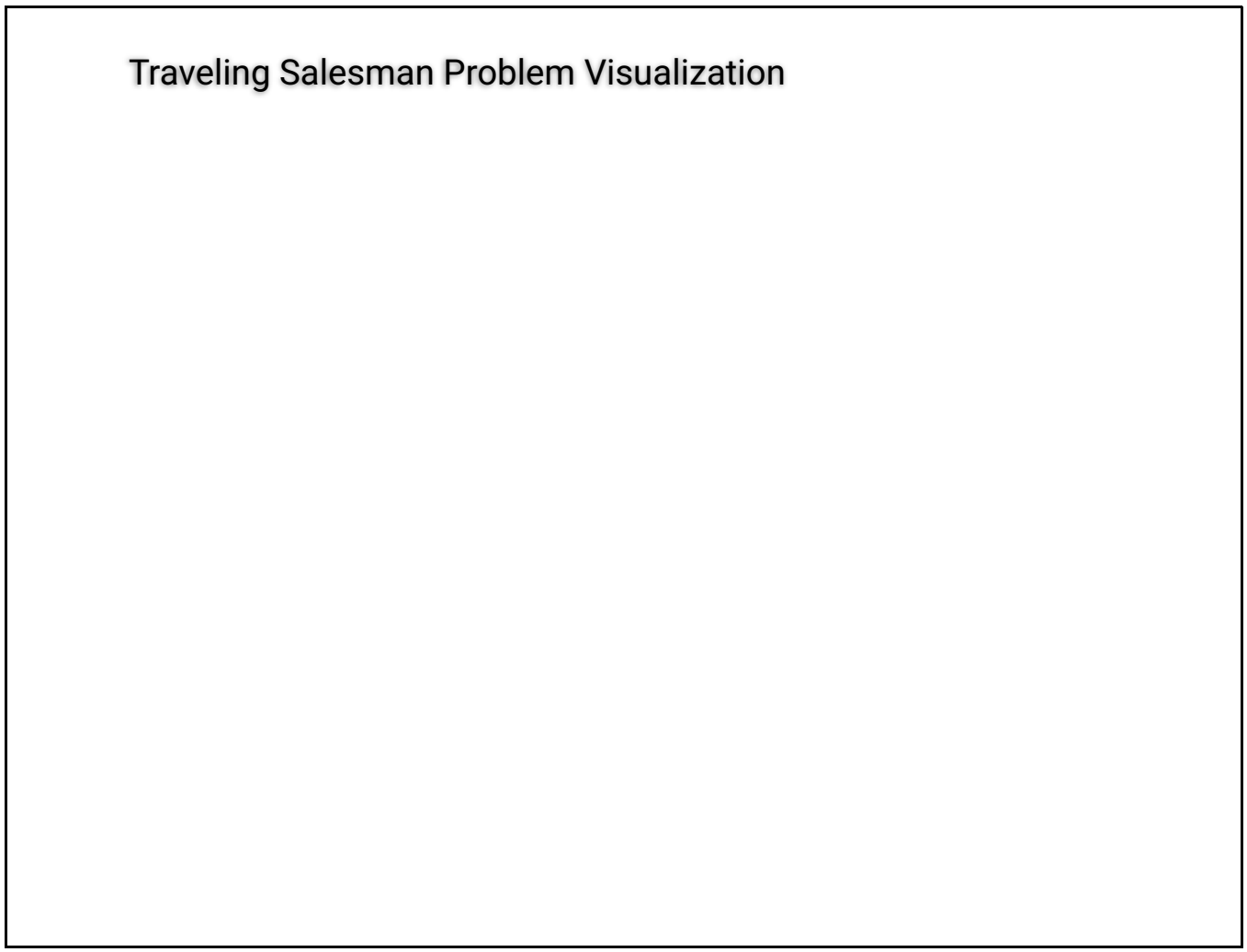
The 2-opt method similar to other heuristic search algorithms is sensitive to its **initial point** in the search space. That means the final outcomes are changed by different initial points. I used a simple trick to work around this

issue and to improve the results of the 2-opt method. I ran the method with different randomized initial points 10 times and selected the best result among them. By this approach, I reduced the sensitivity to the starting point to some extent.

> **Suggestion-** The 2-opt method can be implemented easily and executed fast. Plus, it works much better than the expectation, especially when you decrease its sensitivity to the initial point in the search space. I highly suggest using this method to solve the TSP unless a good-enough result is not appropriate for you.

# IV. Summary

The video below is a good summary of this article. You will enjoy watching it.




# Takeaway

There are many methods to solve TSP. However, the above methods are the most common ones. I highly suggest the 2-opt method since it is implemented simply and executes fast. However, the simulated annealing method is very powerful if you can properly tune it and you do not have a time constraint to find the final result.

> The last words- When you want to find a solution for any problem including TSP, always think about how a simple technique such as the 2-opt method can work well. Why? Because its heuristic is very well-suited to the problem.

## Thanks for Reading!

If you like this post and want to support me...

- *Follow me on Medium!*
- *Check out my books on Amazon!*
- *Become a member on Medium!*
- *Connect on Linkedin!*
- *Follow me on Twitter!*