Simulated Annealing

Blaine Mason Λ and Justin Ventura Λ

Traveling Salesman Problem

- Given a list of cities and the distances between each city, find the shortest route where one can visit every city exactly once and return to the starting city.
- Brute Force
  - A brute force solution would be to measure all permutations of the list and find the shortest cost.
  - The downfall of this method is that it would take $O(n!)$ to complete. :(
- Why is this problem so popular?
  - The TSP can be used as an effective way to test how efficient an optimization method is.
- How can we solve the TSP in a timely manner?
  - Random selection and hope that we get lucky.
  - Dynamic Programming.
  - Greedy Algorithm.
  - **Metaheuristics!**

Metaheuristics

- Procedure designed to select a partial solution that may result in a sufficiently good solution to an optimization problem.
- Most methods can find solutions to optimization problems such that the solution is a small distance from the optimal solution.
- The metaheuristic we chose was Simulated Annealing.

Simulated Annealing

- The algorithm comes from the method of annealing in metallurgy.
- A controlled cooling process.
  - Begins at a high temperature and accepts initial solutions, but as temperature decreases solutions aside from the current best are less likely to be accepted.
  - If δ or the difference between current solution and the proposed solution is positive, we will accept the current solution regardless of the temperature.
  - The acceptance of a solution also relies on the following inequality:

$$e^{\frac{\delta}{T_{curr}}} > U(0,1)$$

  - Finally, if the current solution is smaller that the best solution, the current solution is the best solution.
- We saw a similar process in the ising model

Serial Simulated Annealing

Algorithm 1 Simulated Annealing Pseudocode

1: **procedure** SIMANNEAL($m, iter_{max}, T$)
2: $x_{curr} \leftarrow InitialConfig(m)$ ▷ Select some $x \in S$
3: $x_{best} \leftarrow x_{curr}$
4: **for** $i = 1$ **to** $iter_{max}$ **do**
5: $x_{prop} \leftarrow NeighbourConfig(x_{curr})$ ▷ Propose some neighbour configuration
6: $temp_{curr} \leftarrow CalcTemp(i, T)$ ▷ Anneal system
7: **if** $Cost(x_{prop}) \leq Cost(x_{current})$ **then**
8: $x_{curr} \leftarrow x_{prop}$
9: **if** $Cost(x_{prop}) \leq Cost(x_{best})$ **then**
10: $x_{best} \leftarrow x_{prop}$
11: **end if**
12: **else if** $\exp\left\{\frac{Cost(x_{curr}) - Cost(x_{prop})}{temp_{curr}}\right\} > Random(0, 1)$ **then** ▷ Accept upward step?
13: $x_{curr} \leftarrow x_{prop}$
14: **end if**
15: **end for**
16: **return** x_{best} , $Cost(x_{best})$ ▷ Best configuration and associated cost
17: **end procedure**

Serial Simulated Annealing

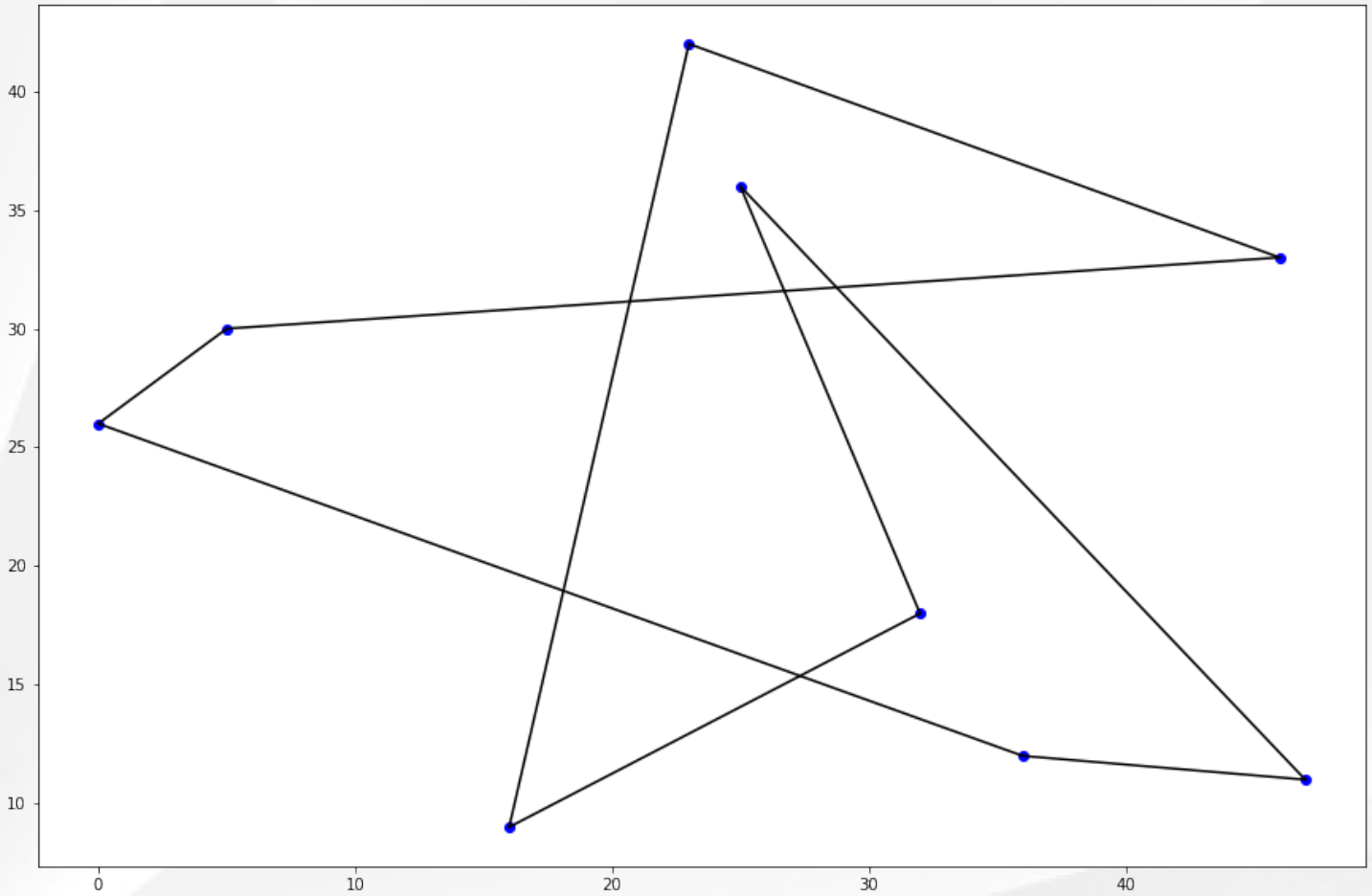

Figure: 1 iteration of SA

Serial Simulated Annealing

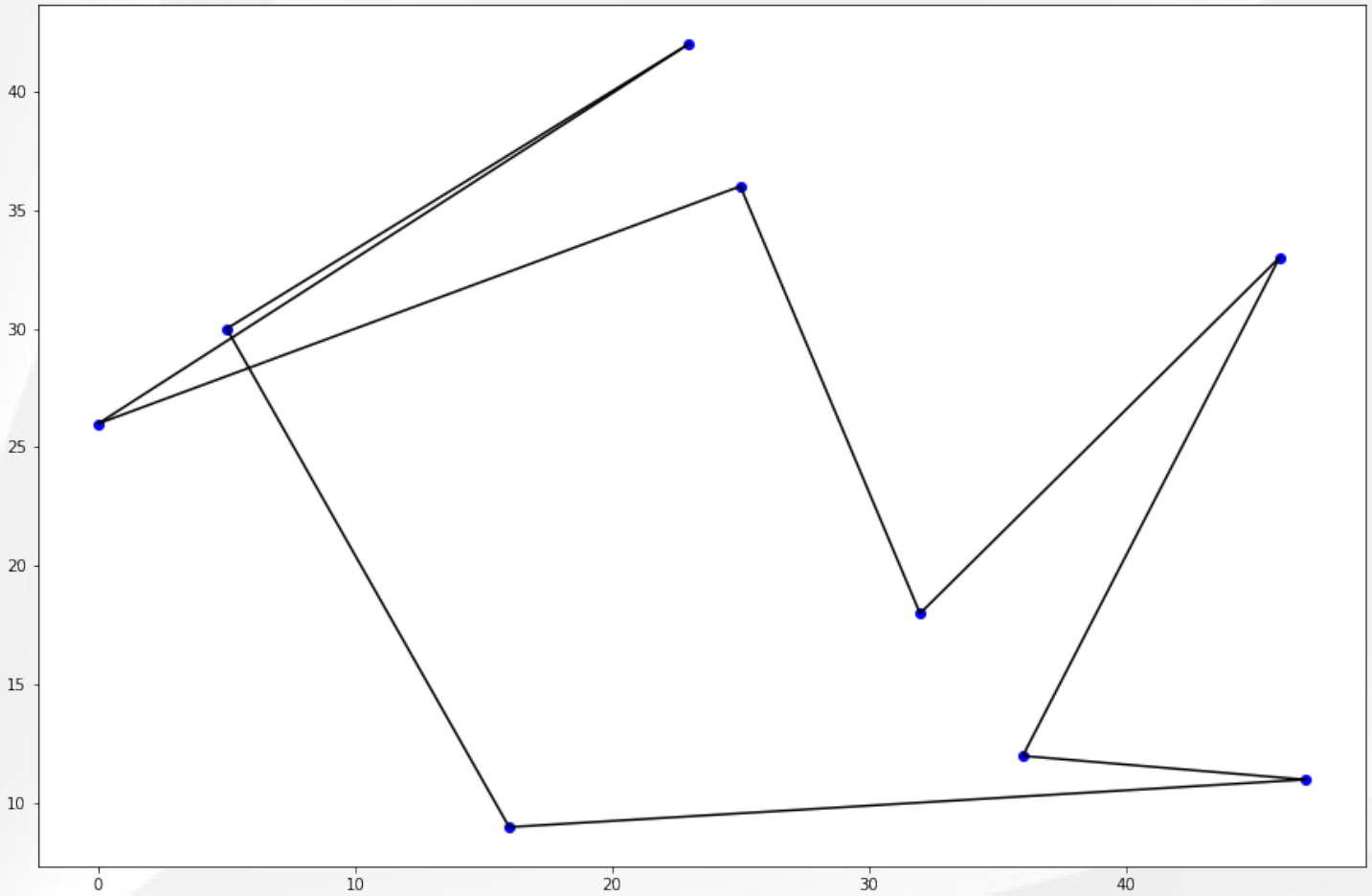

Figure: 10 iterations of SA

Serial Simulated Annealing

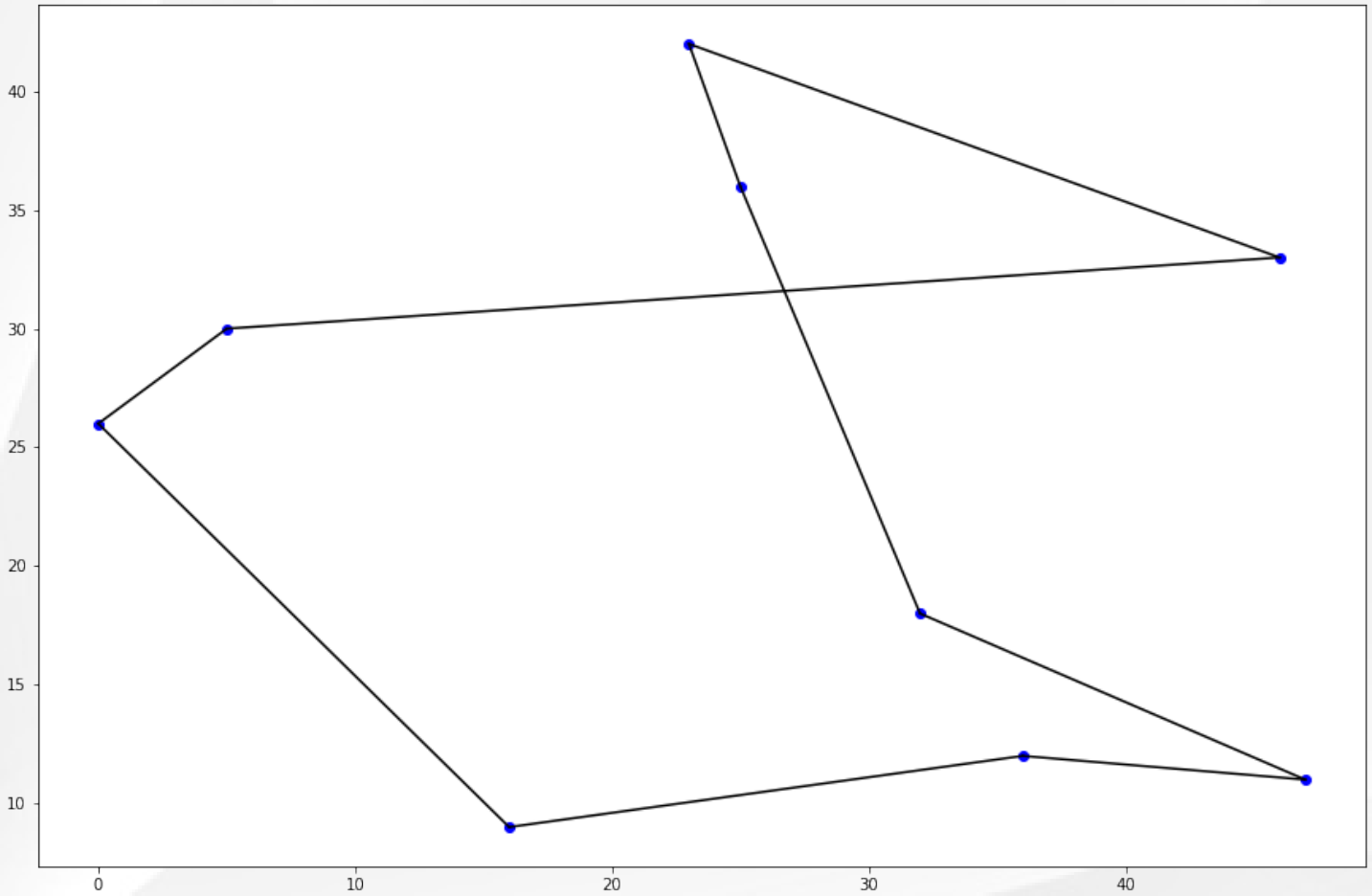

Figure: 100 iterations of SA

Serial Simulated Annealing

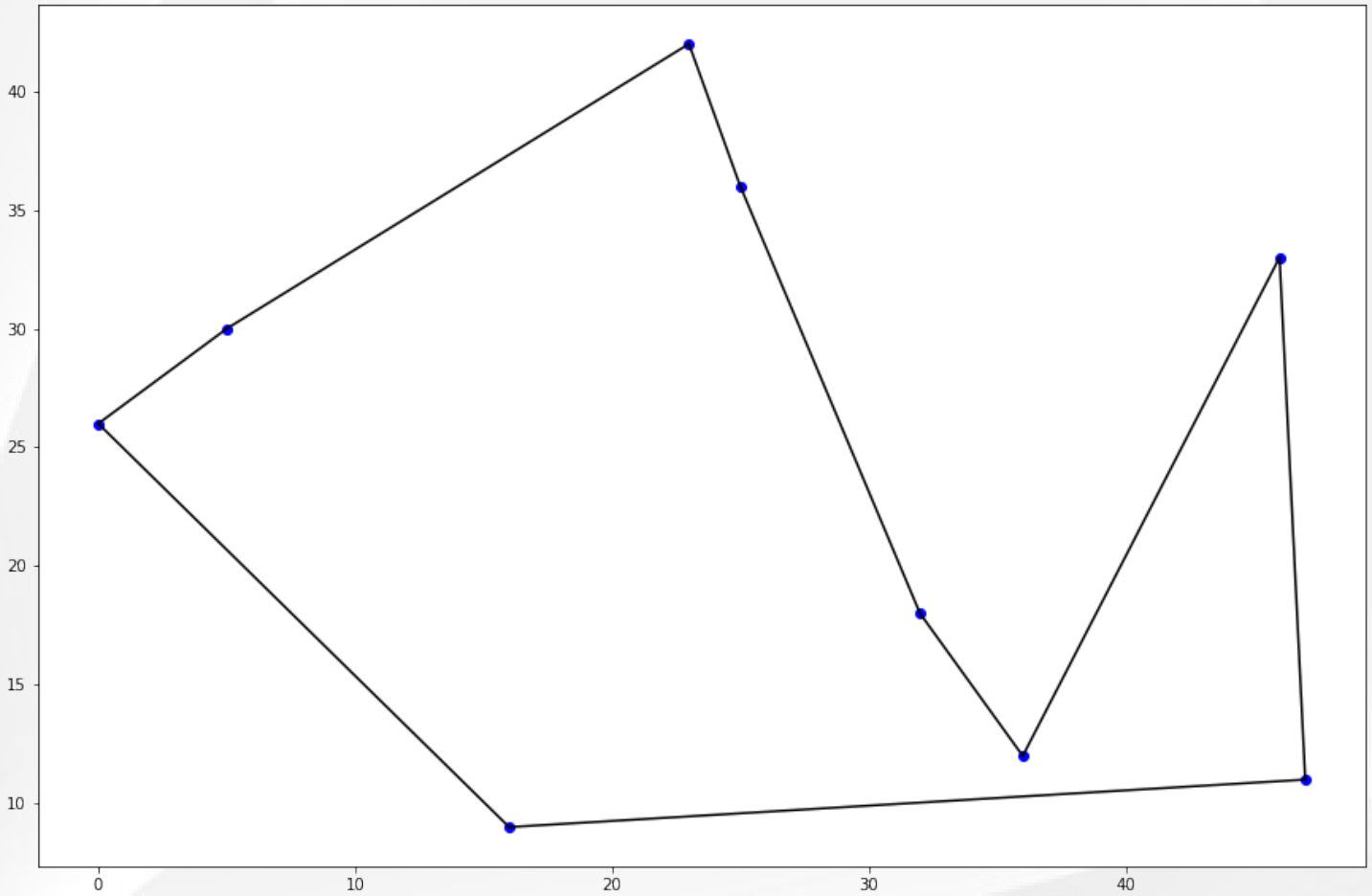

Figure: 1000 iterations of SA

Serial Simulated Annealing

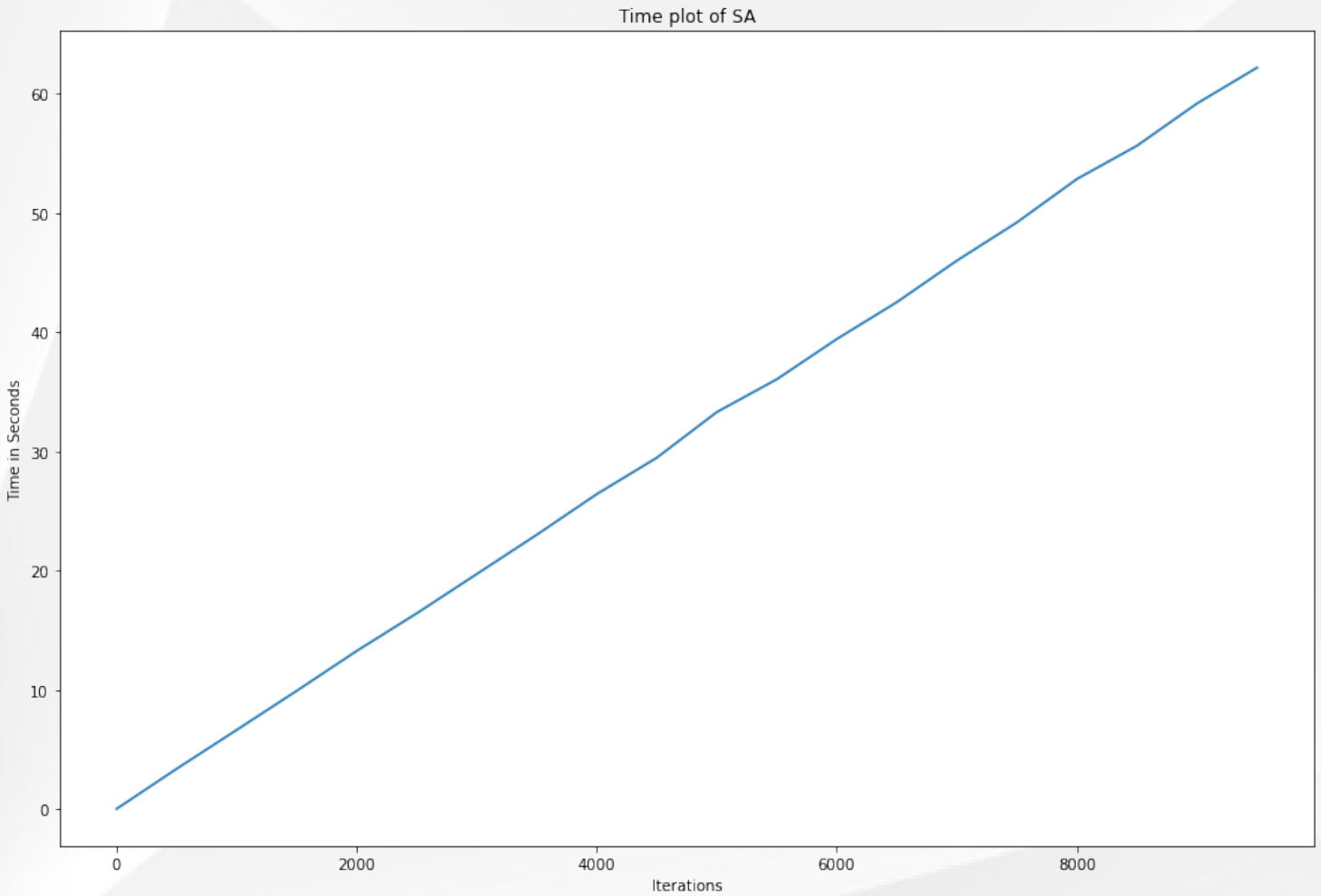

Figure: Time for SA on a nine node TSP

Parallel Simulated Annealing

- The idea of SA in parallel has been implemented by many, but we chose to use the implementation done by Czech and Skinderowicz
- Parallel SA works as a team effort

Parallel Simulated Annealing

```
1   parfor P_j, j = 0, 1, ..., p − 1 do
2     Set co-operation mode to regular or rare depending on a test set;
3     L := (5E)/p; {establish the length of a cooling stage}
4     Create the initial_solution using some heuristics;
5     current_solution_j := initial_solution; best_solution_j := initial_solution;
6     for f := 1 to 2 do {execute phase 1 and 2}
        {beginning of phase f}
7       T := T_{0,f}; {initial temperature of annealing}
8       repeat {a cooling stage}
9         for i := 1 to L do
10          annealing_step_f(current_solution_j, best_solution_j);
11        end for;
12        if (f = 1) or (co-operation mode is regular) then {ω = L}
13          co_operation;
14        else {rare co-operation: ω = E}
15          Call co_operation procedure every E annealing step
              counting from the beginning of the phase;
16        end if;
17        T := β_f T; {temperature reduction}
18      until a_f cooling stages are executed;
        {end of phase f}
19    end for;
20  end parfor;
```

Figure: Main Algorithm

Parallel Simulated Annealing

```
1    procedure annealing_step_f(current_solution, best_solution);
2      Create new_solution as a neighbor to current_solution
         (the way this step is executed depends on f);
3      δ := cost_f(new_solution) − cost_f(current_solution);
4      Generate random x uniformly in the range (0, 1);
5      if (δ < 0) or (x < e^{−δ/T}) then
6        current_solution := new_solution;
7        if cost_f(new_solution) < cost_f(best_solution) then
8          best_solution := new_solution;
9        end if;
10     end if;
11   end annealing_step_f;
```

Figure: Step of Annealing

```
1    procedure co_operation;
2      if j = 0 then Send best_solution_0 to process P_1;
3      else {j > 0}
4        receive best_solution_{j-1} from process P_{j-1};
5        if cost_f(best_solution_{j-1}) < cost_f(best_solution_j) then
6          best_solution_j := best_solution_{j-1};
7          current_solution_j := best_solution_{j-1};
8        end if;
9        if j < p - 1 then Send best_solution_j to process P_{j+1}; end if;
10     end if;
11   end co_operation;
```

Figure: Main Algorithm

Parallel Simulated Annealing

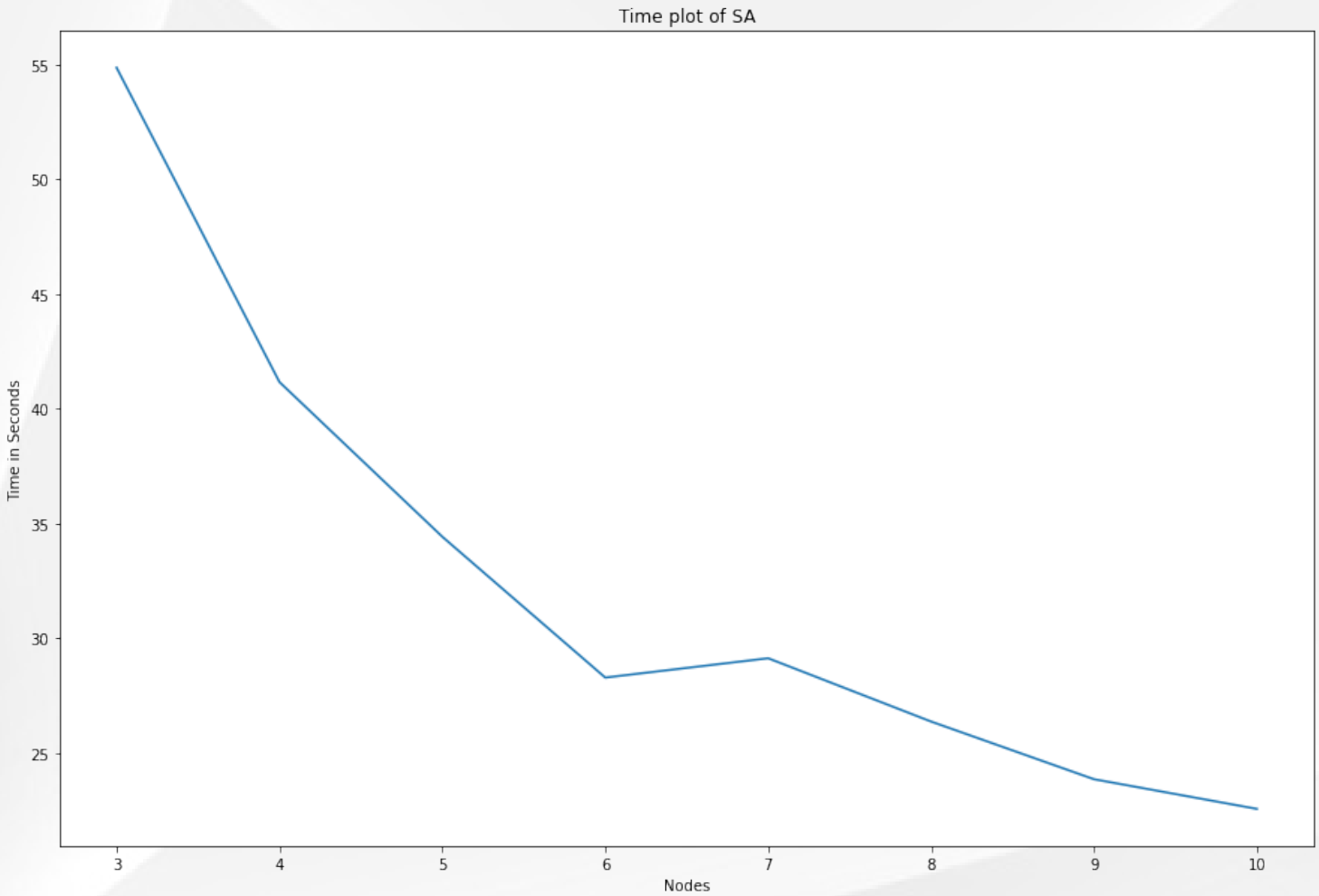

Figure: Parallel Speedup(16)

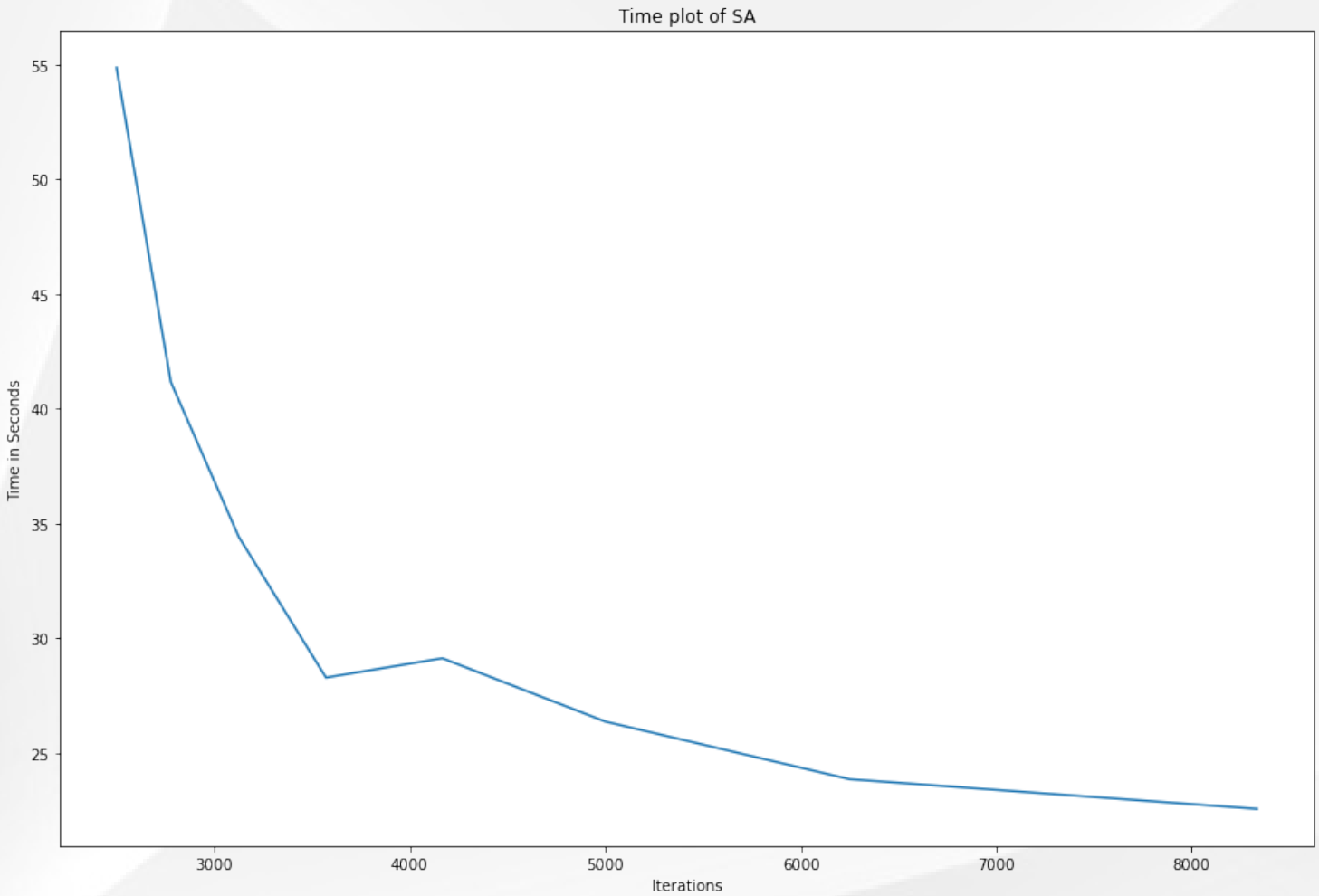

Figure: Iteration Speed Up(16)

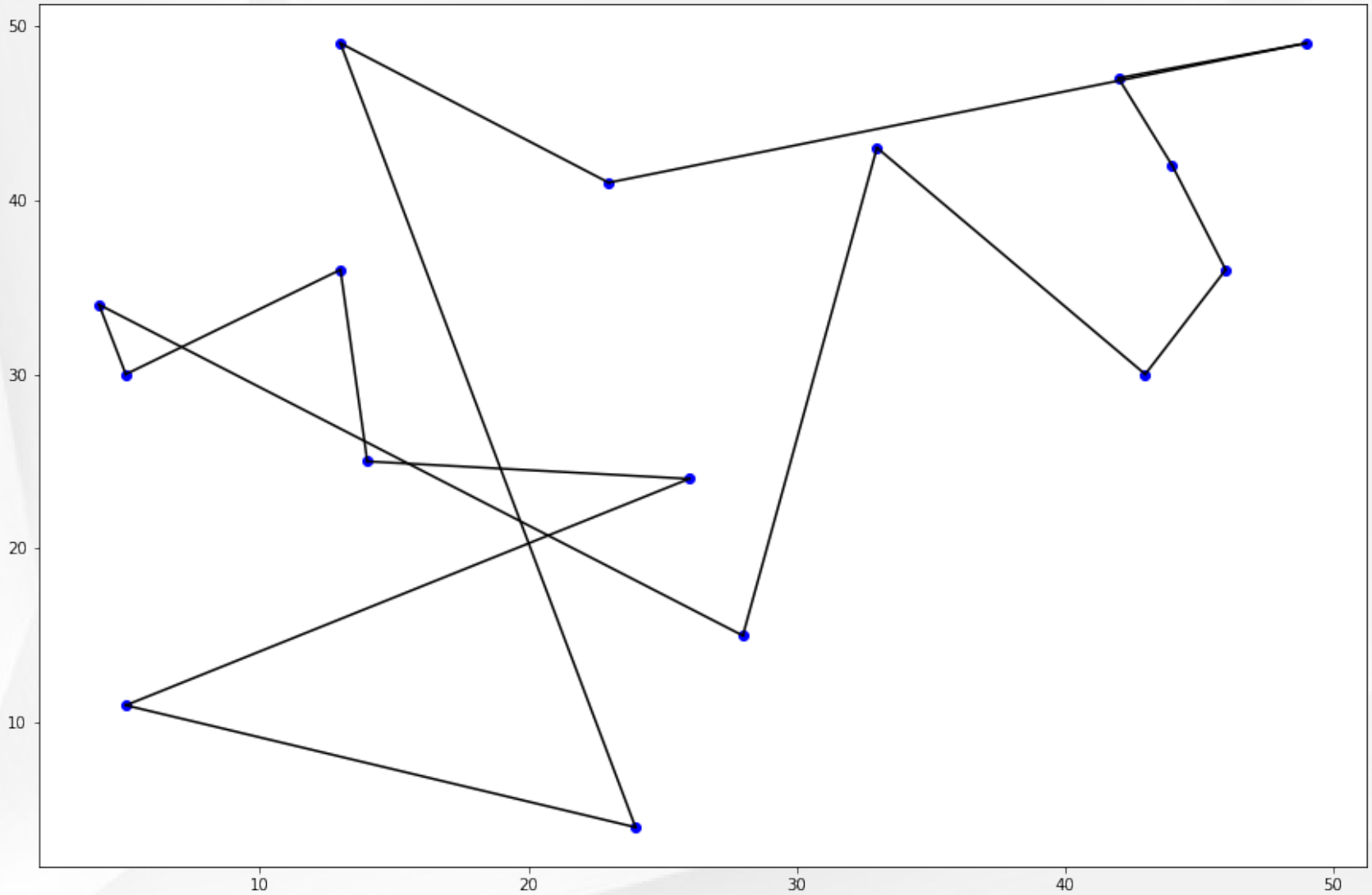

Figure: 16 Node TSP Serial Solution(236)

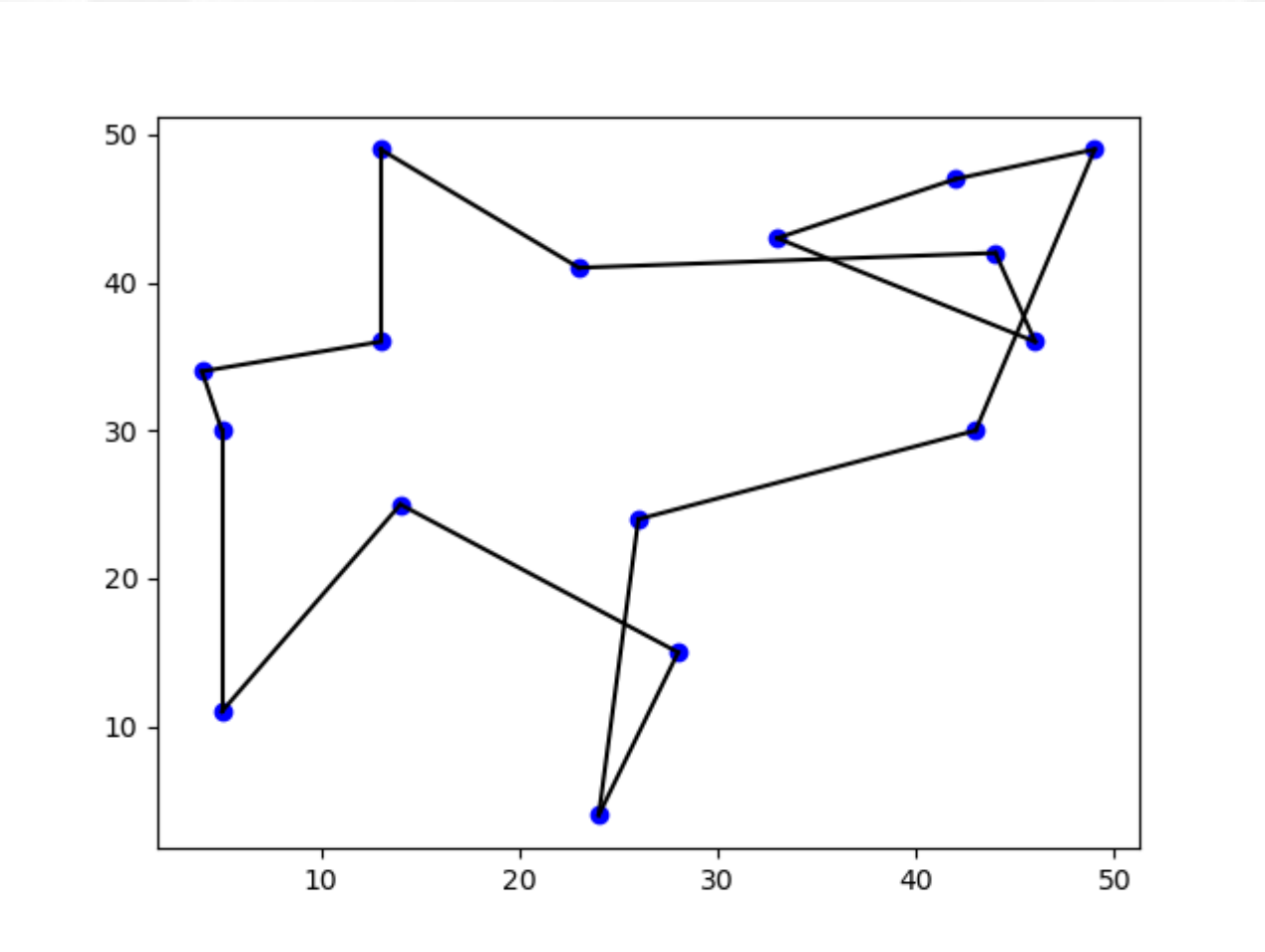

Figure: 16 Node TSP Parallel Solution(236)

Parallel Simulated Annealing

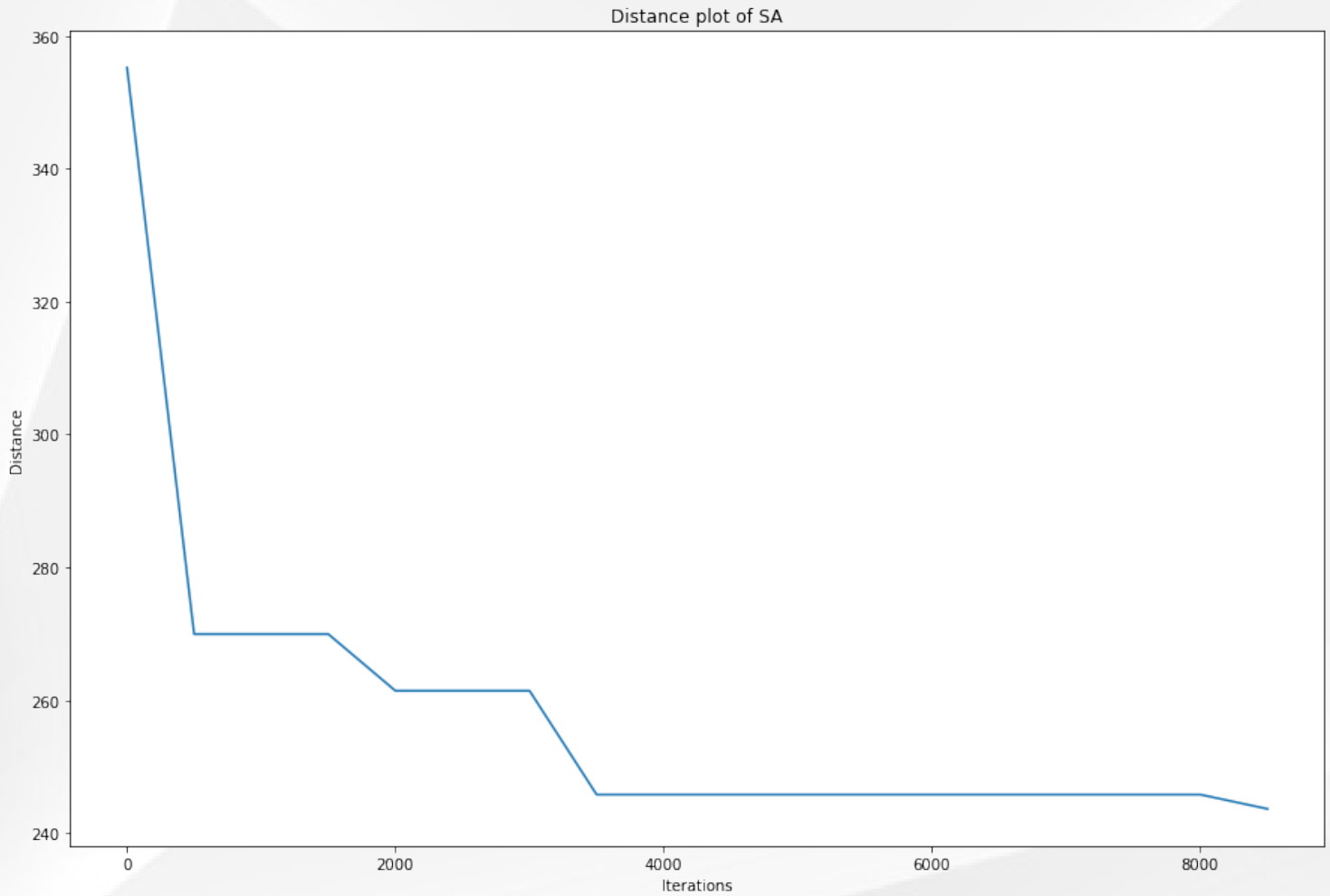

Figure: Distances on 16 node TSP

Conclusions

- We do not fully understand the choice of L, which determines the iterations.
- This algorithm could see improvements in selection of a neighboring solution.
- Python > C
- Stonks only go up.