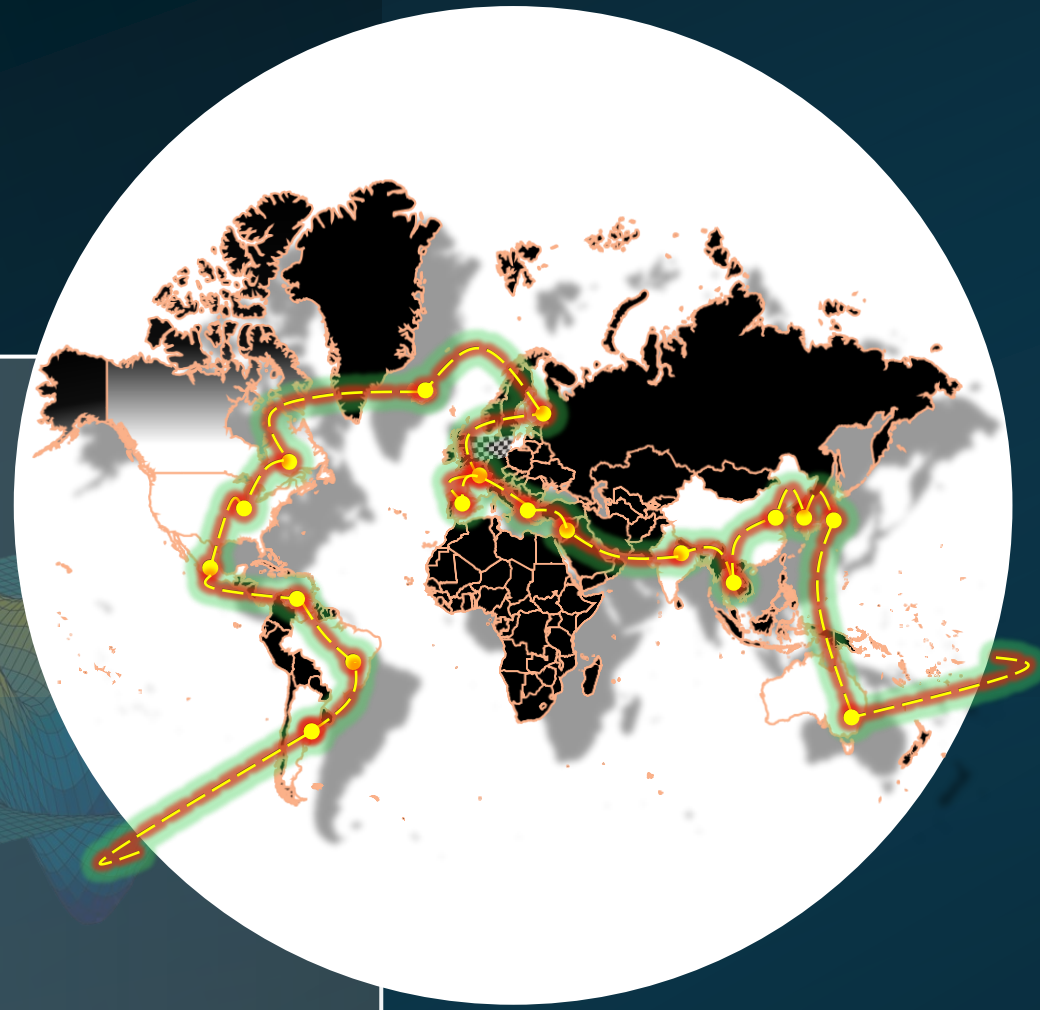
# The Salesman Problem

" Using the simulated annealing method "


- ***A project realized by :***

  *Adan Castillo– Victor Leroy –Asma Soufi*

**1.Introduction :**

1. Traveling Salesman Problem
2. Monte Carlo Optimization
3. Temperature in STMH method

**2. Methodology:**

1. Simplification of the problem
2. The greedy algorithm.
3. Cooling schedules

**3.Analysis :**

1. Time comparison greedy vs annealing
2. Convergence : speed and final result
3. Application : speedrunning in Stones of Barenziah

**4.Application to important problems**


Faculté de physique et ingénierie
Université de Strasbourg


INTRODUCTION

# {"The traveling salesman problem"}

Question

Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once ?"

We have to look for the Minimum of the function = distance

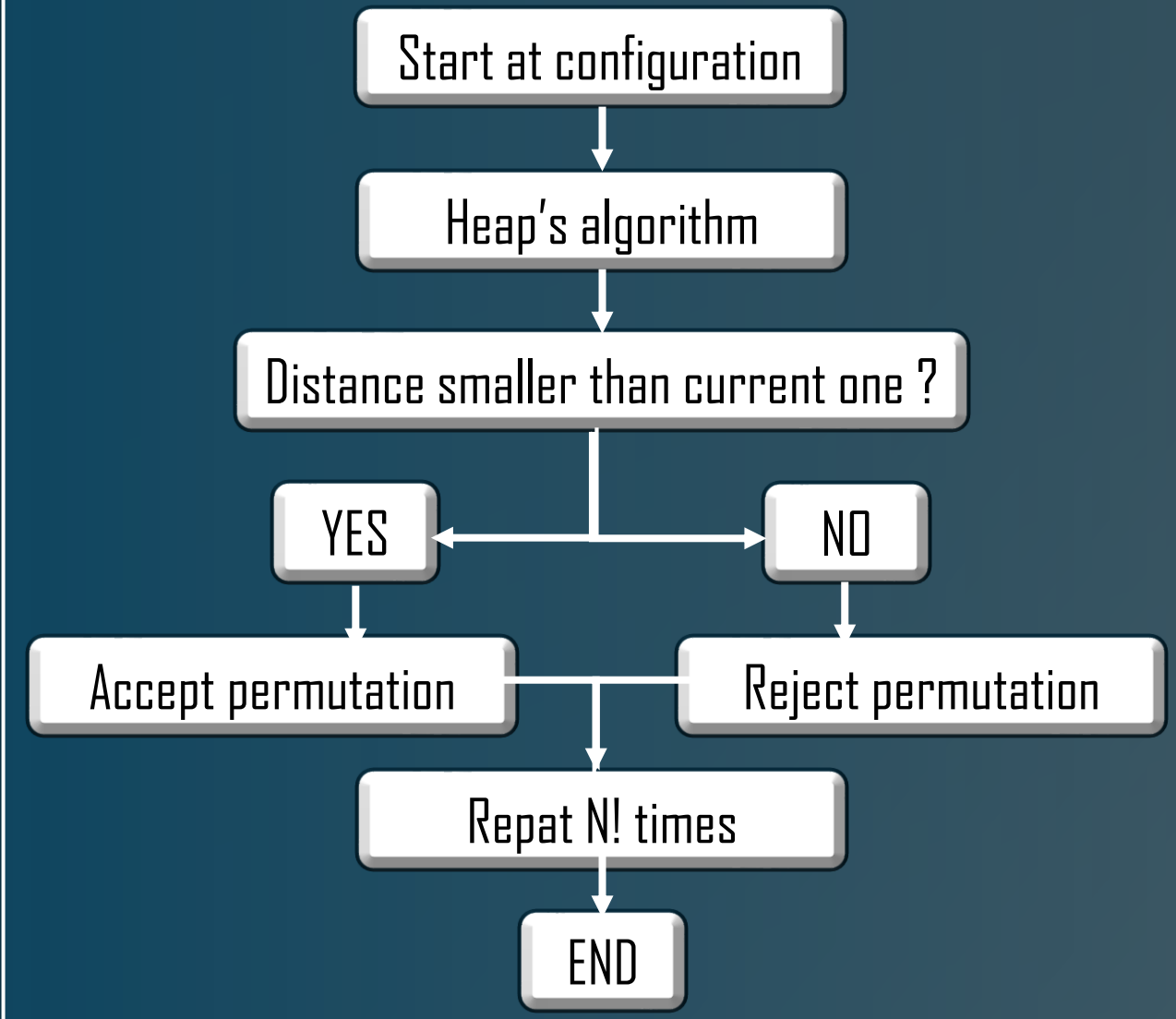
The number of operation required to solve exactly a system of N element is equal to N!

For 7 cities ⇒ 5040 combinations
For 14 cities ⇒ 87 Billion combinations

Use of Monte Carlo method for approximate solutions

**Figure 1:** itinerary for 'N' cities in Germany

# {" The greedy algorithm"}




Before delving into the Monte Carlo method, we take a look at the so called 'greedy' algorithm. The greedy algorithm explores all possible permutations one by one, saving at each step the shortest route.

**This ensures that we get the global minimum.**

To explore all possible permutations, we used Heap's algorithm (B. R. Heap, 1963). This is a combinatorics algorithm that generates all possible permutations.

# "Monte Carlo Methods : Metropolis-Hasting algorithm"

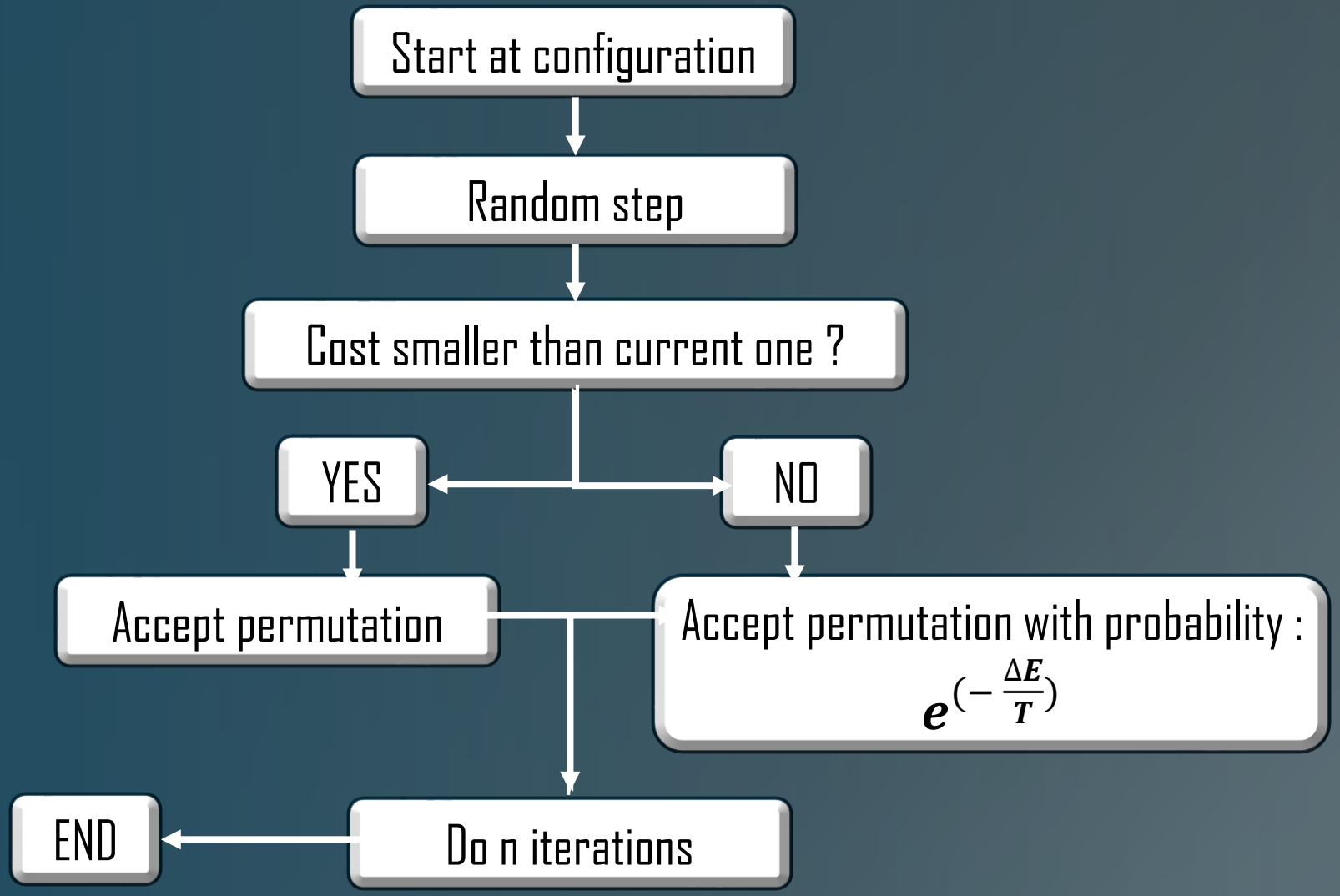
Widely used in statistical mechanics, the Metropolis-Hasting algorithm is a Markov-chain Monte Carlo method that can explore high-dimensional space to find optimum point. (W.K. Hastings, 1970)

2 steps :

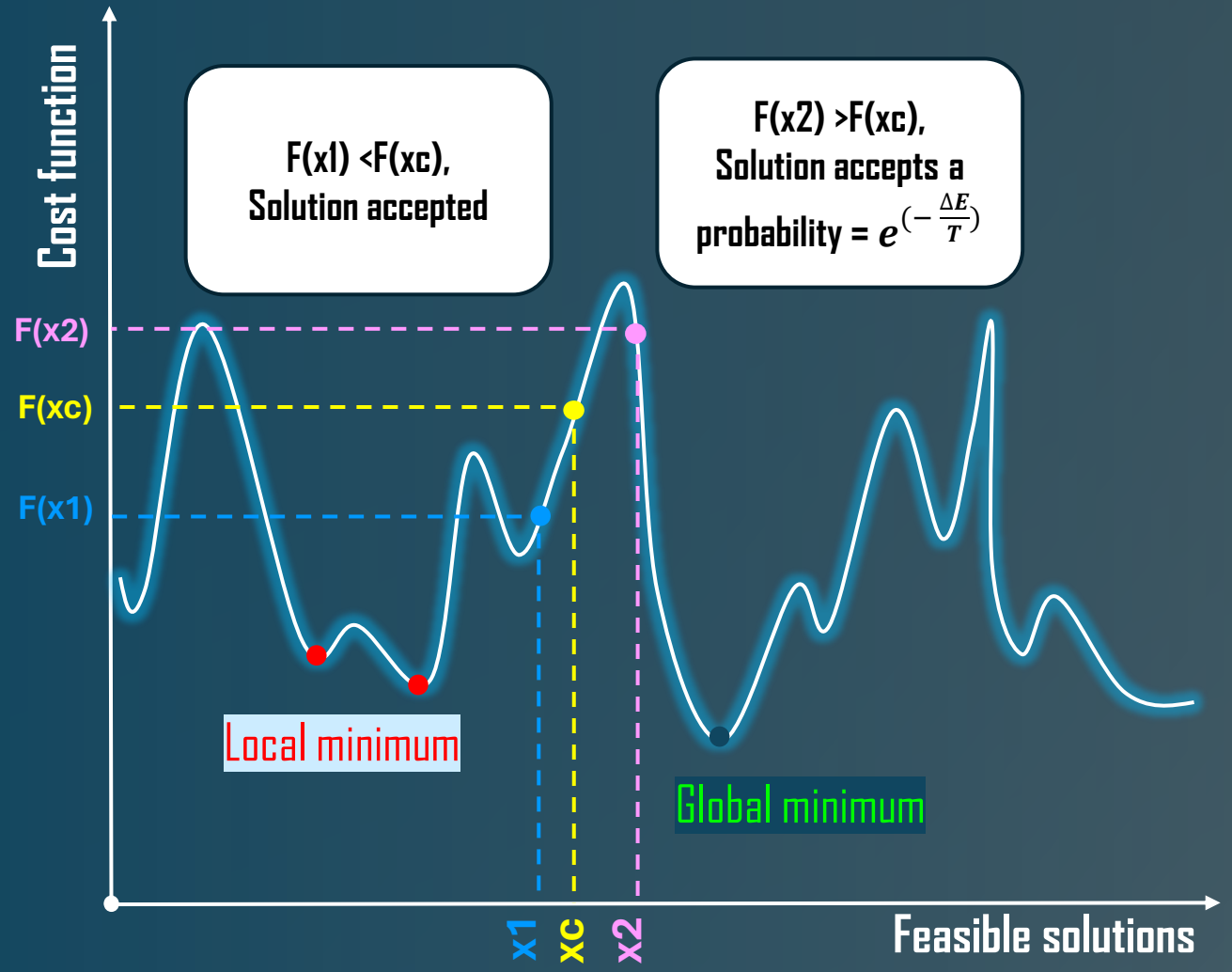
1. Perform a random step within the space and calculate the "cost" of that point .

2. Compare the cost (energy) with the previous one, if the cost is smaller, we accept the point, if not we take it following Boltzmann probability distribution: $e^{(-\frac{\Delta E}{T})}$.

# "Monte Carlo Methods : Metropolis-Hasting algorithm"



# "Monte Carlo Methods : Metropolis-Hasting algorithm"




Configurations with higher energies (Cost) are less likely to be explored due to low probability.

**xc:** Current solution
**x1:** first neighborhood
**x2**: Second neighborhood
$\Delta E = F(xi)-F(xc)$

# { "The role of Temperature" }

Temperature ⇒ Step size

Observation on Temperature behavior

High Temperature ⇒ Global search
Low Temperature ⇒ Local search

New strategy

Start with Global search and then fine tune

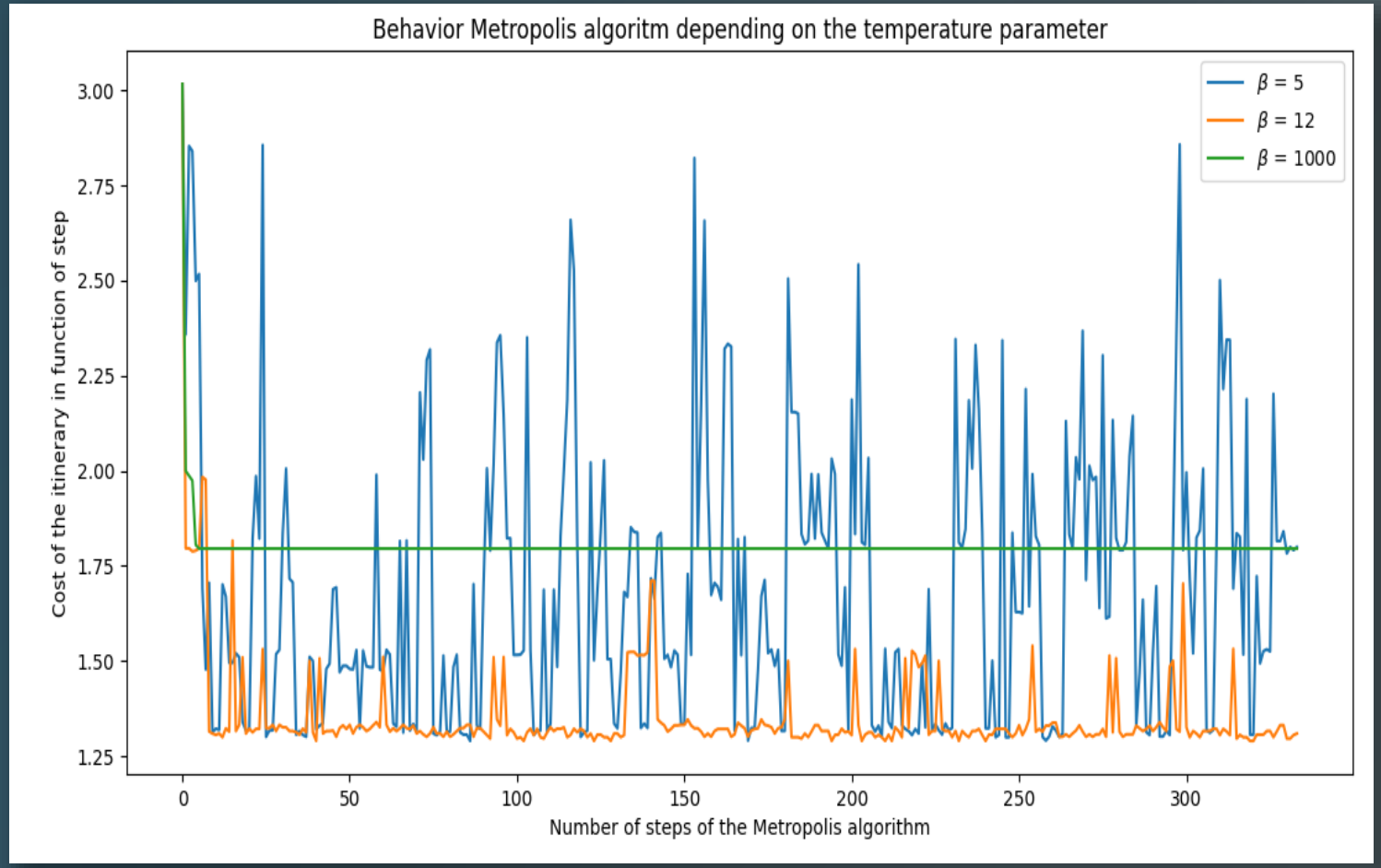



**Figure 2:** Low temperatures converge fast, but not necessarily to optimum. High temperatures are random walks.

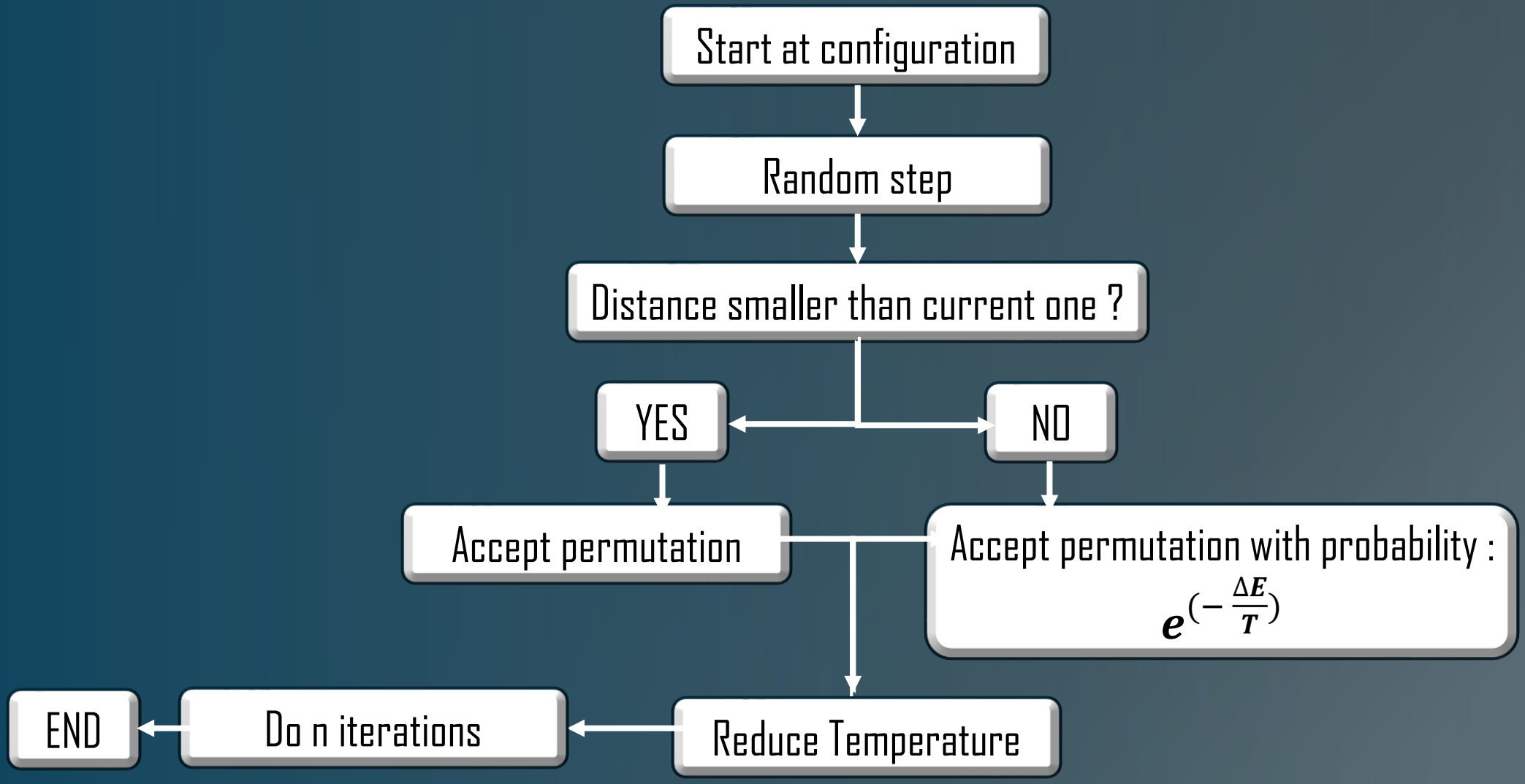
# { "Monte Carlo Methods : Stimulated annealing" }

The name 'Stimulated annealing' comes from metallurgy, where metal is heated and then cooled to change its properties.
In the same spirit, a cooling step is added to the algorithm. This allows to begin the search globally. Then, as temperature decreases, we converge to an optimum.

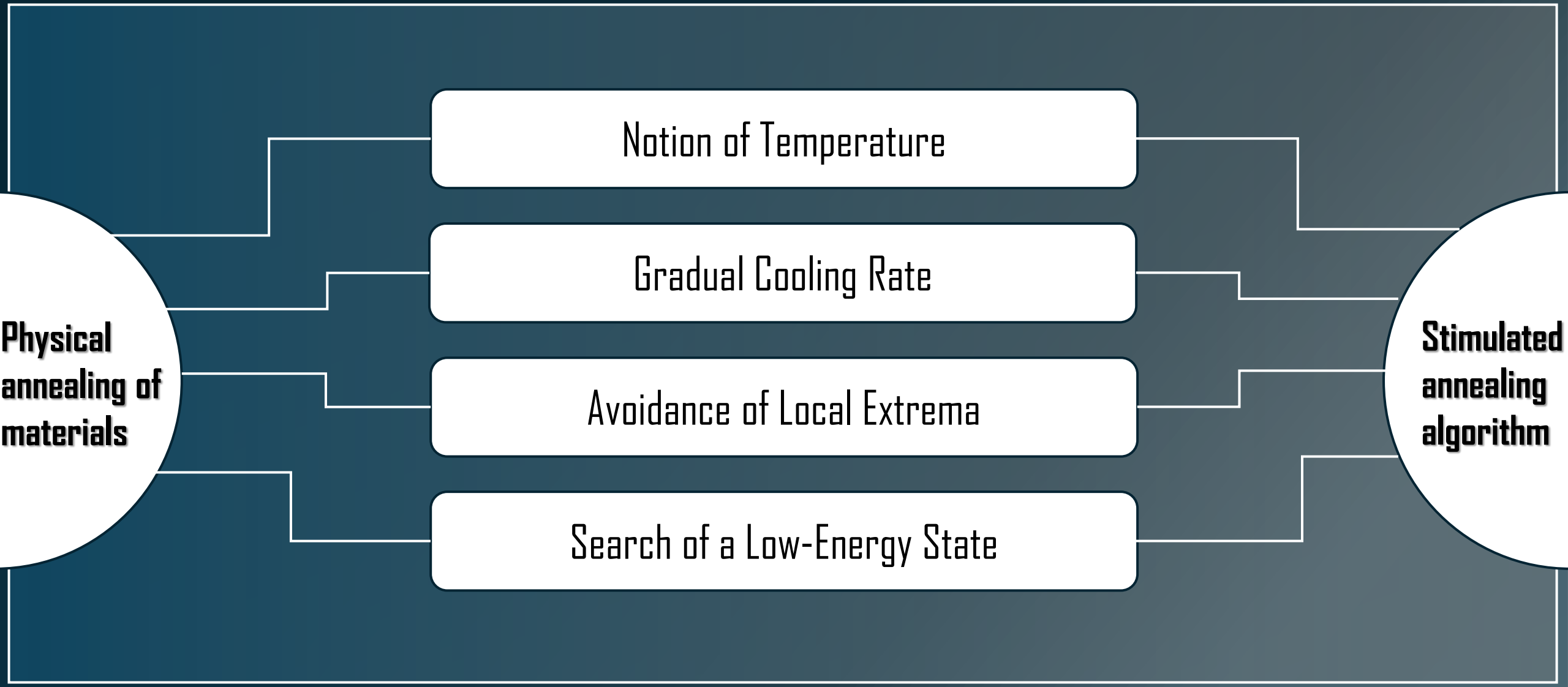
It has ben proven (D. Geman & S. Geman, 1993) that convergence is ensured for $T(t) = T_0/\log(1+t)$

But other ways of updating the temperature are possible (H. Szu, 1987) (L. Ingber, 1988)

# "Monte Carlo Methods : Stimulated annealing"



# { " Analogy " }


Physical annealing of materials
Notion of Temperature
Gradual Cooling Rate
Avoidance of Local Extrema
Search of a Low-Energy State
Stimulated annealing algorithm


METHODOLOGY

# { " Simplification" }

## First Approach

Our first approach to understand the problem was to search for the optimal distance between 6 cities. We did this by using both the stimulated annealing algorithm and the greedy algorithm.

However, doing big optimizations while using spherical coordinates proved to be cumbersome.
Not only that, but we did not know the actual optimal route, so it was impossible to know how close we were to reach the minimum.

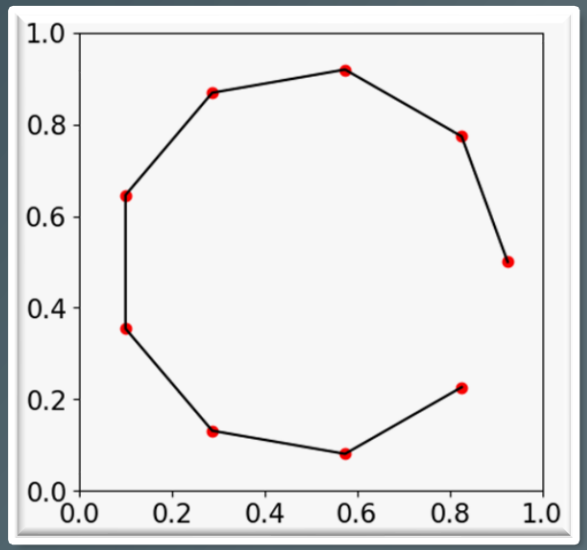
So, we did what physicist do best, and did some approximations to simplify the problem.

# { " Simplification" }

We decided to arrange the 'cities' in a polygon, as the optimal distance would simply be the perimeter.

Spherical Geometry

1x1 square plane




Point configurations in a polygon structure

S.Kirkpatrick; C. D. Gelatt; M. P. Vecchi ,1983

# { " Simplification" }

## Rejected simplification

Manhattan (Taxicab) distance instead of Euclidean distance

— $d_E(\overrightarrow{p_1}, \overrightarrow{p_2}) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

— $d_T(\overrightarrow{p_1}, \overrightarrow{p_2}) = |x_1 - x_2| + |y_1 - y_2|$

— $d_S(\overrightarrow{p_1}, \overrightarrow{p_2}) = Rarcos(\sin(\varphi_1)\sin(\varphi_2) + \cos(\varphi_1)\cos(\varphi_2)\cos(\theta_1 - \theta_2)$

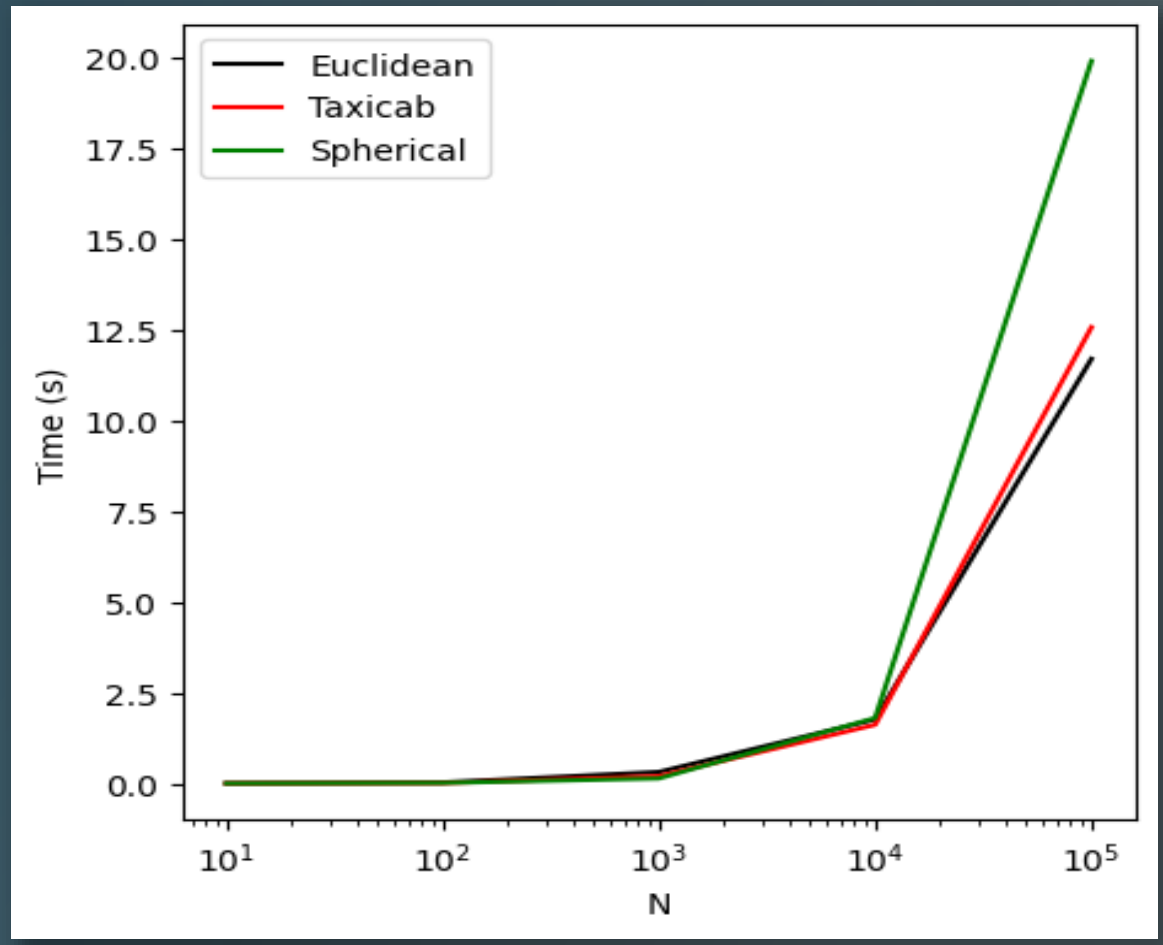



**Figure 3:** Time of computation for calculating the cost of a list of 20 points N times

# “Strategy”

What we did :

1. We calculated the time needed for obtaining the best itinerary using the greedy algorithm.

2. We then compared the different cooling schedules :

First the same but with different cooling rate $\alpha$. This was done by running the same optimization with same initial condition 100 times and getting an average of the reached minimum as a function of the iteration of the algorithm.

We compared the algorithms side by side, given an alpha for each one.

3. We then chose one cooling method to apply it to a non-abstract optimization problem.

# {"Different cooling schedules"}

$$T_k = T_0 \alpha^k$$

Exponential

$$T_k = \frac{T_0}{1 + \alpha \log(k+1)}$$

Logarithmic

$$T_k = \frac{T_0}{1 + \alpha k^2}$$

Quadratic

(Harold Szu) (L. Ingber, 1988)

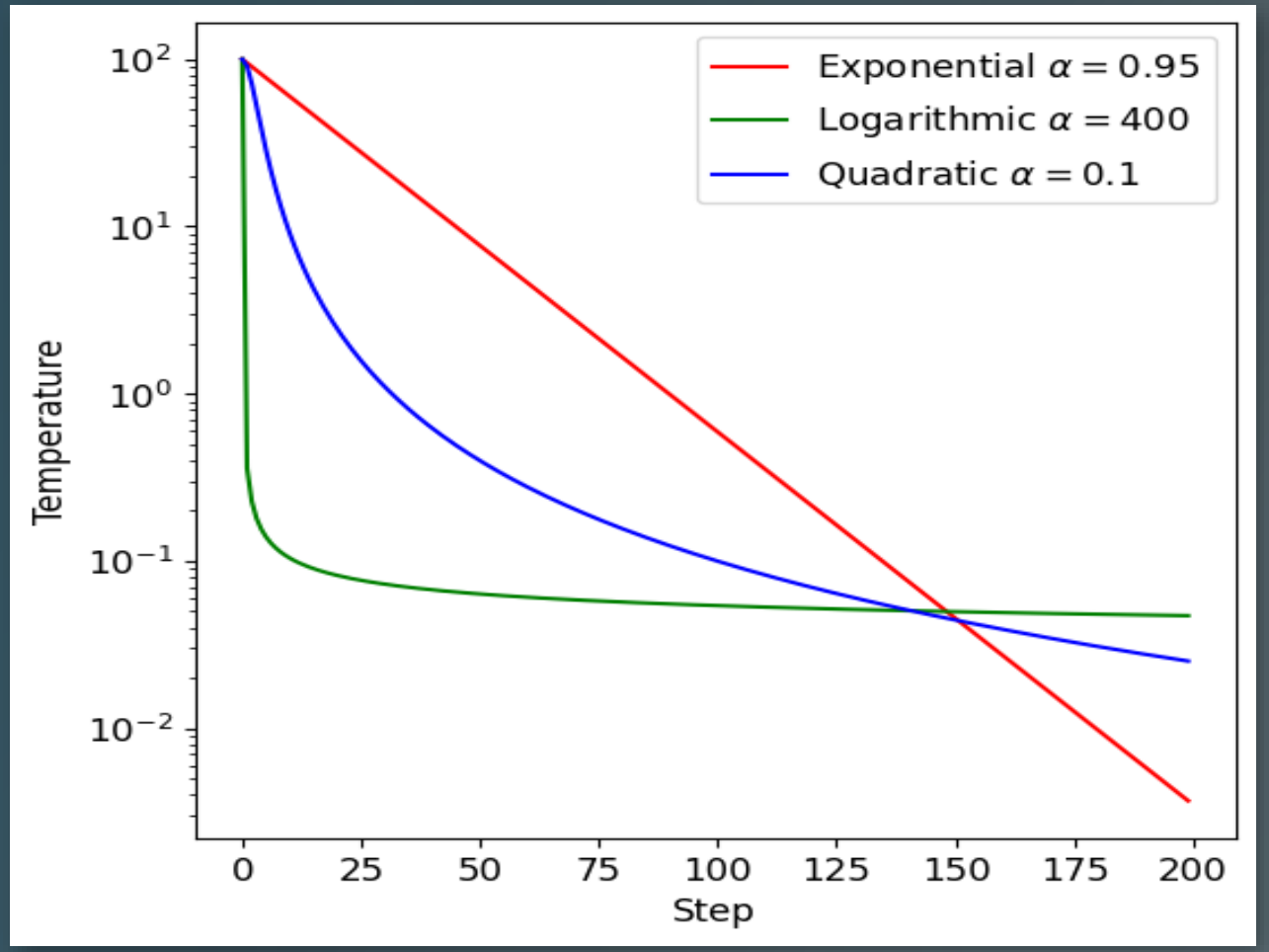
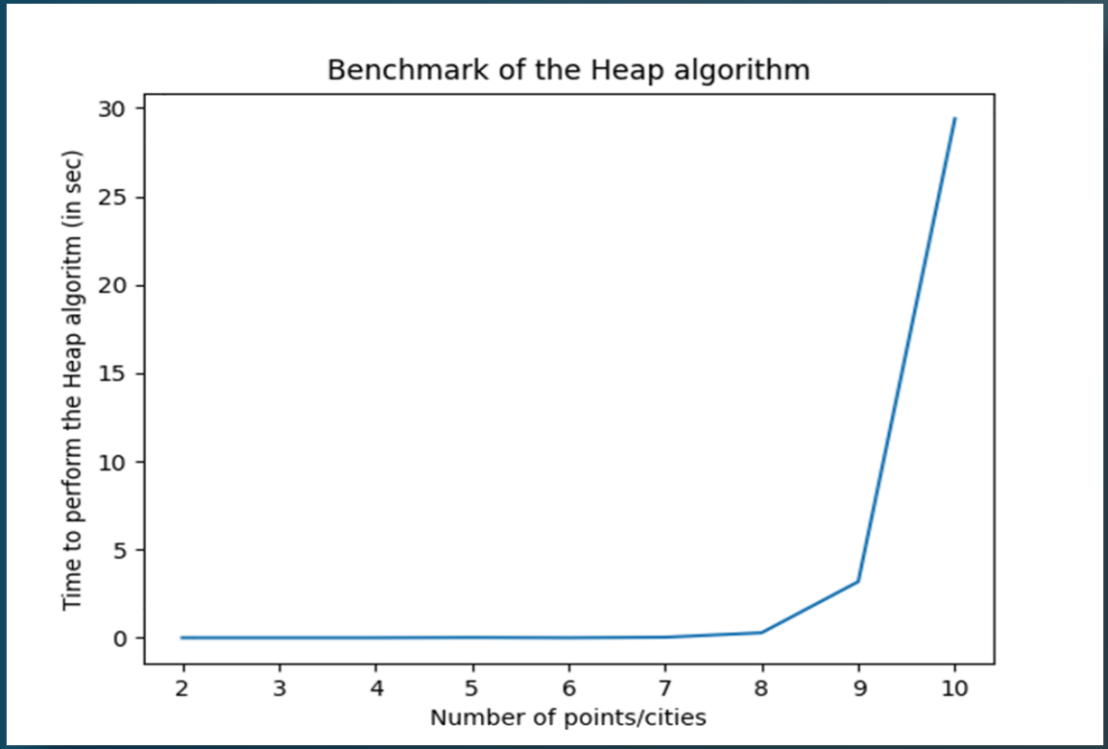


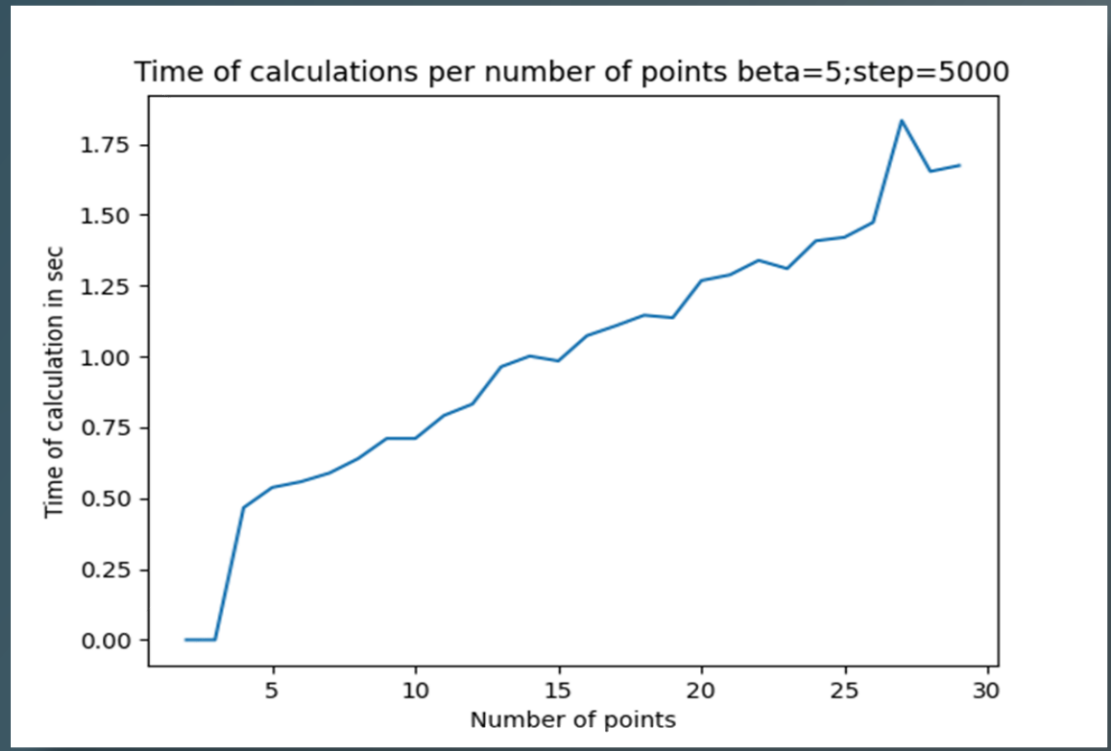

Figure 3: Temperature decrease for the 3 $\alpha$ - dependent methods


ANALYSIS

# {"Greedy algorithm vs Metropolis Hasting algorithm"}

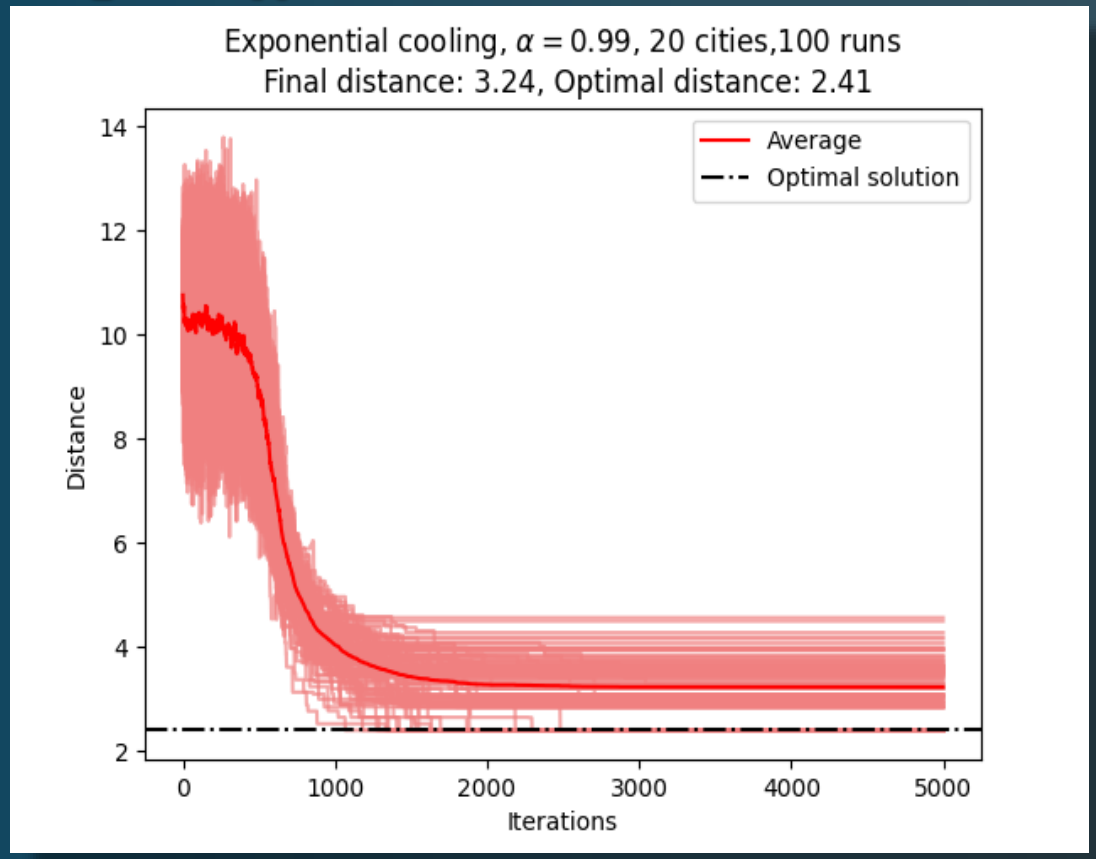
The first result that we want to empathize is the difference in execution time between the two algorithms as a function of the number of cities. On the first graph we observe an exponential growth of execution time, on the second one we observe a rather linear behavior. We could only do calculations for up to 10 cities with the greedy algorithm because our computer crashed multiple times during the process.
Furthermore, we can observe that with the metropolis algorithm we can calculate to really huge set of points.



# {"α exponential - side by side"}

Runs using exponential cooling for different alphas, slower cooling yields better results, indicating that in fast cooling the system gets trapped in local minimum



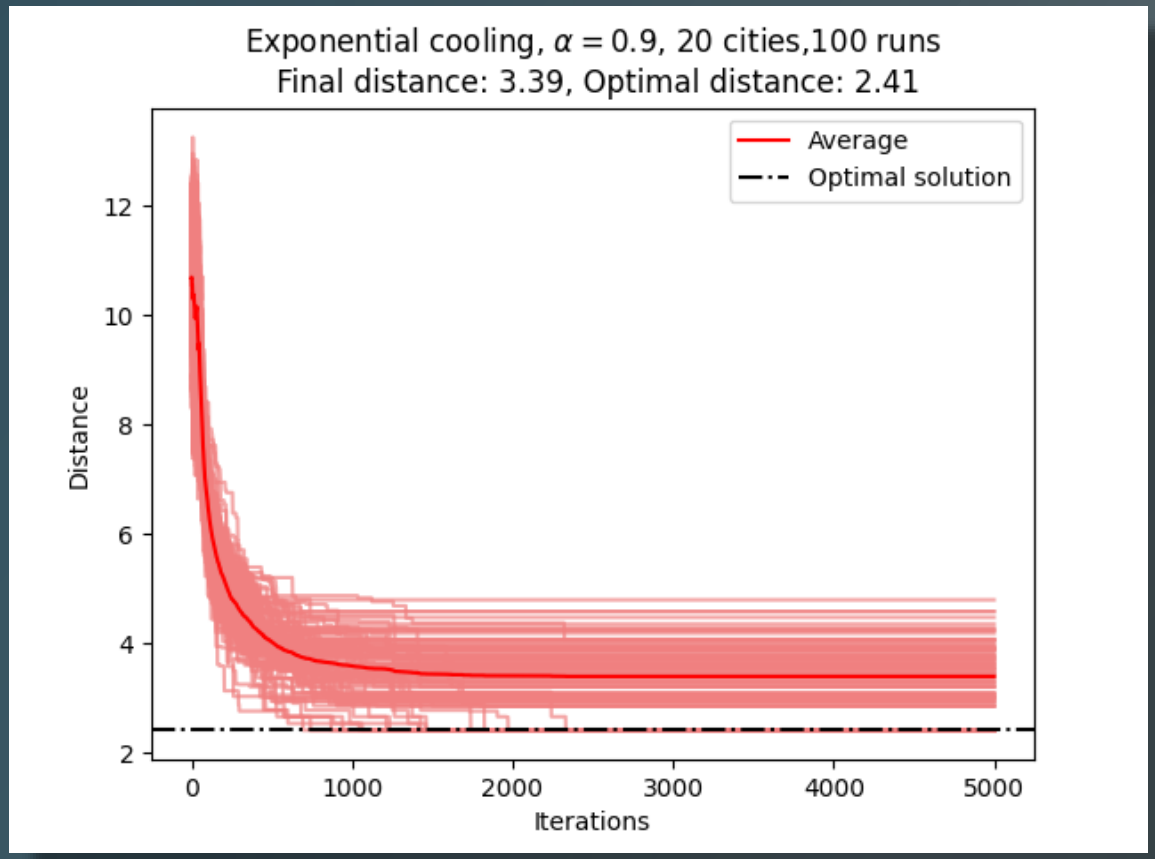
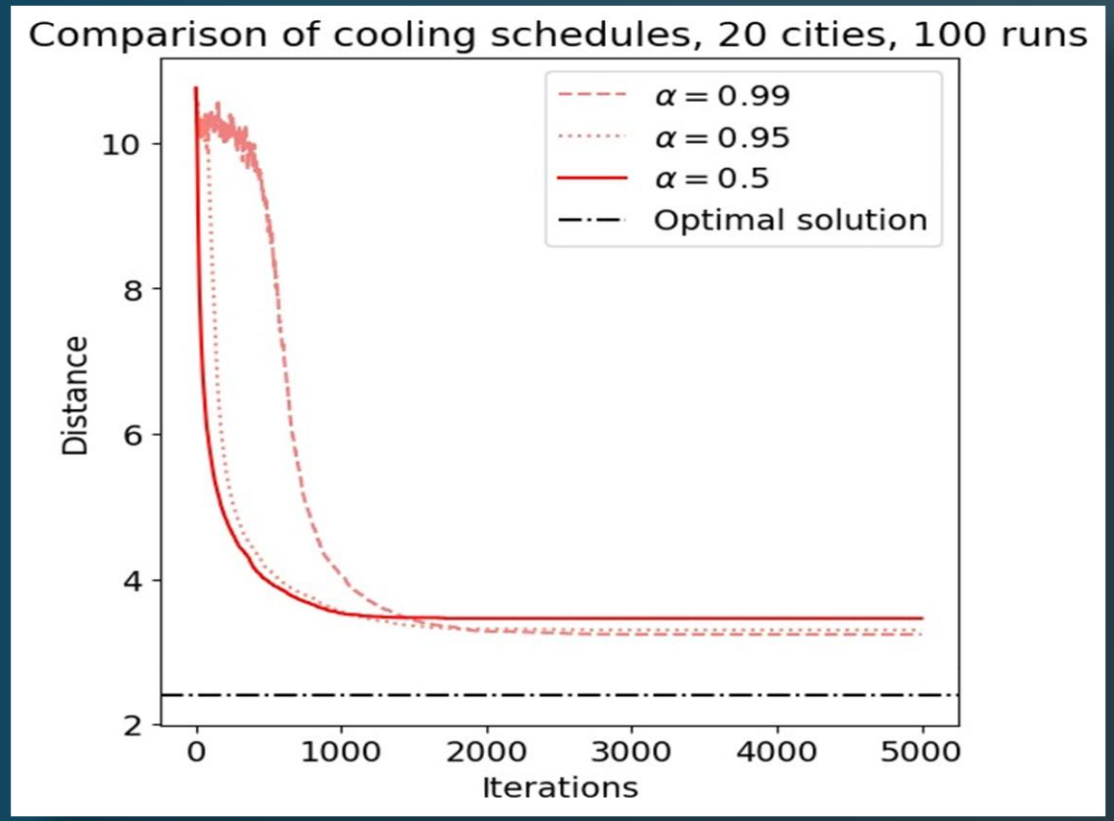




**Figure 4:** Comparison of the average distance for two values for α parameter using the exponential cooling method and maintaining the number of Cities and time runs

# {"Exponential Cooling - α "}

The difference on the evolution of temperature and the yielding of a marginally better result in the case of the slow cooling



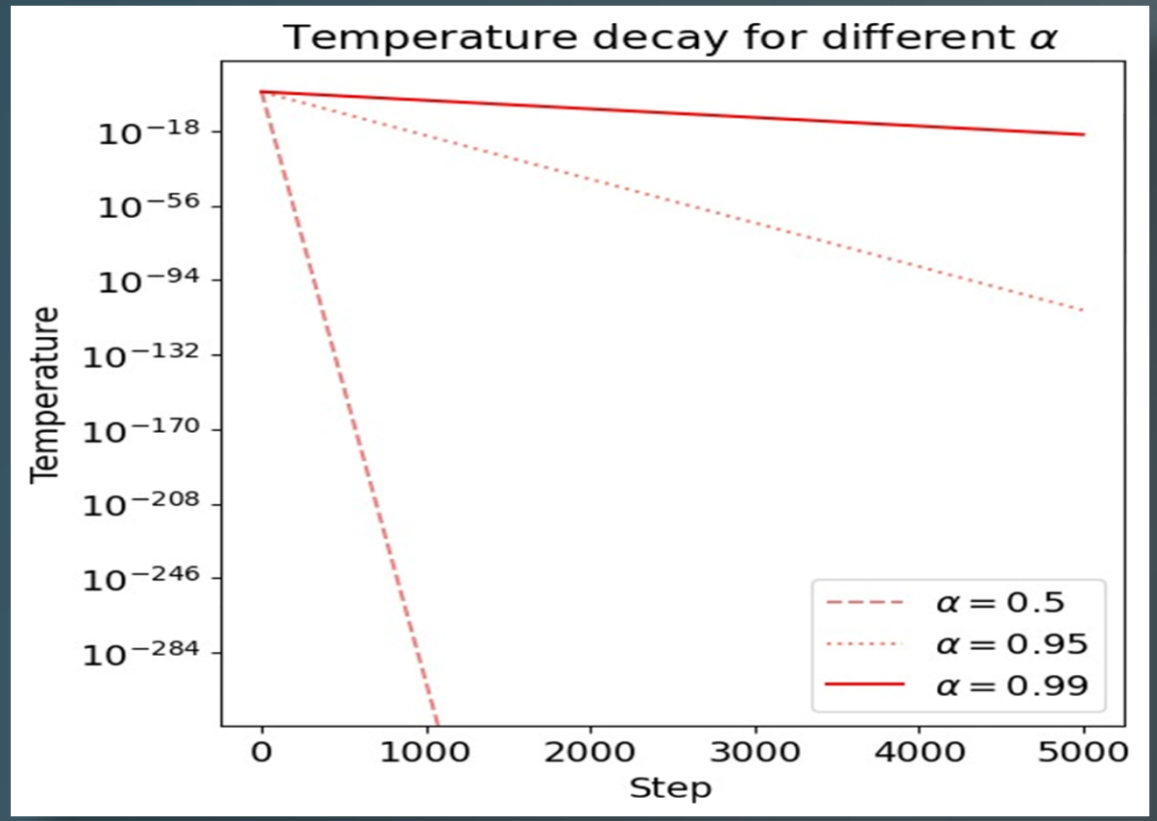
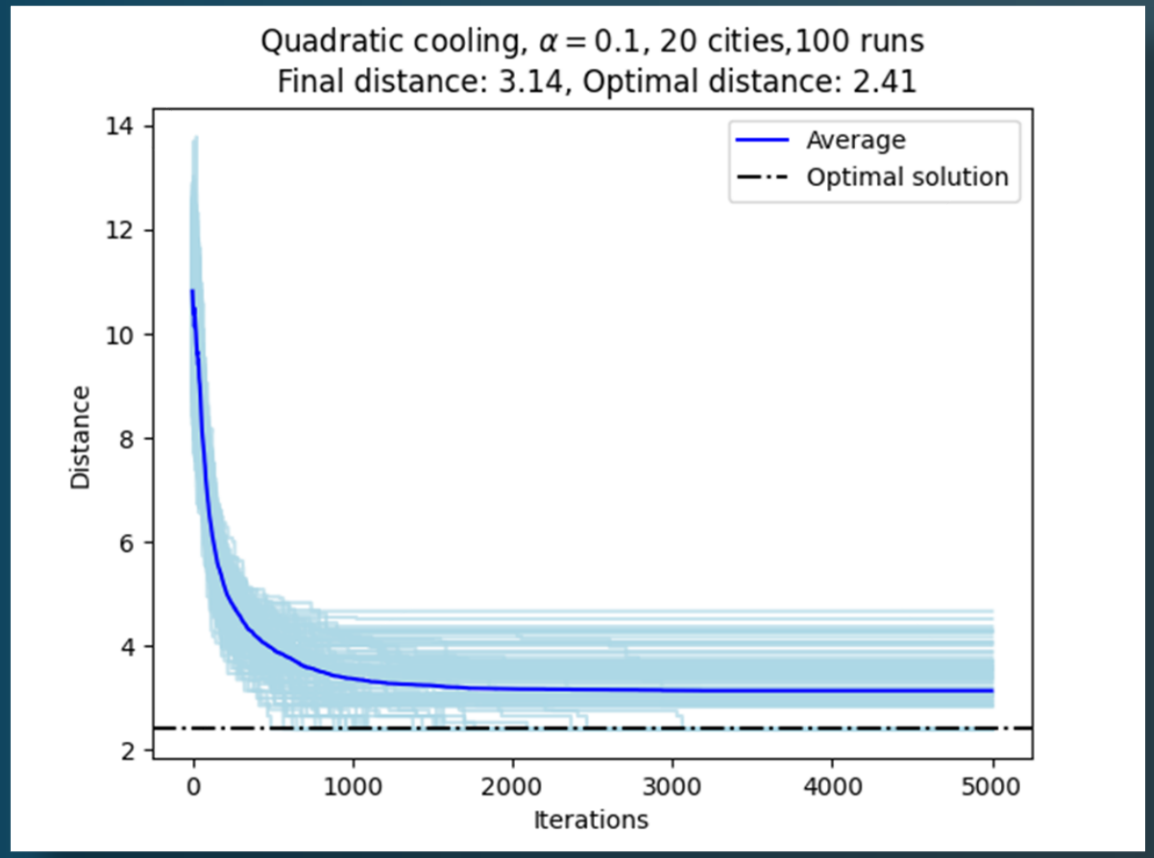




**Figure 5:** Comparison ot the decrease of both Distance and Temperature for different α values in function of iterations

# {"Quadratic cooling - α "}

Results are pretty similar to that of the exponential cooling, both quantitatively and qualitatively. Fast cooling gets trapped,



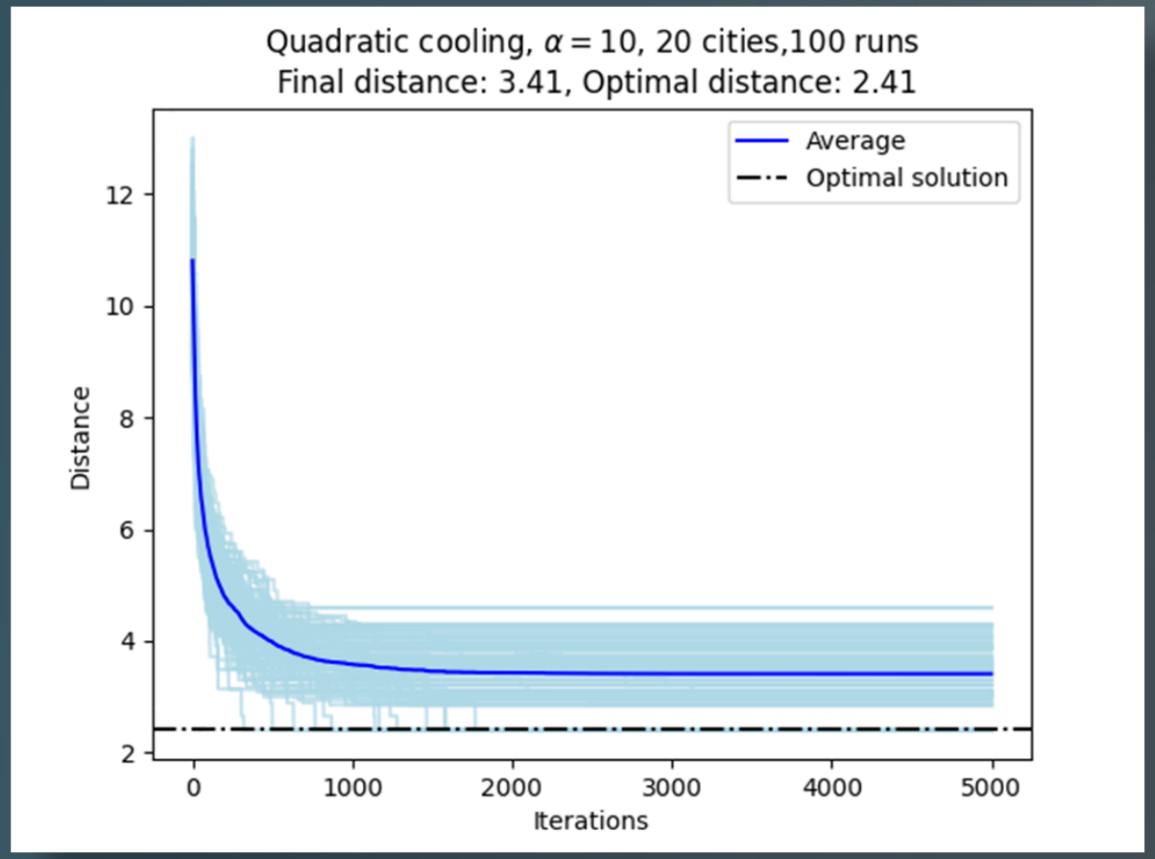
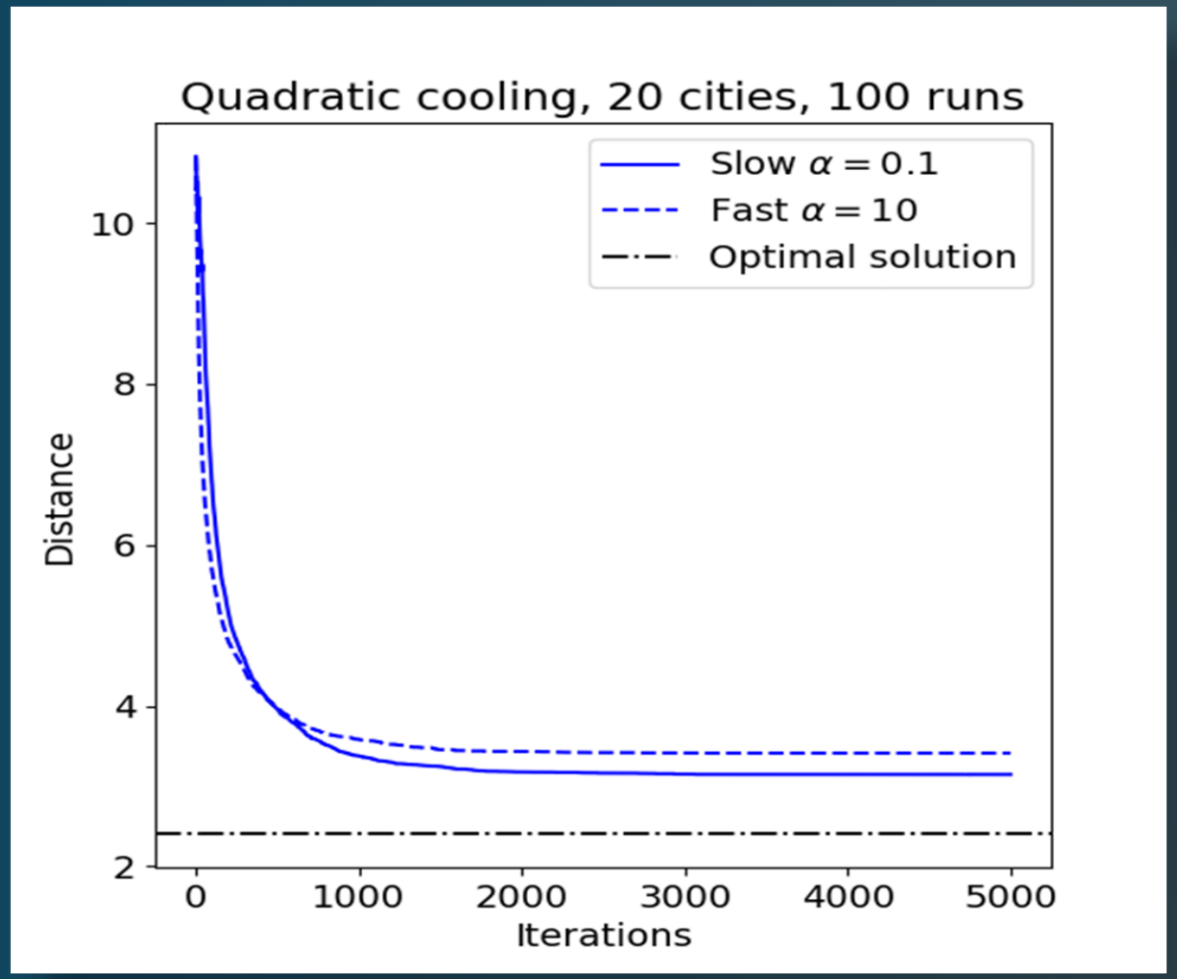




**Figure 6**: Comparison of the average distance in function of number of iterations with two α values using the Quadratic cooling with same number of cities and time runs

# {"Quadratic cooling - α"}



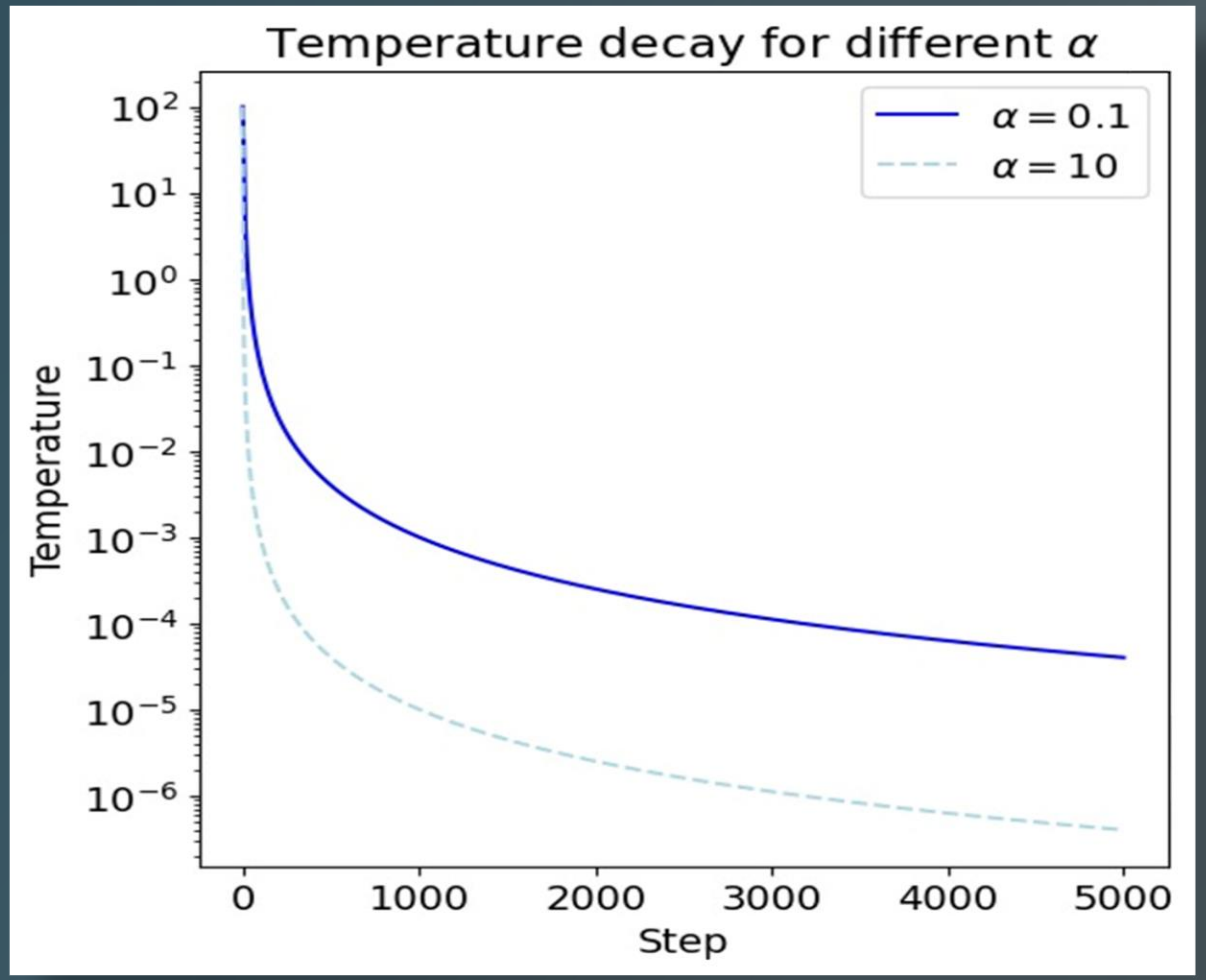
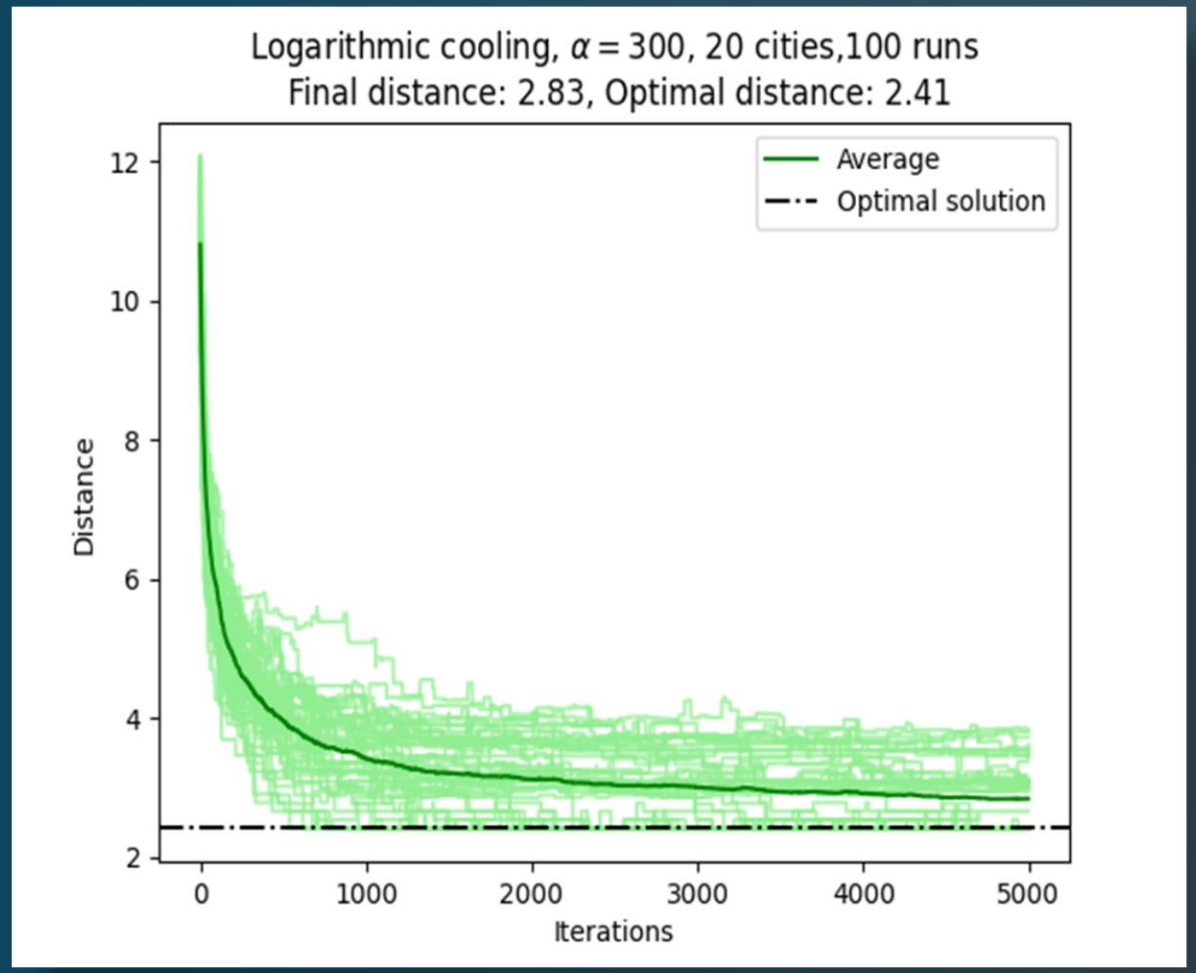




**Figure 7:** Comparison of evolution of both the distance and the temperature using both slow and fast cooling. Fast cooling is better for short times, but gets trapped

# {"Logarithmic cooling - α"}



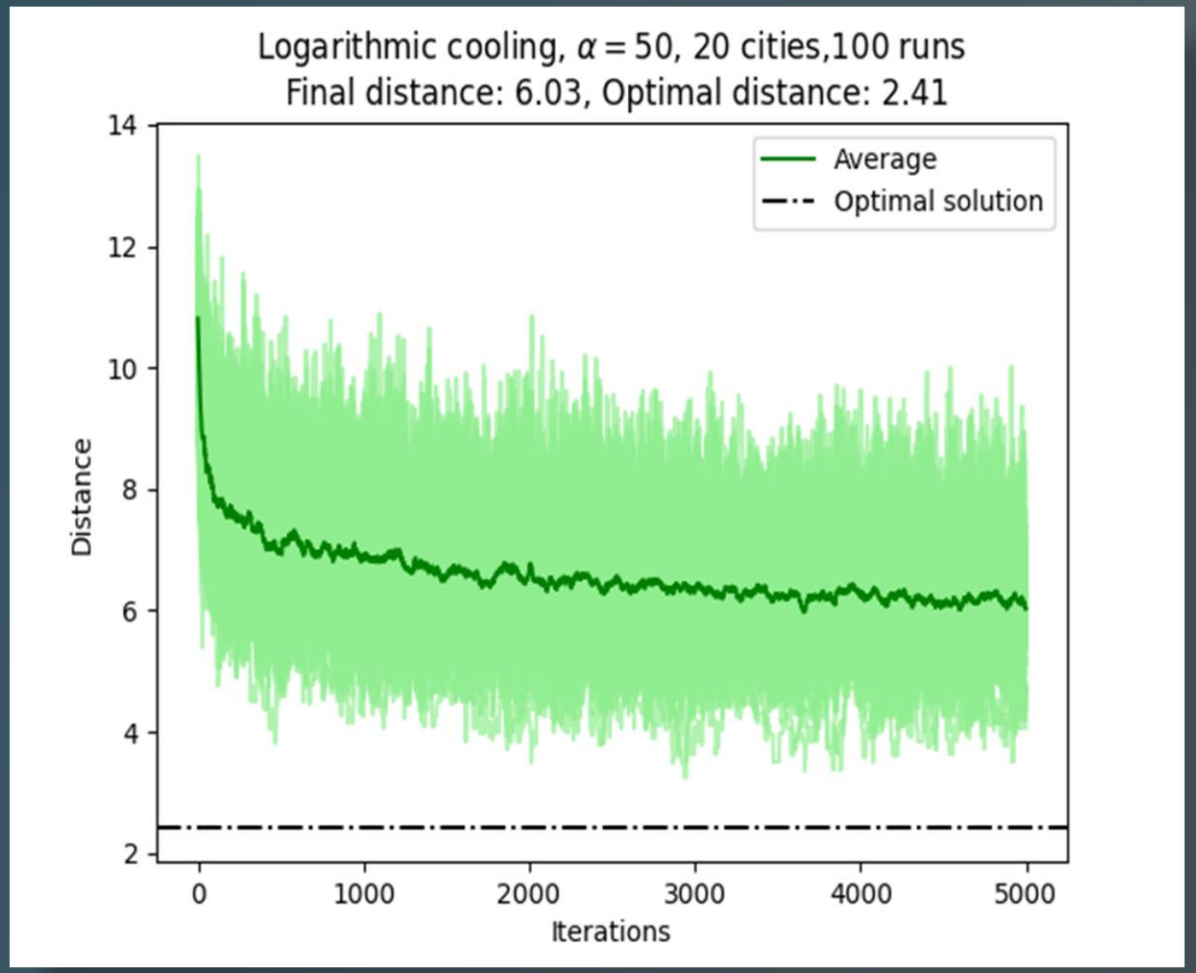
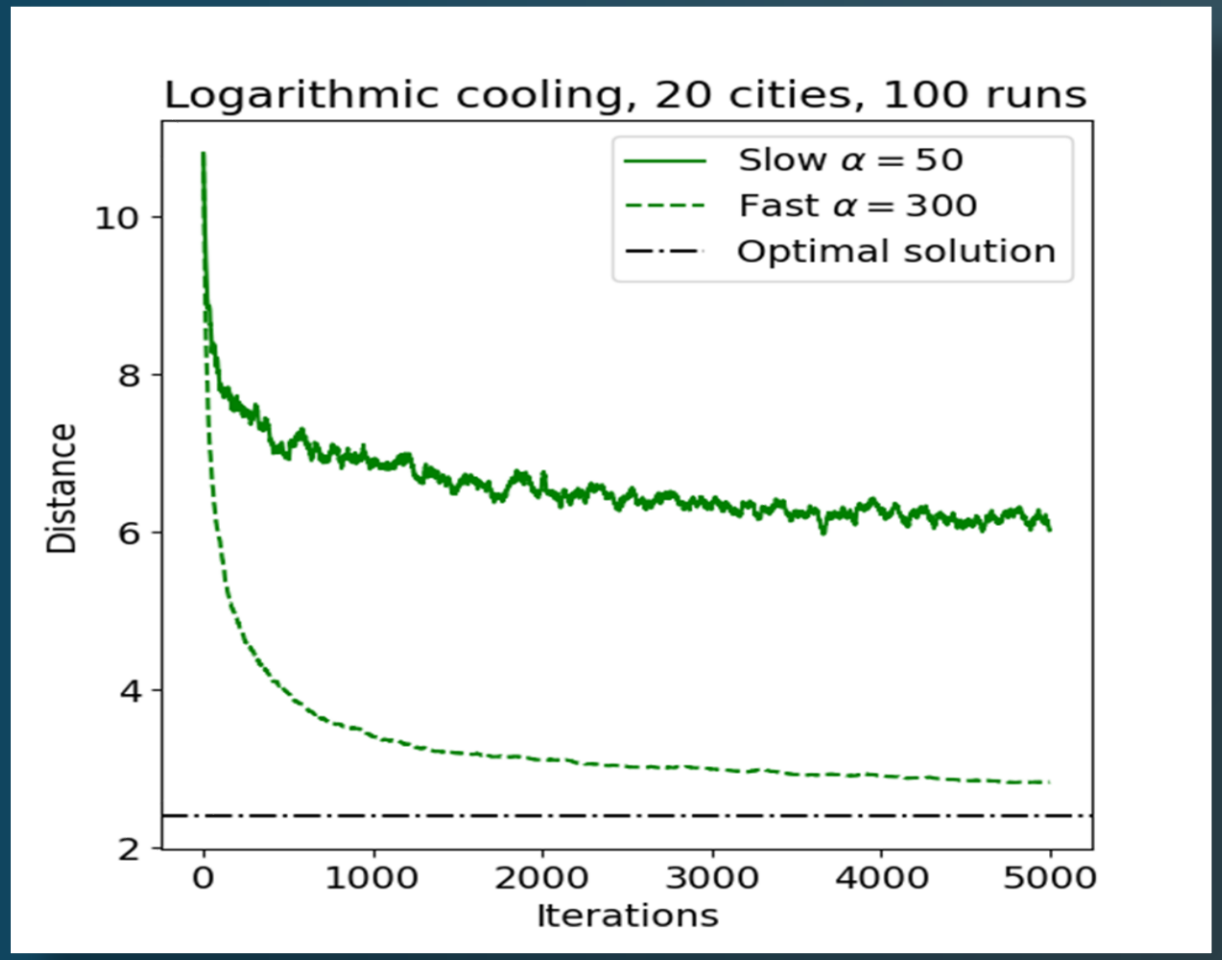




**Figure 8:** Logarithmic cooling didnt present the same result. Slow cooling was too slow in this case. Convergence is ensured (Geman & German, 1993), however.

# {"Logarithmic cooling - α"}



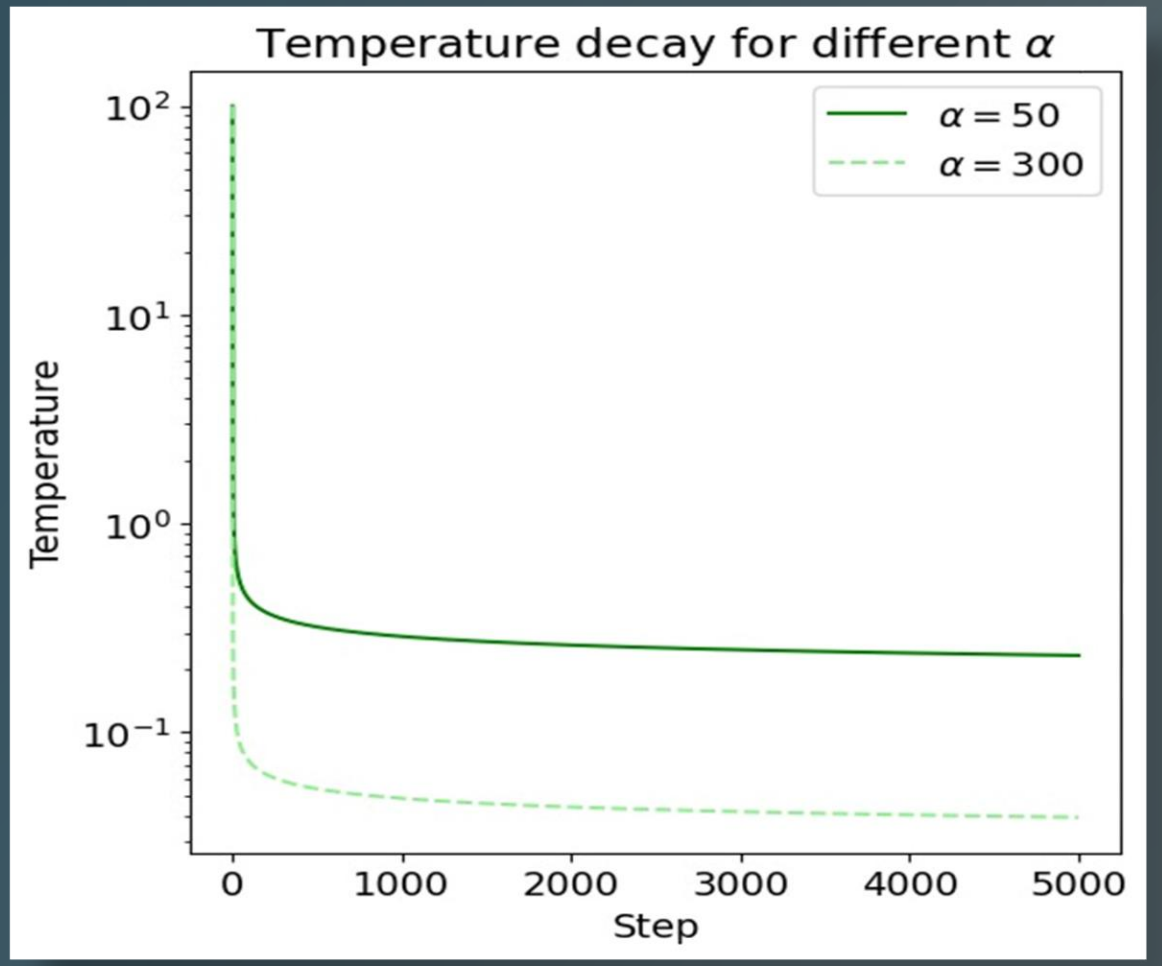
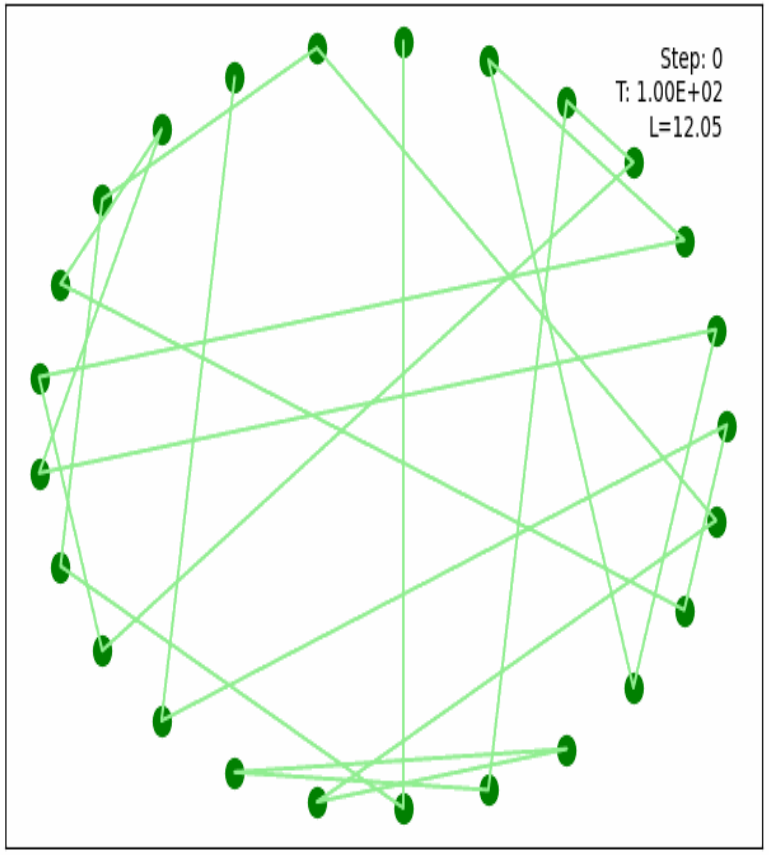



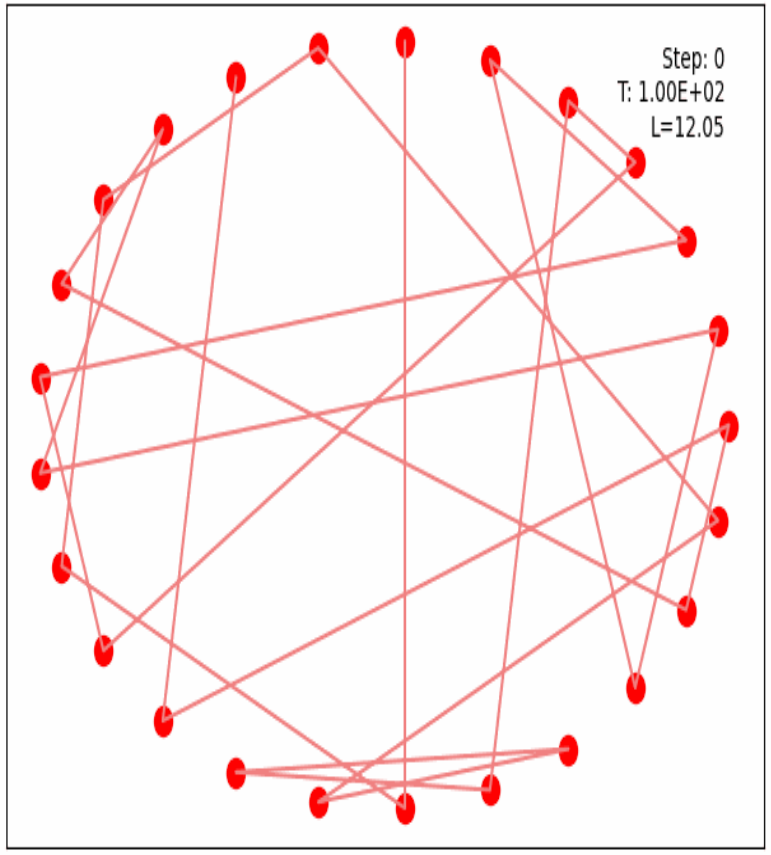

**Figure 9:** After a fast global exploration, logarithmic cooling seems to converge, but it doesn't, it is just very slow

# {"Cooling schedules comparison - qualitative"}

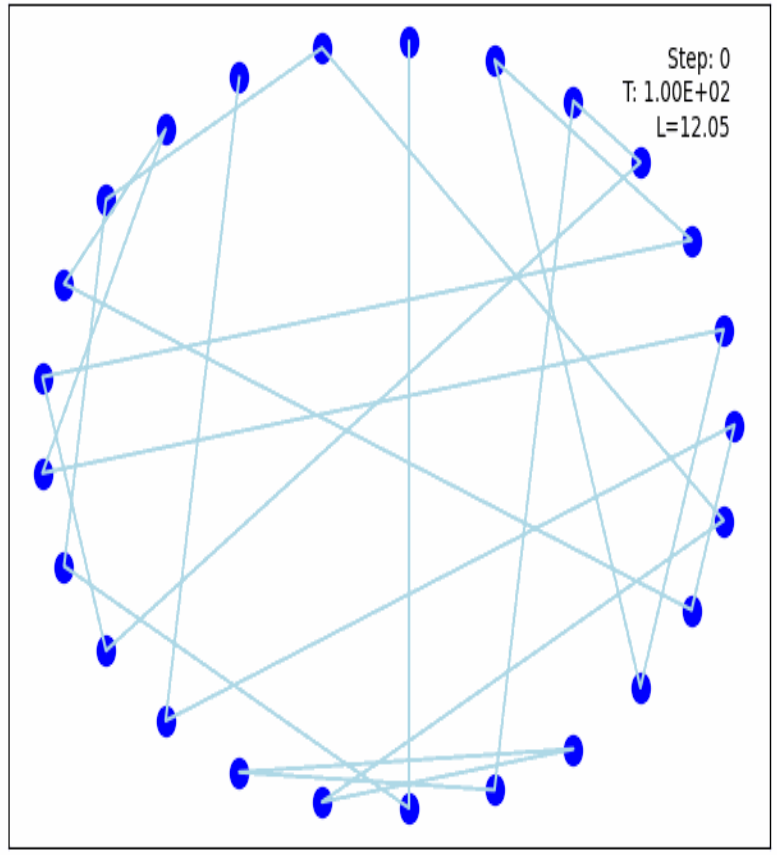
Logarithmic cooling with $\alpha = 200$



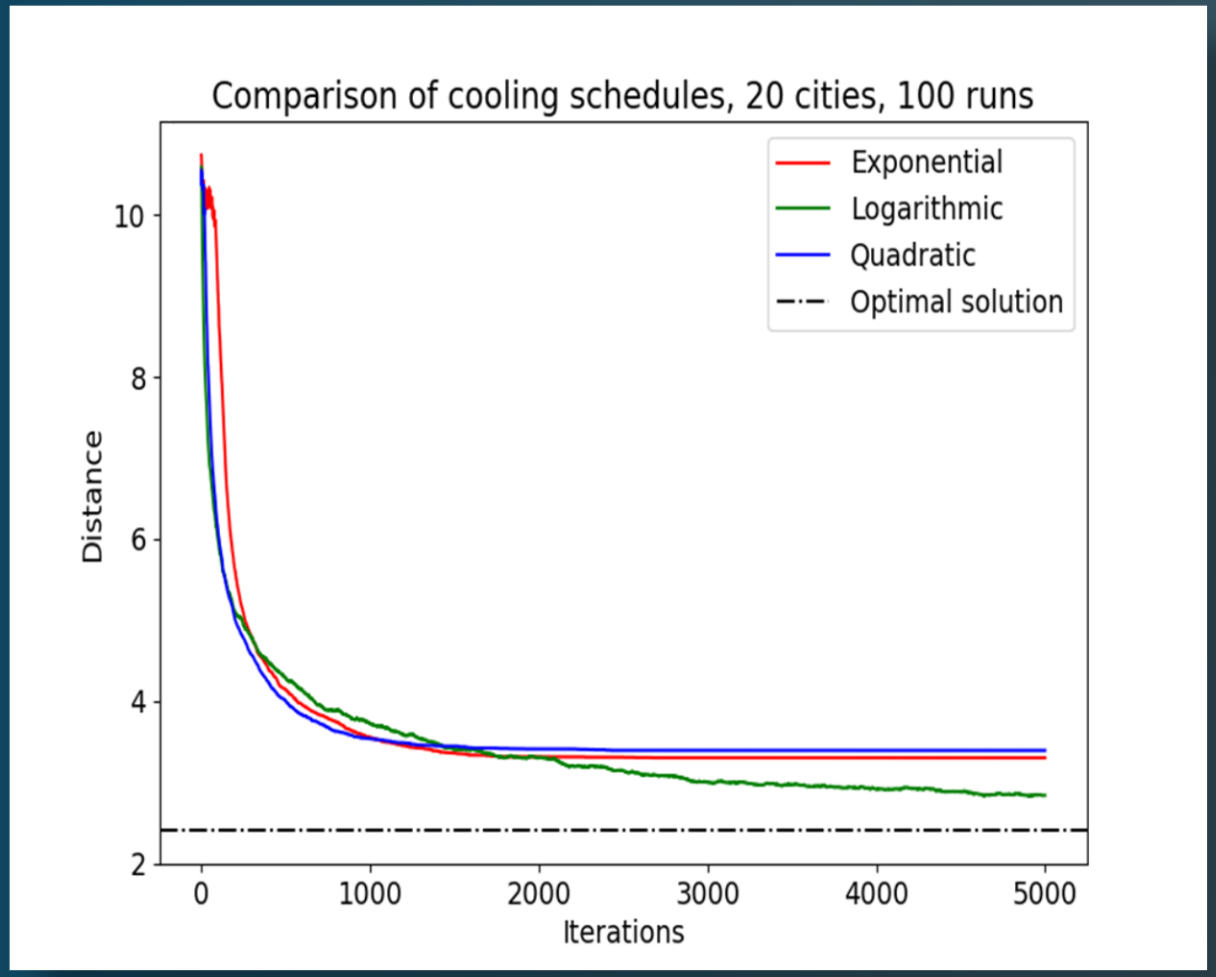

Exponential cooling with $\alpha = 0.95$




Quadratic cooling with $\alpha = 0.3$



# {"Cooling schedules comparison - qualitative"}



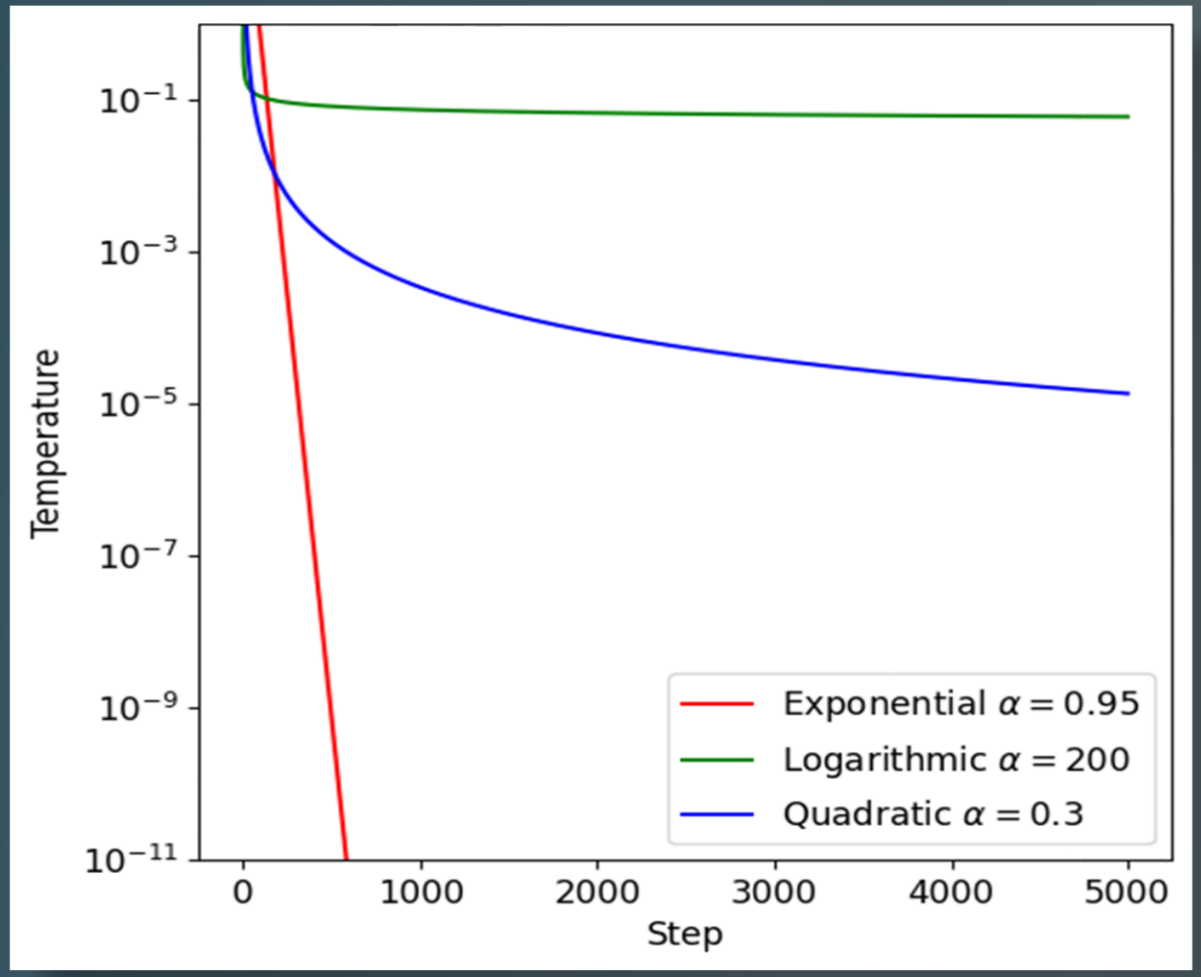




**Figure 10:** Exponential and quadratic converge quickly but to the optimal route, logarithmic looks like could reach the optimum if left running long enough


Application to important problems
"Traveling ~~Salesman~~ Dragonborn"

# Traveling Dragonborn

The infamous quest *No Stone Unturned* in Skyrim Asks the player to find 24 hidden gems called *Stones of Barenziah* around the world.

We will apply the traveling salesman problem to spend the least amount of time in the quest.

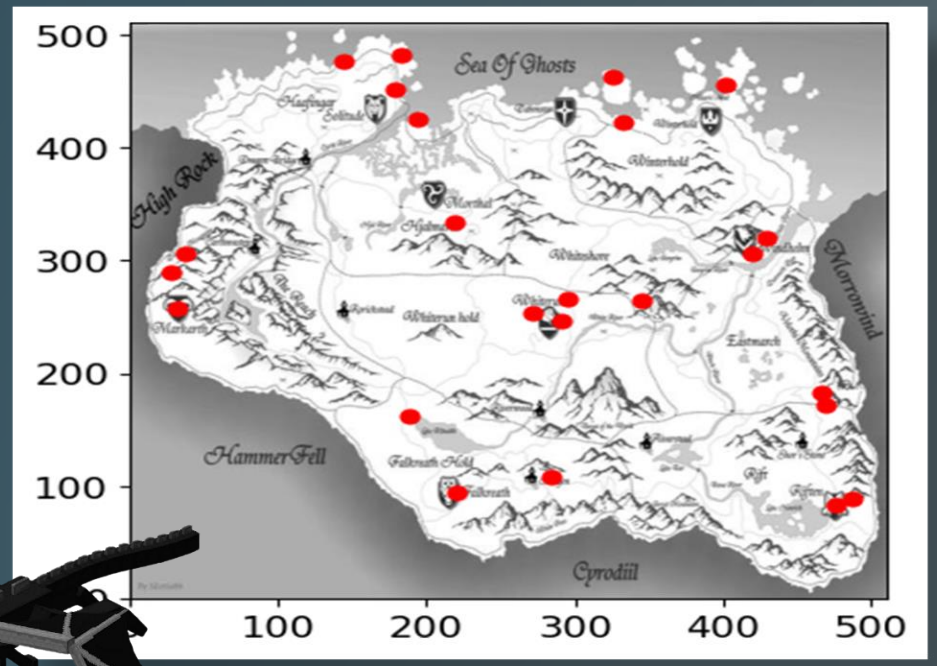
- No fast travel
- Dragonborn has Dragonrend shout, that allows them to mount a dragon and fly, thus not being impeded by geography.
- Quest is not softlocked behind Diplomatic Immunity quest.
- Dragonborn leaves from the thieves' guild at Riften

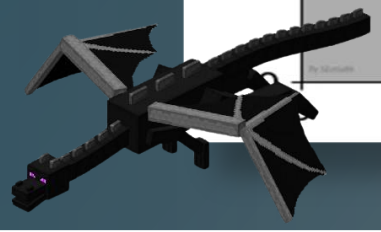
The Elder Scrolls V
SKYRIM

# { "Squarification" }

First, we resize Skyrim map to make it compatible to our square world, then we normalize it into a 1x1 square, then we apply our logarithmic stimulated annealing. In red, the locations of the Stones of Barenziah

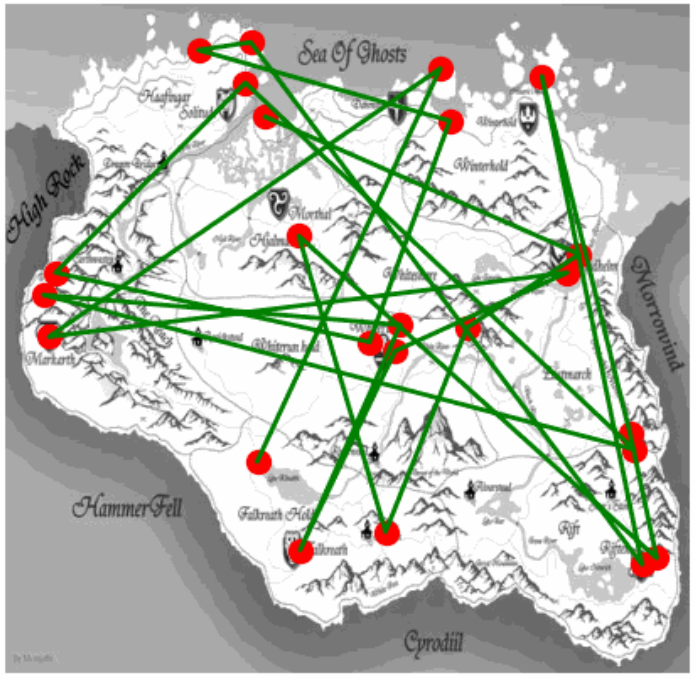


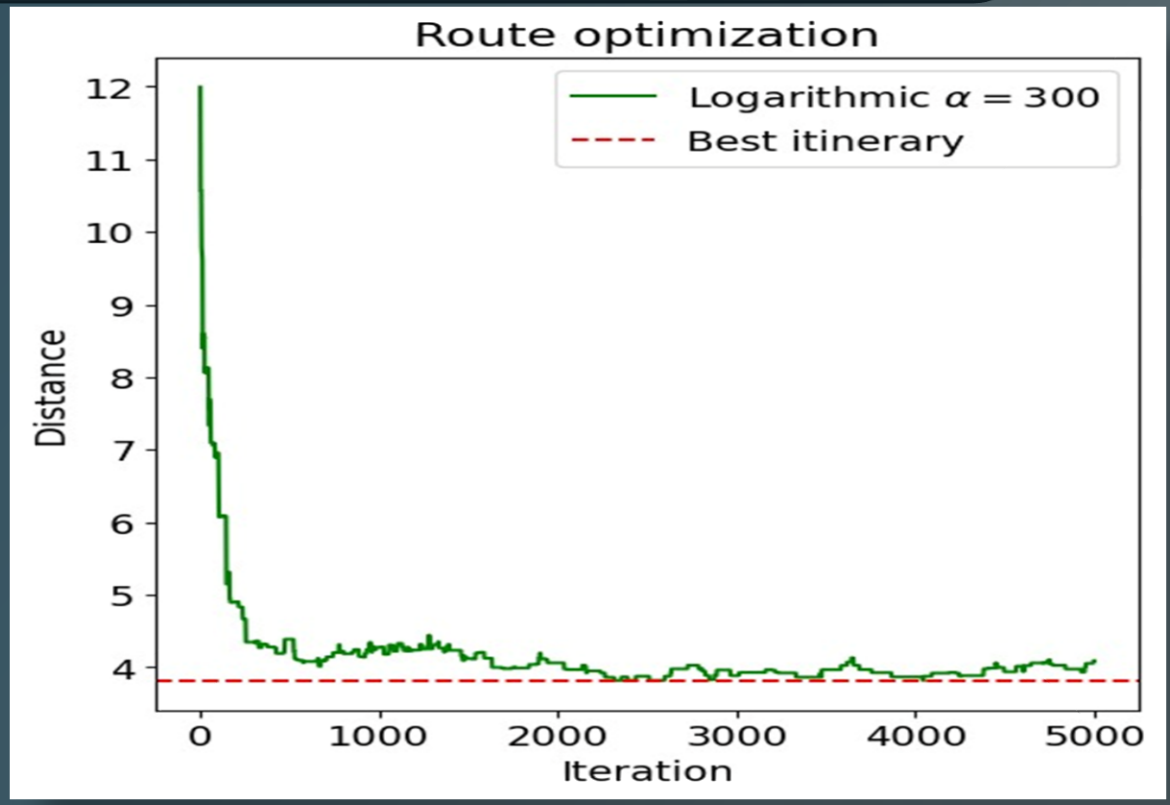



# {"Application of optimization"}

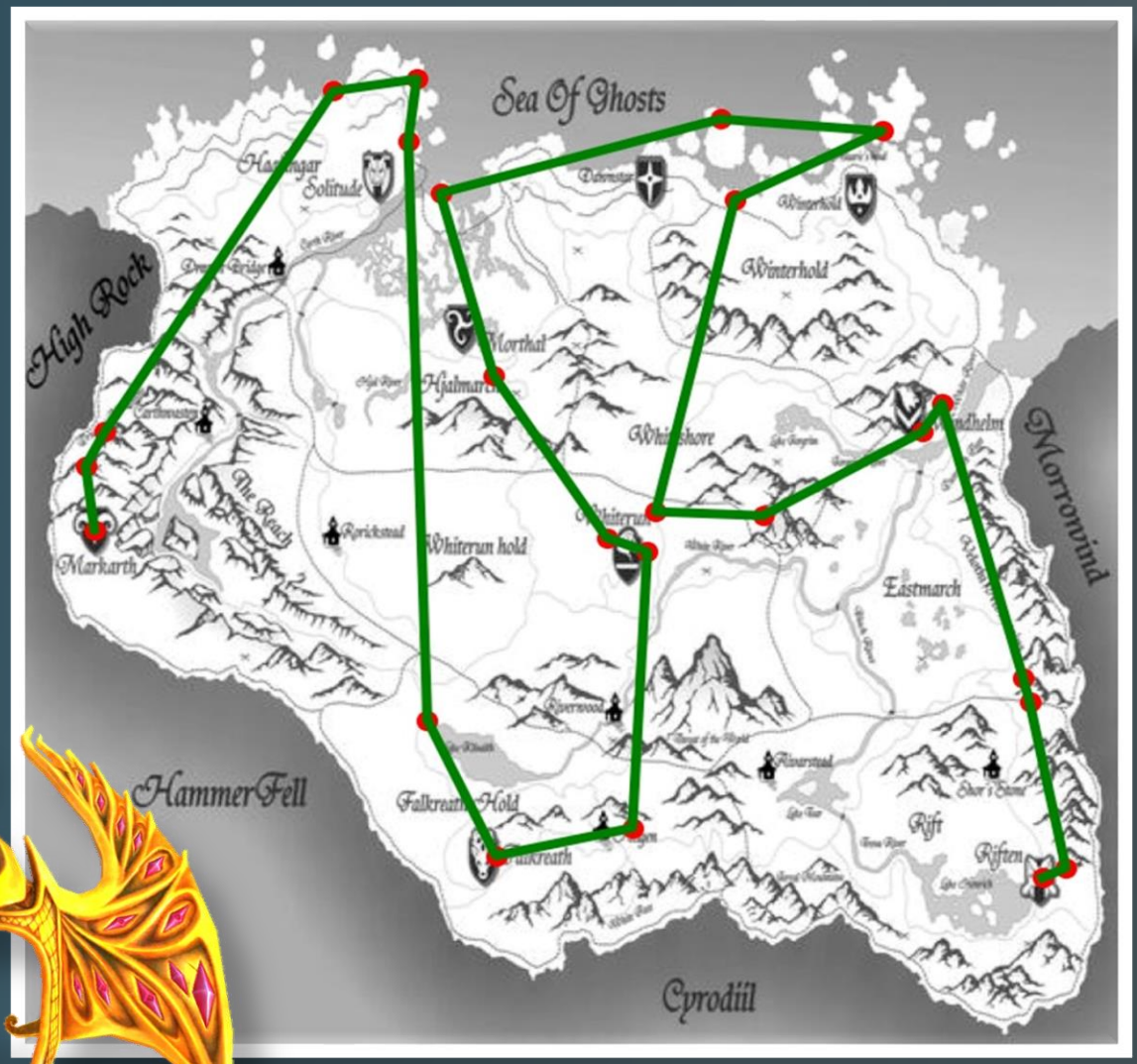
We will use the logarithmic cooling because it was the one that yielded the best results in our analysis. We chose a value of $\alpha = 300$. Below optimization process and the evolution of Distance



# { "Traveling Dragonborn result" }

We successfully obtained the optimal route to spend the least time possible in the quest and obtain the **Crown of Barenziah**





CONCLUSION

# { “Comments/feedback” }

- A greedy algorithm ensures optimal result  but is limited and inefficient.
- The problem is dependent on initial state.
- Cooling schedule depends on needs :

  If the configuration space is very ‘rocky’ (many local extrema), logarithmic cooling is best.

  If you know your initial guess, exponential cooling or quadratic are better to choose .

  If you don’t have information about either the configuration space or quality of guess, logarithmic cooling is also a safe bet.

# { "Outlook" }

- INCLUDING RESTARTS :

  First method : each time the system converges, increase temperature to keep looking for the best option.

  Second method : if the system converges, go back to previous better option with higher temperature.

- Apply this to other systems (see S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi (1983))
- Study other cooling schedules
- Study behavior change depending on parameter $\alpha$


THE END
Thank you for your attention