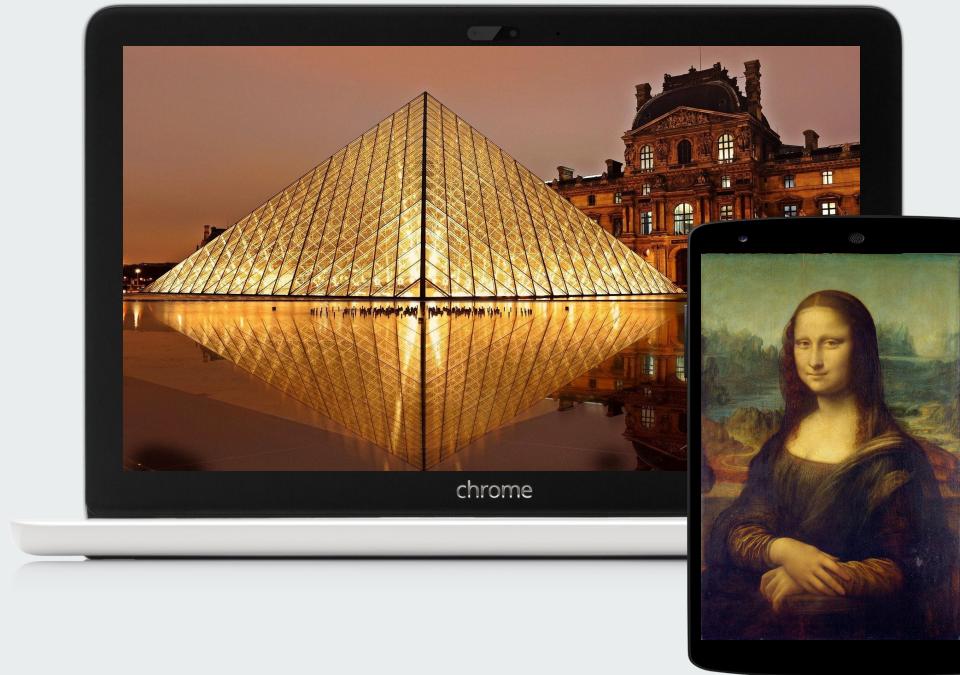


---

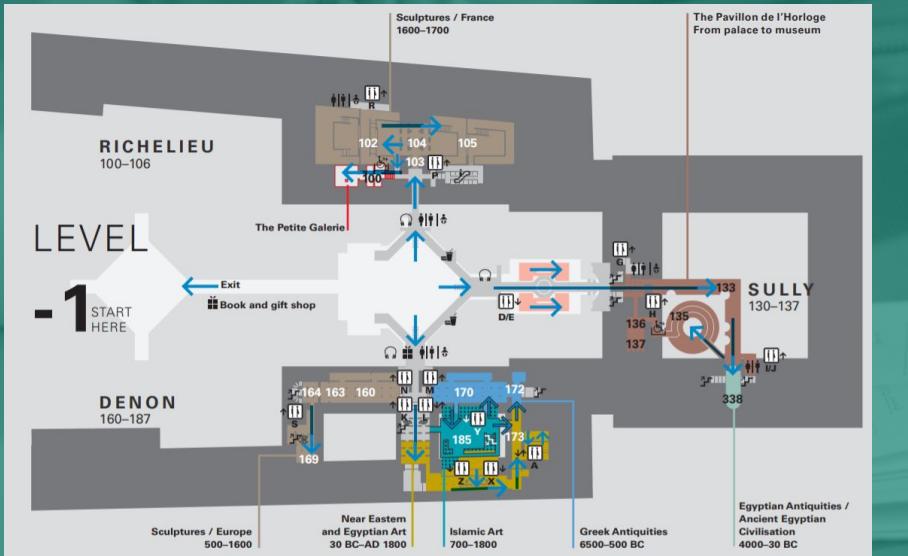
# The Museum Problem

Édition du musée du Louvre

Team - Sic Mundus



# Problem Statement



Optimize the route for a tourist visiting The Louvre Museum, such that the satisfaction level is maximised by visiting all/select exhibits in a single working day. With coinciding exit and entry points.

## Design Variables

1.  $(2 \times N C_2)$  - Path indicator variables
2. (N) - Time to be spent at each exhibit

# Problem 1

---

- This is the simplest version of the museum path optimization problem, which requires the tourist to visit all the exhibits located at lattice points separated by known distances.
- The objective is to find an optimized sequence of visiting the exhibits such that the total path length is minimised.

$$\min \sum_i \sum_{j=1} c_{ij} y_{ij}$$

$$s.t \sum_{i < k} y_{ik} + \sum_{j > k} y_{kj} = 2, k \in V$$

$$\sum_i \sum_j y_{ij} \leq |S| - 1, S \subset V, 3 \leq |S| \leq n - 3$$

$$y_{ij} \in \{0, 1\}, \forall [i, j] \in E$$

# Problem 2

---

- Consider a model, where certain points cannot be reached by all of the points in the space of lattice points
  - Models exhibits in museum situated at different floors; can be accessed only from certain entry/exit points.
- Problem is asymmetric, represents paths which don't exist in both directions.
  - Models one-way routes, and/or routes with different departure and arrival rates.

$$\min \sum_i \sum_{j=1} c_{ij} y_{ij}$$

$$s.t \sum_j y_{ij} = 1, i = 0, 1, \dots, n-1$$

$$\sum_i y_{ij} = 1, j = 0, 1, \dots, n-1$$

$$\sum_i \sum_j y_{ij} \leq |S| - 1, S \subset V, 2 \leq |S| \leq n-2$$

$$y_{ij} \in \{0, 1\}, \forall [i, j] \in E$$

$$y_{ij} \in \{0, 1\}, \forall [i, j] \in E'$$

# Problem 3

---

- Consider a model, where the satisfaction level of the tourist needs to be maximised over a fixed interval of time
  - Models a tourist who prioritises visiting exhibits with higher popularity index in order to leave the museum at the end of the day with maximum satisfaction

Note: Satisfaction level based on popularity index of an exhibit is approximated to be a deterministic variable in this case, complexities can be added in the future by considering it to be a probabilistic variable.

$$\max \sum_{i=1}^n s_i \sum_j y_{ij}$$

$$\sum_{i=1}^{n-1} y_{ij} = \sum_{k=2}^n y_{jk} \leq 1, \text{ s.t. } j = 2, \dots, n-1$$

$$y_{ij} \in \{0, 1\} \forall i, j \in V$$

$$\sum_i \sum_j y_{ij} \leq |S| - 1, S \subset V, 2 \leq |S| \leq n-2$$

$$\sum_i \sum_j \left( \frac{c_{ij}}{v} + \tau \right) y_{ij} \leq T_0$$

---

# Optimization Results

Link to GitHub Repository: [Museum-Path-Optimization](#)

# Branch and Bound

---

```
=====
```

Solver Summary:

```
=====
```

```
Run on: 18/03/2021 16:35:32 | PC: LAPTOP-L1T2T565  
Test case name: Manual input - 5 nodes
```

```
Solved in: 0.004 s  
Number of reduction function calls: 17
```

The optimal tour is:

```
0 --> 2  
2 --> 3  
3 --> 1  
1 --> 4  
4 --> 0
```

Total cost is 34.0

```
=====
```

```
=====
```

Solver Summary:

```
=====
```

```
Run on: 18/03/2021 16:31:11 | PC: LAPTOP-L1T2T565  
Test case name: burma14
```

```
Solved in: 9.248 s  
Number of reduction function calls: 12985
```

The optimal tour is:

```
0 --> 9  
9 --> 8  
8 --> 10  
10 --> 7  
7 --> 12  
12 --> 6  
6 --> 11  
11 --> 5  
5 --> 4  
4 --> 3  
3 --> 2  
2 --> 13  
13 --> 1  
1 --> 0
```

Total cost is 3323.0

```
=====
```

# Branch and Bound

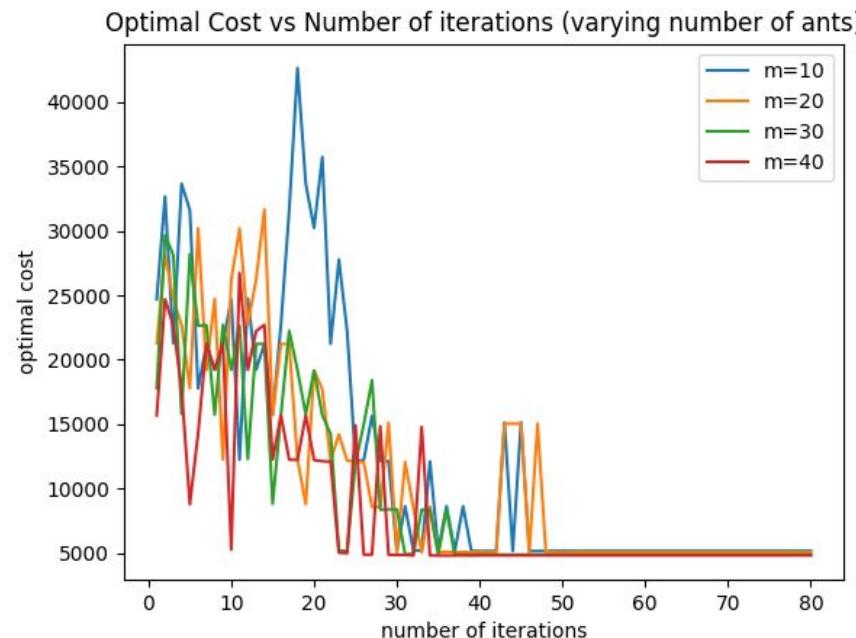
---

- Exponential time complexity  $O(n!)$
- On applying it to Problem 3 it gets transformed to Greedy algorithm. So its implementation is not possible here.
- A variant of problem 3 which reduces the penalty in correspondence to the popularity of the exhibit can be attempted.

# Ant-Colony Optimization

---

## Problem 1 - Symmetric Problem

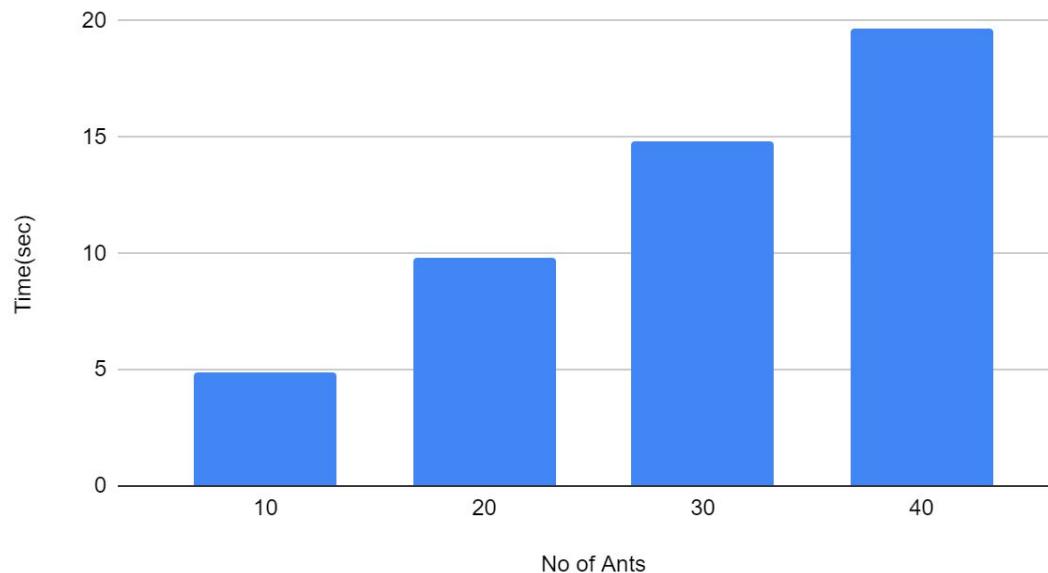


# Ant-Colony Optimization

---

## Problem 1- Symmetric Problem

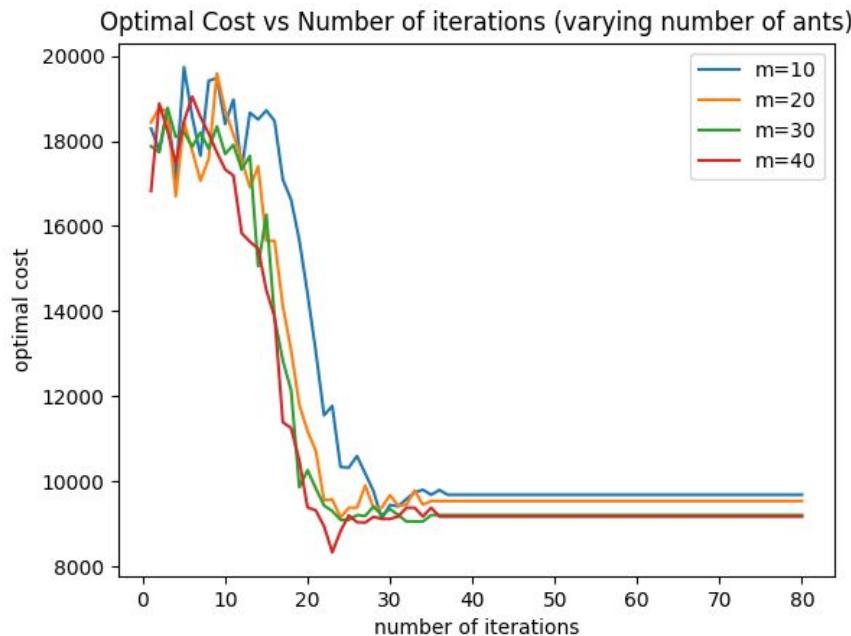
Running Time vs. No of Ants



# Ant-Colony Optimization

---

## Problem 2 - Asymmetric Problem

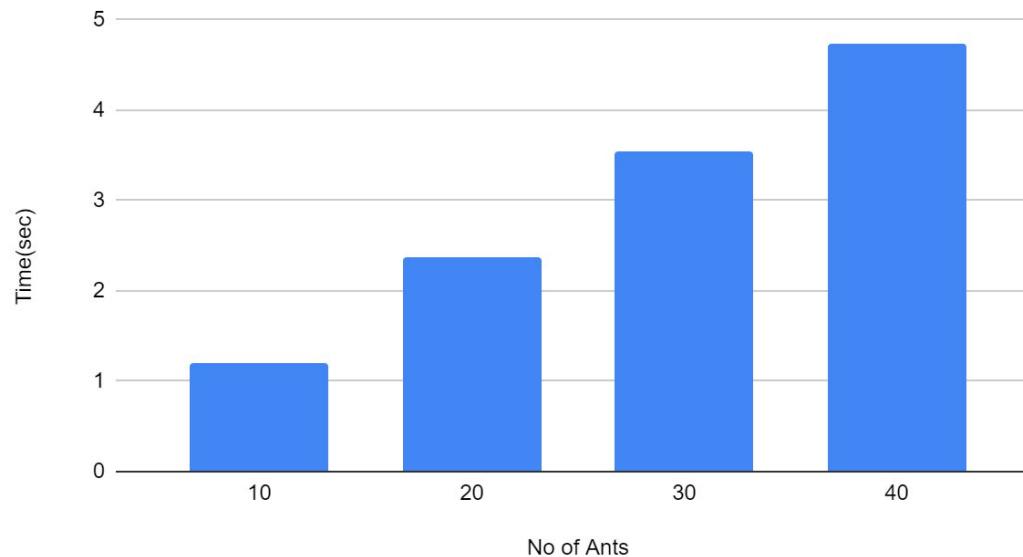


# Ant-Colony Optimization

---

Problem 2 - Asymmetric Problem

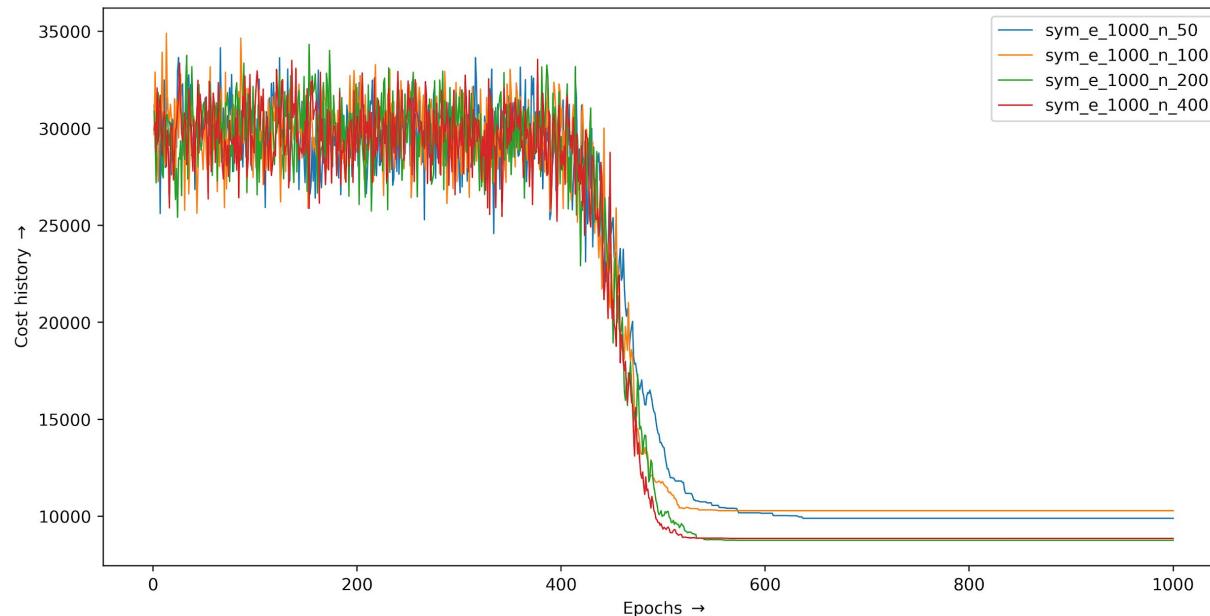
Running Time vs. No of Ants



# Simulated Annealing

---

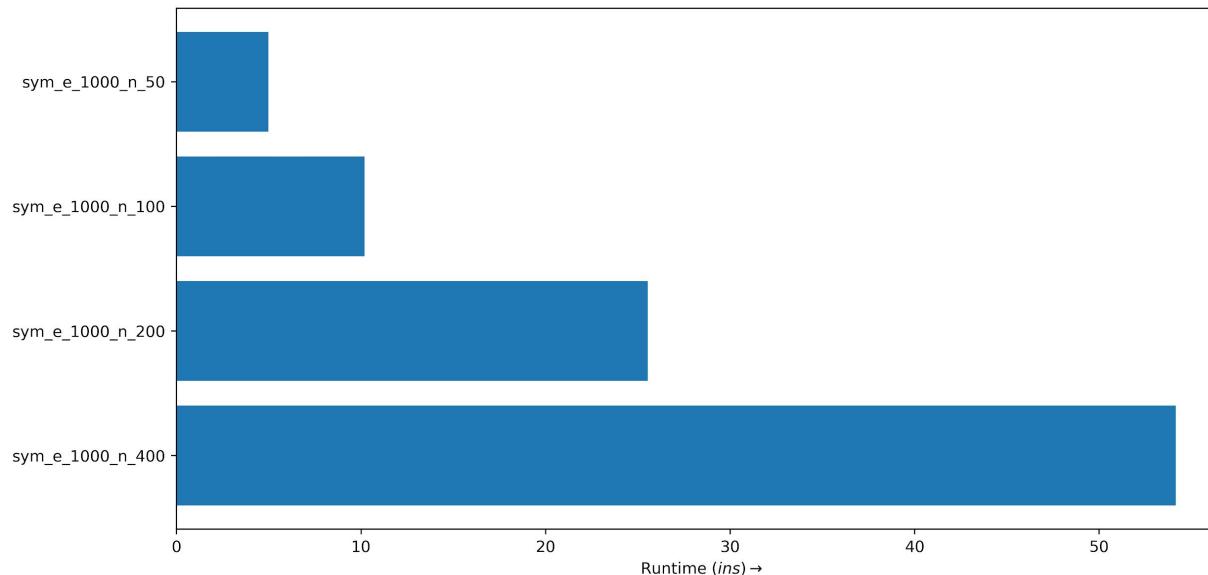
- Problem 1 - Symmetric TSP



# Simulated Annealing

---

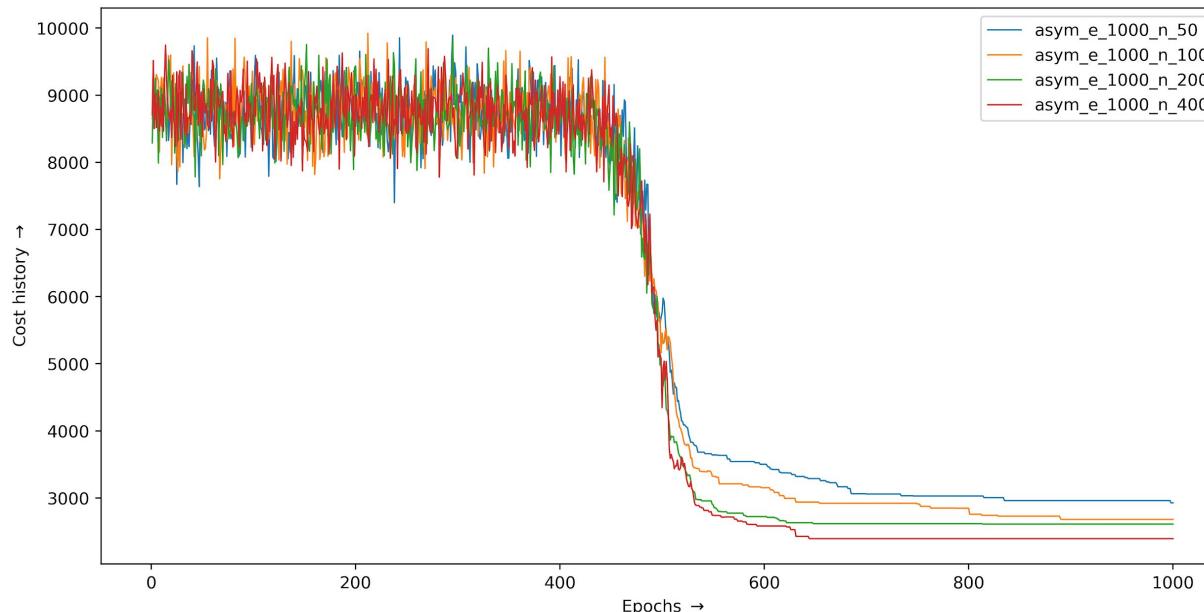
- Problem 1 - Symmetric TSP



# Simulated Annealing

---

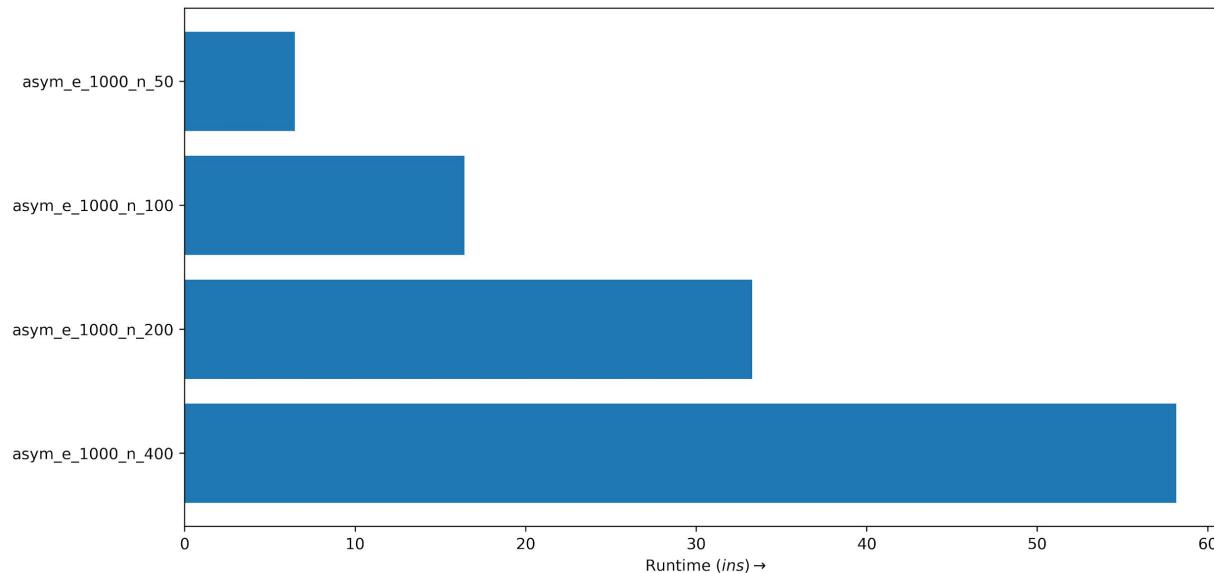
- Problem 2 - Asymmetric TSP



# Simulated Annealing

---

- Problem 2 - Asymmetric TSP



# Simulated Annealing

---

- Problem 3 - Satisfaction Index
  - Solution Representation

**Nodes being visited:** [1, 2, 3, 4, 5]

**Nodes not visited:** [9, 8, 7]

**Representation:** [1, 2, 3, 4, 5 | 9, 8, 7]

**Actual input:**

- Single Array
- Location of bar

`nodes=[1, 2, 3, 4, 5, 9, 8, 7]`

`loc_bar=5` (*= Number of elements in array of nodes being visited*)

# Simulated Annealing

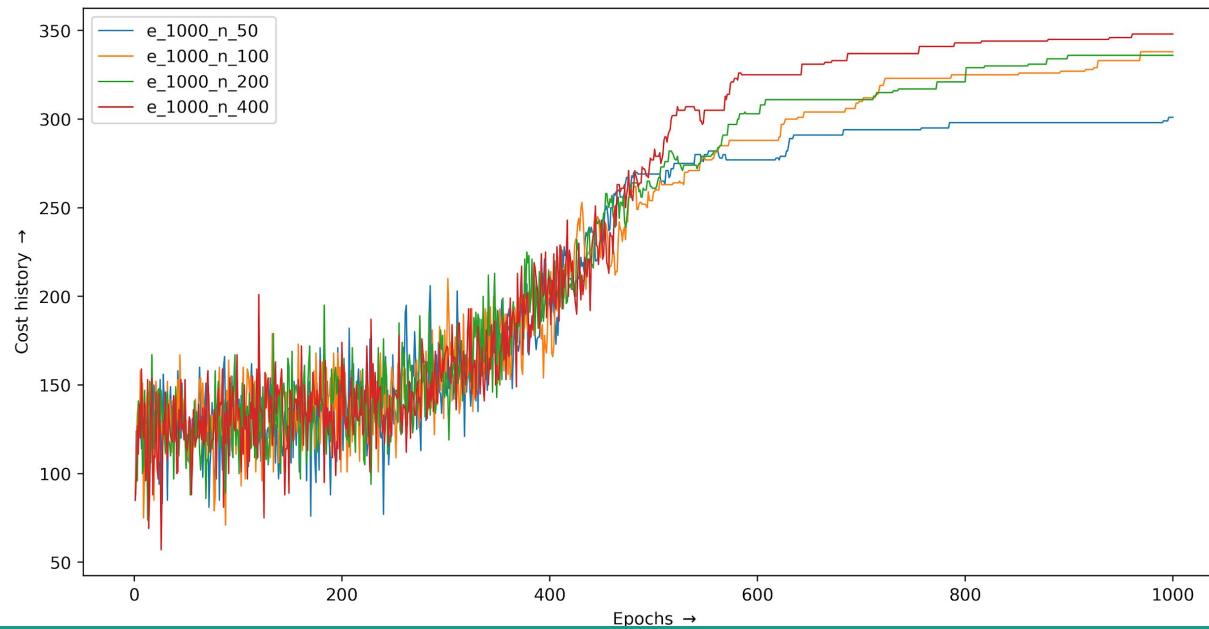
---

- Problem 3 - Satisfaction Index
  - Functions:
    - **ApplySwap()**: Swap 2 elements in the array of nodes being visited  
Eg: `nodes=[1, 2, 3, 4, 5 | 9, 8, 7] → nodes=[4, 2, 3, 1, 5 | 9, 8, 7]`
    - **ApplyConsecutiveSwap()**: Swap 2 consecutive elements in the array randomly (*To be added*)
    - **ApplyShake()**: Swap 1 element from the array of nodes being visited and 1 from the array of nodes not being visited  
Eg: `nodes=[1, 2, 3, 4, 5 | 9, 8, 7] → nodes=[1, 2, 7, 4, 5 | 9, 8, 3]`
    - **ModifyNodes()**: Change location of bar (Increase or decrease number of nodes being visited), i.e., (+1/-1 with probability k)  
Eg: `loc_bar=5 → loc_bar=6`

# Simulated Annealing

---

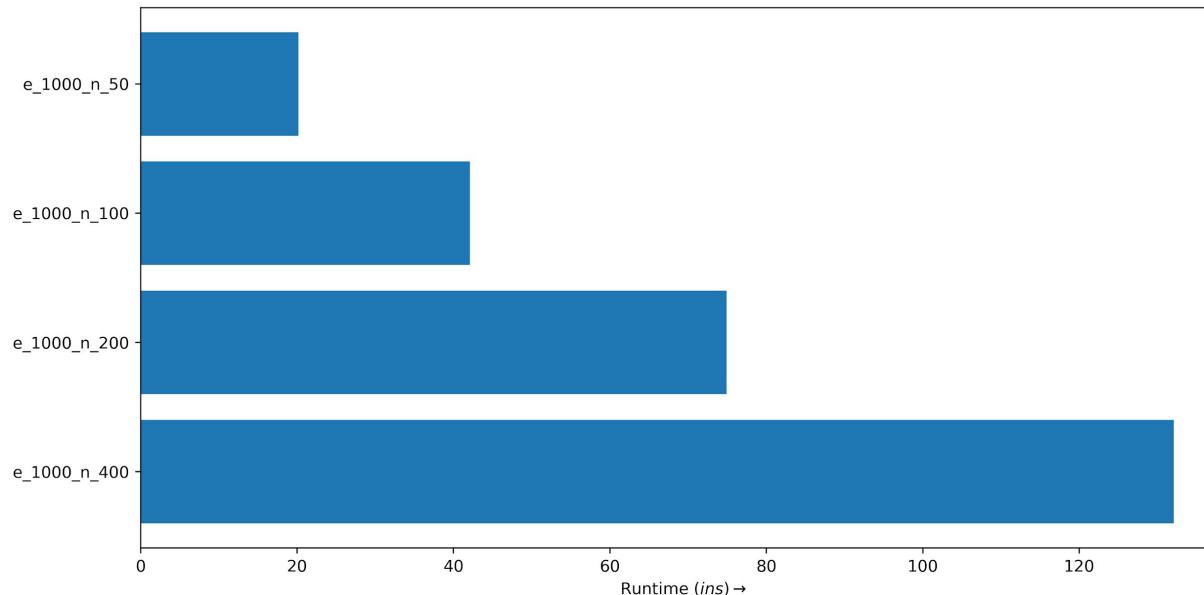
- Problem 3 - Satisfaction Index



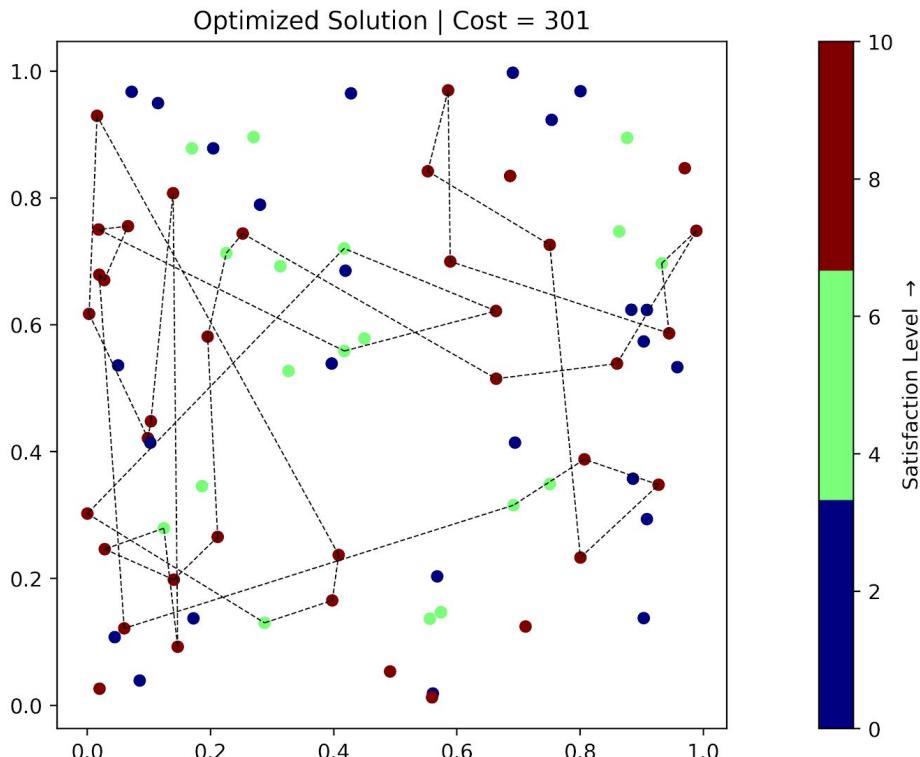
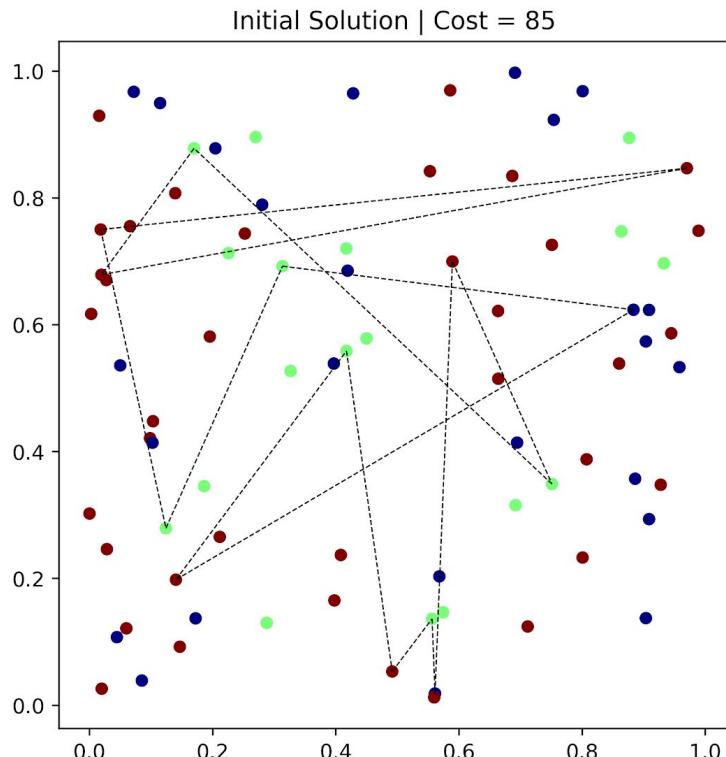
# Simulated Annealing

---

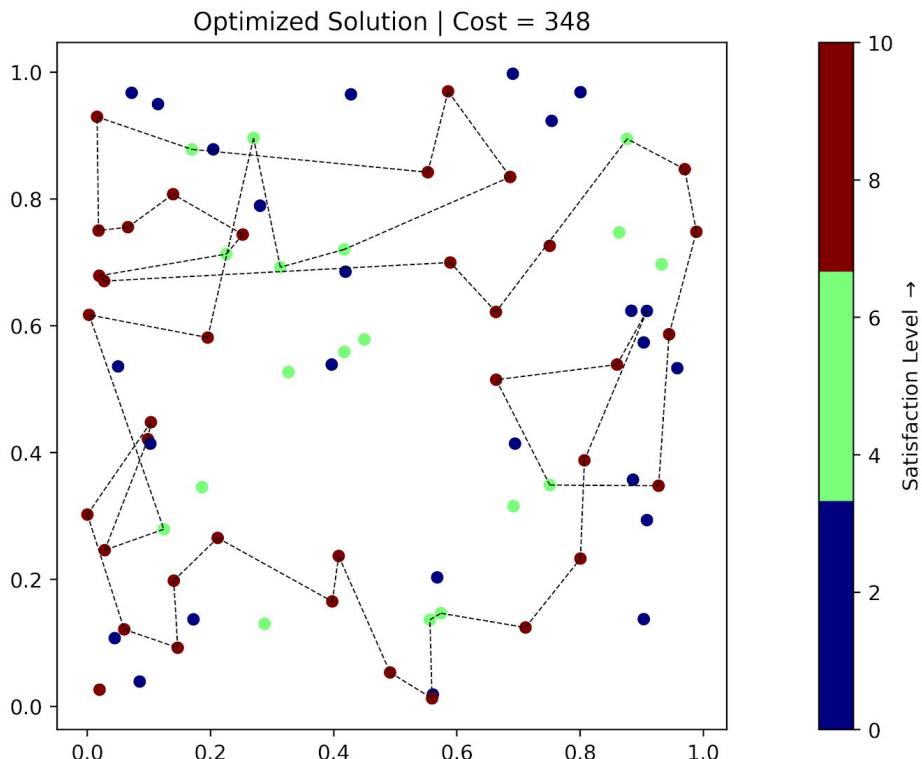
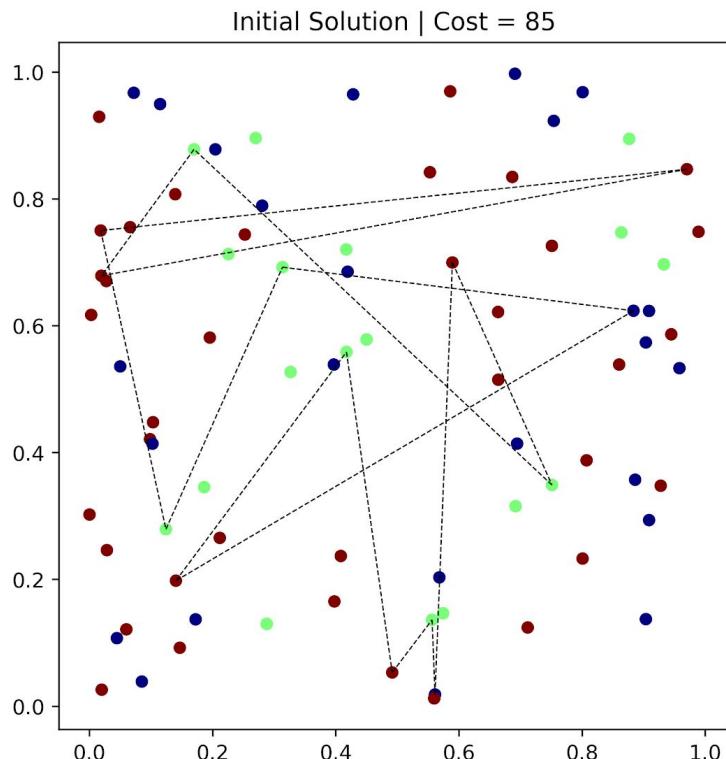
- Problem 3 - Satisfaction Index



Epoch = 1000 | Iterations/Epoch = 50 | Exhibits = 80



Epoch = 1000 | Iterations/Epoch = 400 | Exhibits = 80



# Performance Metrics

---

- Once the algorithms are implemented they can be compared using performance metrics
- The performance metrics to be used are Efficiency, Reliability and Quality of Solution
- Efficiency involves measuring number of fundamental evaluations, running time and memory consumption of the algorithms
- Reliability can be checked by considering aspects like success rate, number of constraint violations, if any, and effect of starting point choice.
- Quality of Solution is determined by measuring computational accuracy, time taken to find optimal value within a fixed range of the solution or accuracy of the solution after running for a certain number of iterations or a certain time.

# TSPLIB Test Cases

---

- TSPLIB is a library of sample instances for the TSP (and related problems) from various sources and of various types.
- TSPLIB has test case problems of Symmetric and Asymmetric TSP Problems, with varying number of nodes, which will be used for performance evaluation of the implemented algorithms
- The data provided in TSPLIB includes coordinates of nodes or costs between the nodes and the best found cost for each test case. For some test cases the optimal path is also given.

# Benchmark Cases - Symmetric

---

Test Case	Run Time (in sec)			
	Simulated Annealing	Ant Colony	Genetic	Branch and Bound
<i>burma14</i>	3.8	1.9		19.8
<i>att48</i>	7.4	5.4		
<i>berlin52</i>	7.6	6	-	$\rightarrow \infty$
<i>bier127</i>	14.9	15.2		
<i>brg180</i>	19.9	24.3		

# Benchmark Cases - Symmetric

---

Test Case	Final Cost				
	Simulated Annealing	Ant Colony	Genetic	Branch and Bound	Expected Optimum
<i>burma14</i>	3323	3323		3323	3323
<i>att48</i>	11992	12600			10628
<i>berlin52</i>	9465	8500	-	Hard Luck! 	7542
<i>bier127</i>	180727	141513			118282
<i>brg180</i>	4350	4690			1950

# Benchmark Cases - Asymmetric

---

Test Case	Run Time			
	Simulated Annealing	Ant Colony	Genetic	Branch and Bound
<i>br17</i>	4.0	1.9		
<i>ft53</i>	7.9	6.2		
<i>ftv64</i>	9.1	8.5	-	$\rightarrow \infty$ 
<i>ft70</i>	10.0	9.2		
<i>rbg403</i>	45.9	167.5		

# Benchmark Cases - Asymmetric

---

Test Case	Final Cost				
	Simulated Annealing	Ant Colony	Genetic	Branch and Bound	Expected Optimum
<i>br17</i>	39	89			39
<i>ft53</i>	8630	8969			6905
<i>ftv64</i>	2436	2233	-	Hard Luck! 	1839
<i>ft70</i>	44497	45384			1950
<i>rbg403</i>	3137	5179			2465

# Genetic Algorithm

---

- Updates
  - Code is completed
  - Yet to be tested with standard test cases

# Thank You!

---

---

# Appendix

---

# Outline

The Problem we face - a statement

Design Variables we understand

The simplest we begin

The complexities we include

The algorithms we employ

Taking the next step

---

# The Problem

The problem started with someone at the gate wondering:

“What if I could avoid crowds, see lots of exhibits and still get back on time?”

Bringing us to their rescue

# List of Symbols

---

$$y_{ij} = \begin{cases} 1, & \text{If path goes from exhibit } i \text{ to exhibit } j \\ 0, & \text{otherwise} \end{cases}$$

$c_{ij}$  := Penalty incurred for going from  $i$  to  $j$

$$c_{ij} > 0$$

$$G := (V, \mathbb{E})$$

$V$  := Vertices

$$\mathbb{E} := \{(x, y) | x, y \in V\}$$

$S$  := Set of all tours of  $G$

$E$  := Subset of permitted paths in  $\mathbb{E}$

$E'$  := Complement of  $E$

$v$  = Walking speed

$\tau$  = Time spent at each exhibit

$T_o$  = Total available time

---

# Pseudo Codes

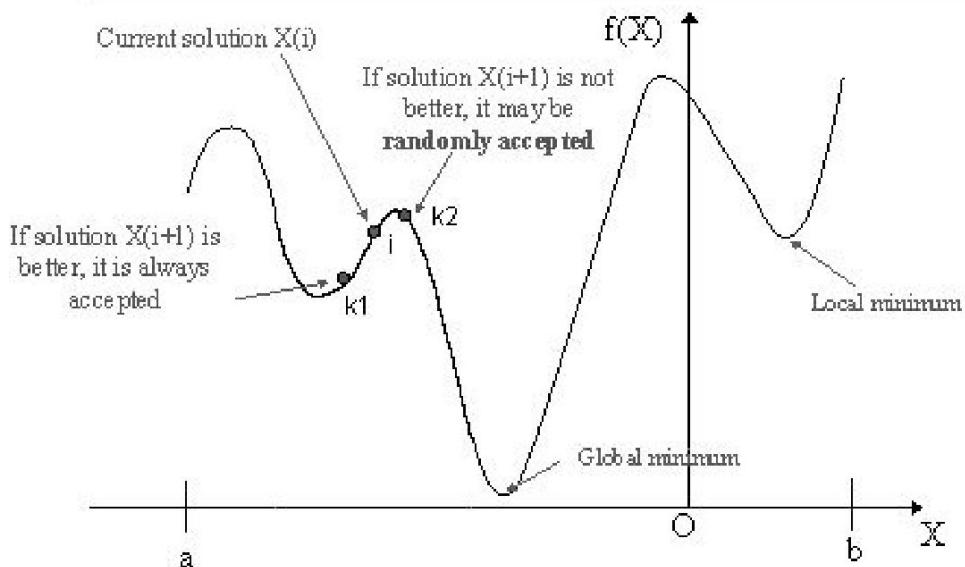
# Simulated Annealing

[Link](#) to Pseudo Code.

$$S_c = \{[1, 8, 2, 5], [9, 7, 10]\}$$

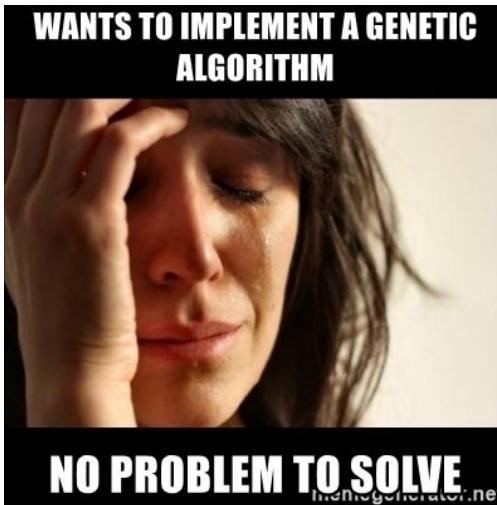
$$\text{QueueList} = [3, 4, 6]$$

## Principle of simulated annealing



# Genetic

---



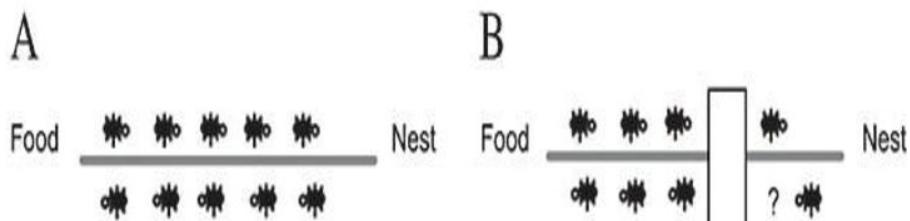
When you watch the first generation of your genetic algorithm

[Link](#) to Pseudo Code.

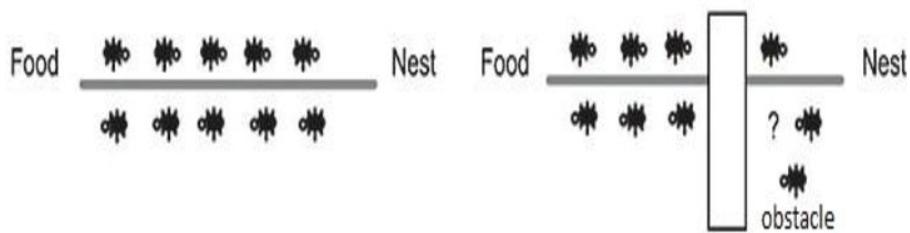
# Ant Colony



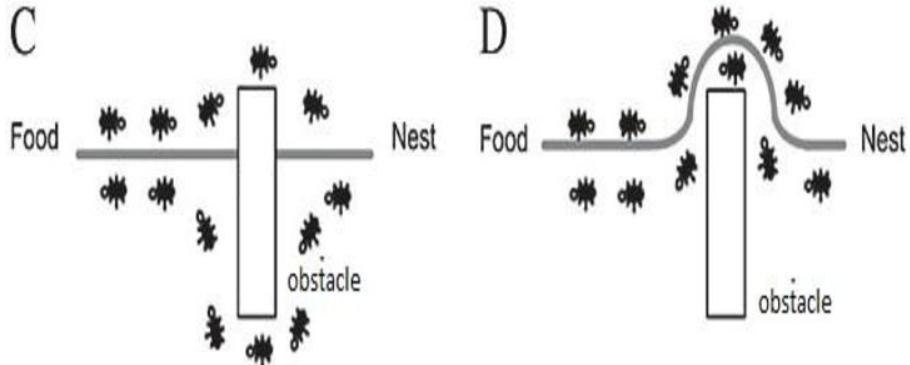
A



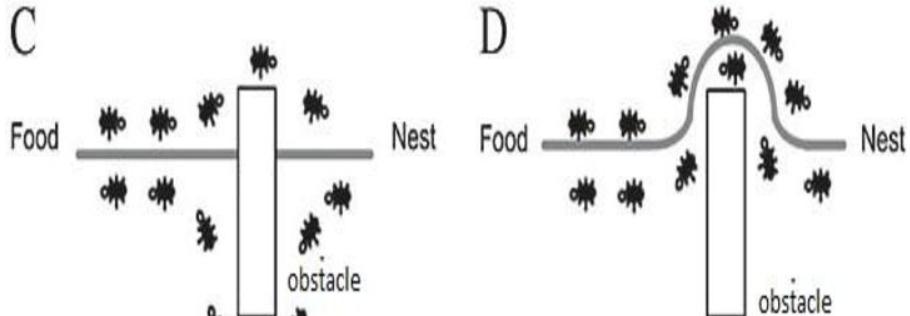
B



C



D



[Link](#) to Pseudo Code.

---

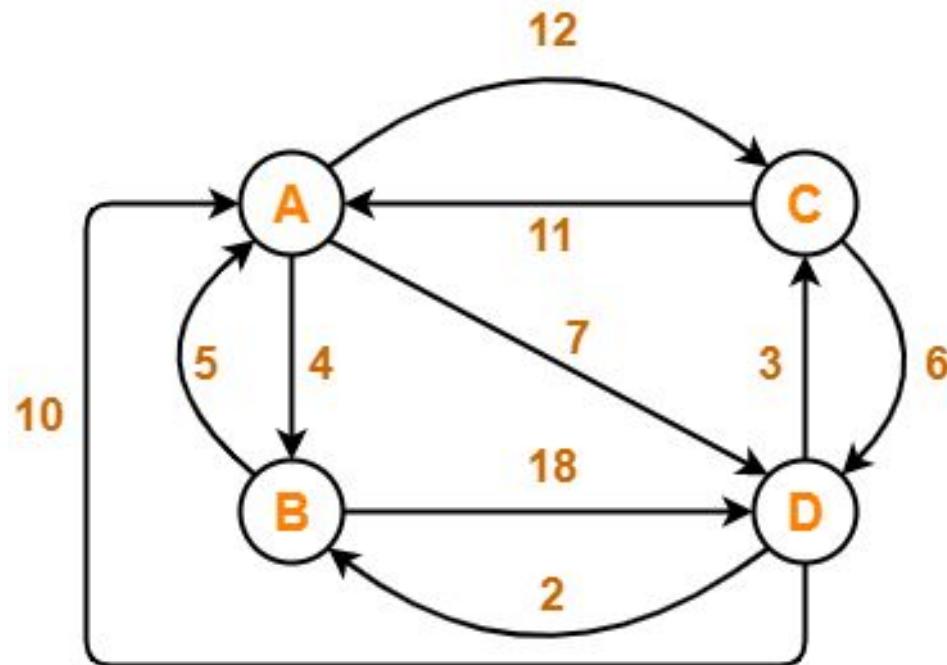
## Plan of Action - Analysis

- Solve each problem separately using all the selected algorithms to understand the nuances of the problem and search algorithms
- Use Branch-and-Bound algorithms to obtain the global optimal solution.
- Compare the results with available literature and TSP libraries
- Solve the complete problem statement with all of the complexities

# Branch-and-Bound

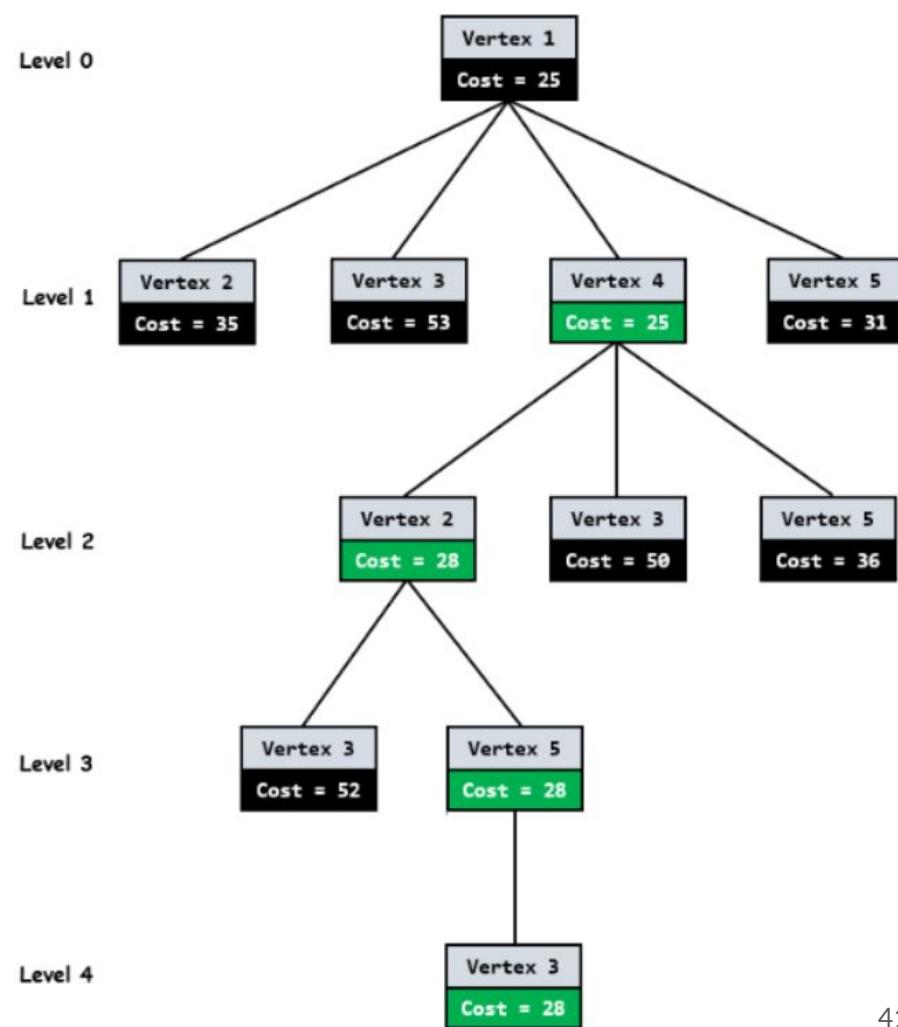
---

[Link](#) to Pseudo Code.



## Matrix Reduction

				<b>INF</b>	20	30	10	11
				15	<b>INF</b>	16	4	2
				3	5	<b>INF</b>	2	4
				19	6	18	<b>INF</b>	3
<b>INF</b>	10	20	0	1	16	4	7	16 <b>INF</b>
13	<b>INF</b>	14	2	0				
1	3	<b>INF</b>	0	2				
16	3	15	<b>INF</b>	0	<b>INF</b>	10	17	0
12	0	3	12	<b>INF</b>	12	<b>INF</b>	11	2
					0	3	<b>INF</b>	0
					15	3	12	<b>INF</b>
					11	0	0	12 <b>INF</b>



# Possible Applications

---

- Around the world tour while optimizing airline cost
- 



# References

---

Optimal walk around path in a museum to view all exhibits

- [Optimal Museum Traversal Using Graph Theory](#)
  - Explains basics of Hamiltonian path
- [Warehouse Optimization - Algorithms For Picking Path Optimization](#)
  - Gives a brief about all kinds of algorithms which can be employed for Path Optimization

Travelling Salesman Problem (TSP)

- [Travelling salesman problem - Wikipedia](#)
  - Explains the problem, formulations and constraints
  - Talks about the different algorithms as well
- [Travelling Salesman Problem | Set 1 \(Naive and Dynamic Programming\)](#)
  - There a lot of implementations of different algorithms for solving TSP in the Related Articles section
- [Chapter 10 The Traveling Salesman Problem](#)
  - Detailed paper talking about various solutions to TSP, and their analysis and their time complexity
- [Particle swarm optimization for traveling salesman problem](#)
- [Traveling Salesman Problem: an Overview of Applications, Formulations, and Solution Approaches](#)

# References

---

## Problem Formulation

- Travelling Salesman Formulation
  - [Traveling salesman problems](#)
- School Bus Problem - This paper has a lot of variants
  - <https://dspace.mit.edu/bitstream/handle/1721.1/5363/OR-078-78.pdf?sequence=1&isAllowed=y>
- An Exact Algorithm for the Time-Constrained Traveling Salesman Problem
  - <https://scihub.wikicn.top/10.1287/opre.31.5.938>

## Algorithms

- Dynamic Programming, Simulated Annealing, and 2-opt:
  - [How to Solve Traveling Salesman Problem — A Comparative Analysis](#)
- Genetic Algorithm, Simulated Annealing, Particle Swarm Optimization, Ant Colony Optimization, Bacteria Foraging Optimization, and Bee Colony Optimization
  - [\(PDF\) Optimization Techniques for Solving Travelling Salesman Problem](#)

# References

---

## Genetic Algorithm

- [A Genetic Algorithm for Solving Travelling Salesman Problem](#)
- [A genetic algorithm for the orienteering problem - IEEE Conference Publication](#)
- [A genetic algorithm to design touristic routes in a bike sharing system | Request PDF](#)

## Simulated Annealing

- [A simulated annealing heuristic for the team orienteering problem with time windows](#)
- [A simulated annealing heuristic for the multiconstraint team orienteering problem with multiple time windows](#)
- [Solving tourist trip planning problem via a simulated annealing algorithm](#)

## Ant Colony Algorithm

- [\(PDF\) An ant colony approach to the orienteering problem](#)
- [Ant colony approach to the orienteering problem](#)

## Branch-and-Bound

- [A Proposed Solution to Travelling Salesman Problem using Branch and Bound](#)
- [Traveling Salesman Problem: an Overview of Applications, Formulations, and Solution Approaches](#)
- [Branch and Bound for TSP](#)