Traveling Salesman Problem via Swarm Intelligence

Pei-Chen Yen and Frederick Kin Hing Phoa

Institute of Statistical Science, Academia Sinica, Taiwan

Presenter: Pei-Chen Yen

July 18 2021

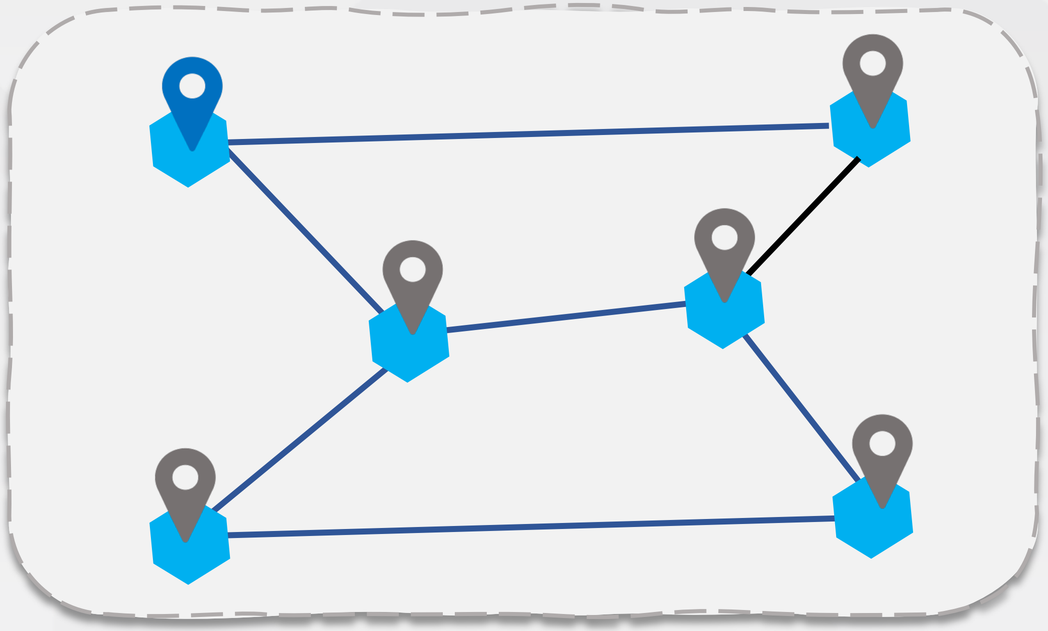
The Twelfth International Conference on Swarm Intelligence

Outline

- Introduction
- Motivation
- SIB Method
- Implementation
- Summary

Introduction

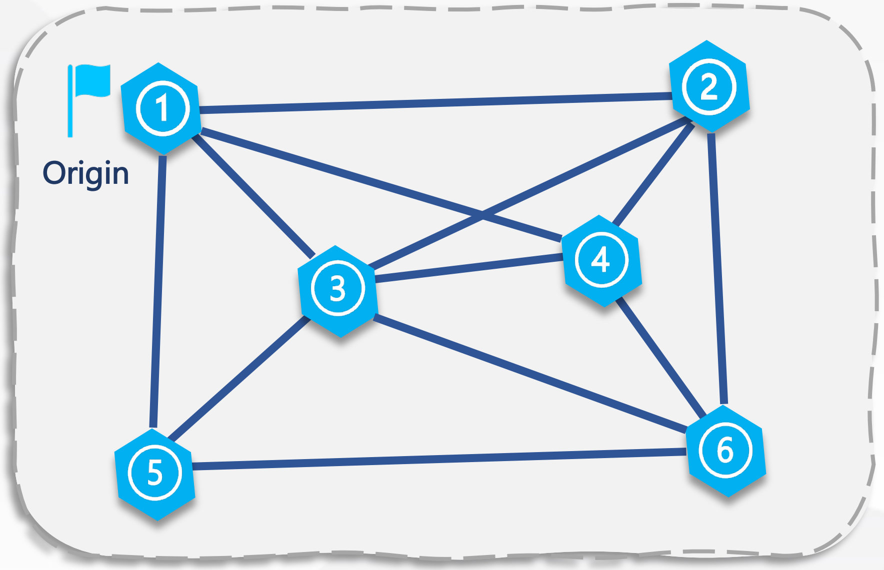
Traveling Salesman Problem



Given:
- N cities
- A distance matrix

Goals:
- Find a route with the shortest distance
- Pass all cities and return to the original place

Traveling Salesman Problem



What is the shortest route that visits every city and returns to the origin station?

- To find a route with the shortest distance…

1 4 6 2 3 5 1 = D1

1 2 6 4 3 5 1 = D2

⋮

- There are (N-1)! possible routes

# Cities	# tours
5	24
10	362,880
15	87,178,291,200
20	121,645,100,408,832,000

- A NP-Hard problem!

Traveling Salesman Problem

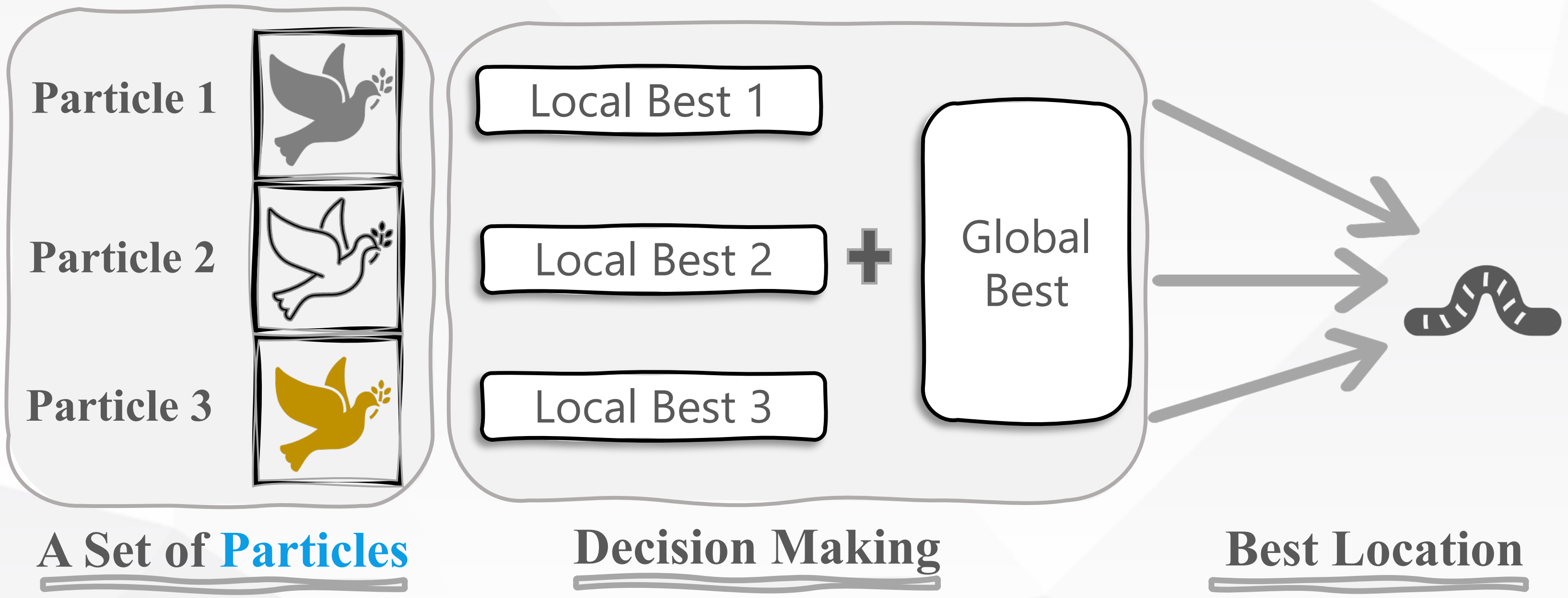
Introduction

Nature-Inspired Metaheuristic Methods

- ✔ Particle Swarm Optimization (PSO)
- ✔ Ant Colony Optimization (ACO)
- ✔ Artificial Bee Colony (ABC)
- ✔ Bat Algorithm (BA)
- ✔ Cuckoo Search (CS)

Particle Swarm Optimization (PSO)

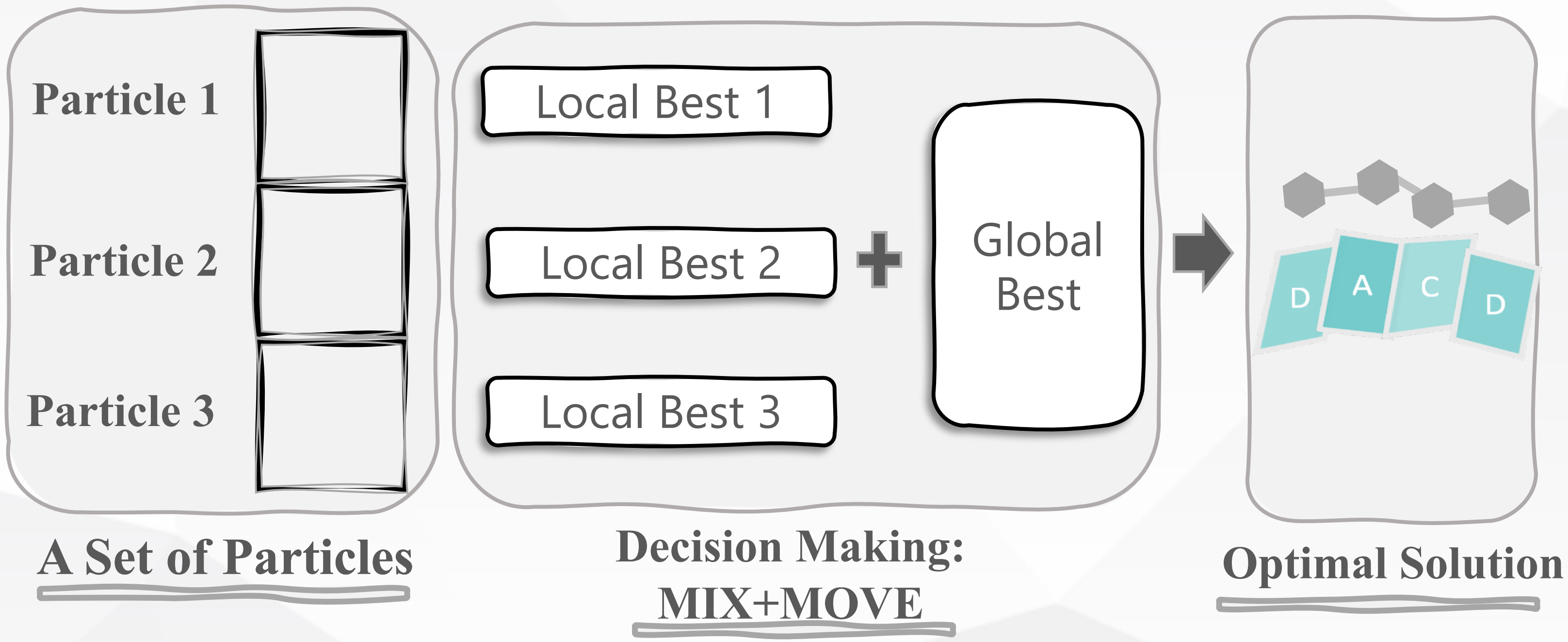
- Mimicking the foraging behaviors of bird flocks to find the global best position



- Each particle modifies its position based on the Local Best and Global Best information
- For problems with continuous domain

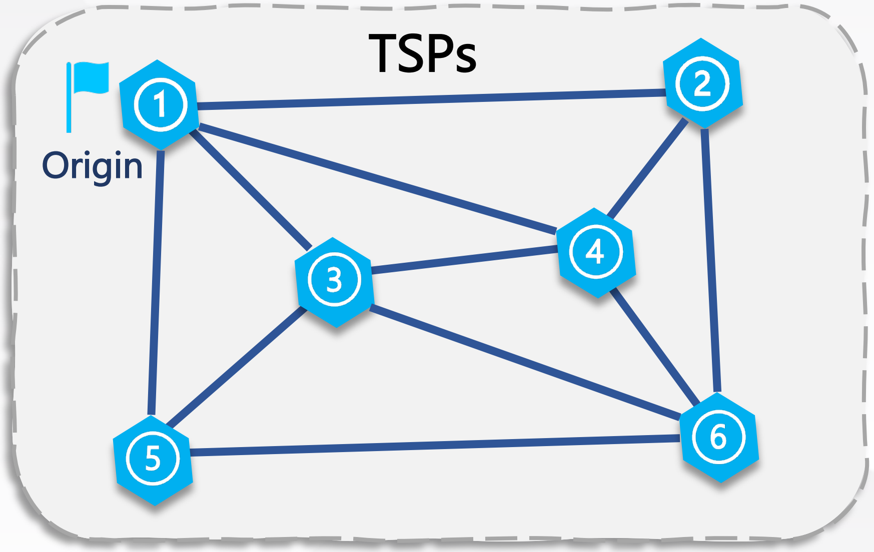
PSO Style Method-SIB Method

- For problems with discrete domain:



- Small number of tuning parameters
- Memory Saving

The SIB Method for Solving the TSPs



- Graph G=(V, E), V=nodes, E=edges.
- V = {1,2,...,N}: N cities.
- Each edge (i, j) ∈ E is of distance $d_{i,j}$.

- A graph **G=(V, E)** , where V is a group of **nodes** and E is a group of **edges**.
- Each **node** represents a target station and each **edge** represents a path between stations if the path exists.
  - Each edge (i, j) ∈ E is assigned a distance $d_{i,j}$
  - Form a Distance Matrix
- Find the best route with the shortest distance

$\Downarrow$

Find an **order** of {1,2,...,N} such that its total distance $\sum d_{i,j}$ is the minimum.

The SIB Method

- SIB Method – Swarm Intelligence Based Method (Phoa, 2017)
  - ✔ A PSO-style method to solve optimization with discrete domains
- **Basic Framework**

Definition

① Initialization Step

- ✔ Enter the parameters: N_loop, qLB, qGB
- ✔ **Generate particles**
- ✔ Determine the distance matrix

② Iteration Step: Information Update

- ✔ Run MIX operation ➡ MOVE operation
- ✔ Record Local Best(LB), Global Best(GB)
- ✔ Update Particles

③ Get the Optimal Value

1. Initialization Step (1/2)

SIB Method

- Number of Particles (M)
  - ✔ 100-300 for city size <=40
- Distance Matrix

$$\begin{bmatrix} \times & d_{1,2} & d_{1,3} & d_{1,4} \\ d_{2,1} & \times & d_{2,3} & d_{2,3} \end{bmatrix}$$

- Number of iterations (N_loop)
  - ✔ 100-300 for city size <=40
- Number of discrete units being exchanged with LB and GB particles (qLB and qGB)

1. Initialization Step (2/2)

Distance Matrix | M Particles | N_loop | qLB, qGB

- Step 1: Randomly generate M particles
- Step 2: Calculate objective function value for each particle
- Step 3: Get the initial **LB** and **GB**

SIB Method

Form A Set of Particles

Unit=City number

								Objective Function Value $\sum d_{i,j}$	
Particle 1 (Route 1)	9	8	7	…	21	15	9	D1	(LB 1)
Particle 2 (Route 2)	16	7	5	…	11	2	16	D2	(LB 2)
⋮								⋮	
Particle M (Route M)	17	20	3	…	4	6	17	DM	(LB M)

Number of Stations=N

GB

2. Iteration Step- MIX Operation(1/4)

Initial Particles | LB and GB | N_loop | qLB, qGB

SIB Method

- **Step 1**: For each unit, sum up the two distances between itself to its adjacent neighbors, and find the unit with the **maximum value(X_MAX)** and **its location**

X_MAX

Particle X

9	8	7	...	21	15	9	LB

GB

0 $d_{9,8}$ $d_{8,7}$ $d_{7,...}$ $d_{21,15}$ $d_{15,9}$ 0

$0 + d_{9,8}$	$d_{8,7} + d_{7,...}$	MAX & location=NO.2	$d_{15,9} + 0$

- **Step 2**: Refer to its Local Best array, find the **corresponding location** that is the same with the maximum value in X, and record the value in this location(X_LB)

$0 + d_{9,8}$	$d_{8,7} + d_{7,...}$	MAX	...	$d_{15,9} + 0$

LB Array of X

9	21	1	...	20	15	9

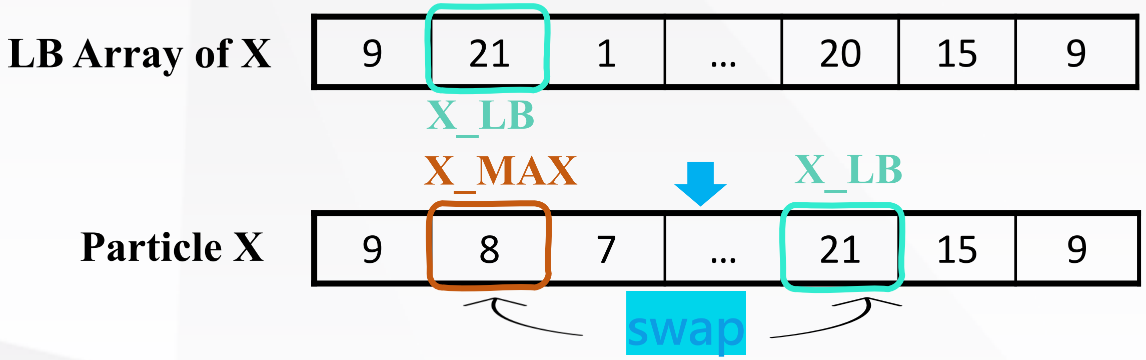
Location:N0.2=> X_LB

2. Iteration Step- MIX Operation(2/4)

X_MAX in X | X_LB in LB | N_loop | qLB, qGB

SIB Method

- Step 3: for X_LB found in LB array in step 2, go back to particle X and find the X_LB value in X



- Step 4: SWAP the X_MAX and X_LB
- Step 5: Do Step 1 to Step 4 qLB times (swap qLB times) for this X particle ➡ mixw LB
- Apply the same procedure for GB (using qGB) ➡ mixw GB

2. Iteration Step- MOVE Operation(3/4)

mixw LB | mixw GB | X(original) | N_loop

- Decision Making Procedure for this X particle
- Step 1: Calculate the Objective Function Value $\sum d\_(i,j)$ for each candidate.
- Step 2: Choose the one with the minimum $\sum d\_(i,j)$ and replace X with it.
  - ✔ If original X wins, then randomly choose two units and swap them ➡ do it q_GB times

X(original)
mixw LB ➡ X(new)
mixw GB

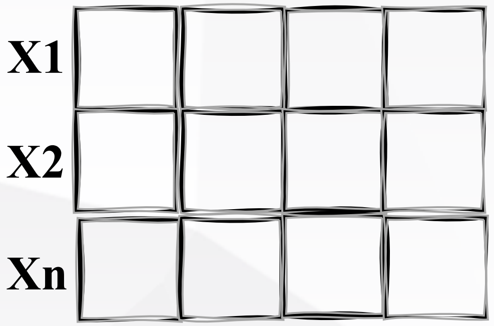
SIB Method

2. Iteration Step- Update(4/4)

X(new) | N_loop | LB(original) | GB(original)

SIB Method

- Apply MIX and MOVE to all particles
- Get a new set of particles
- Update LB and GB again



LB1(new)

LB2(new) → GB1(new)

LBn(new)

- After getting the new GB for this iteration, apply the same procedure to the remaining of N_loops! We will get the Optimal GB.

Optimal Final GB →

Implementation

Dataset

- ✔ Delivery data provided by a logistic company
- ✔ Station Size: 10 to 30

Data Input

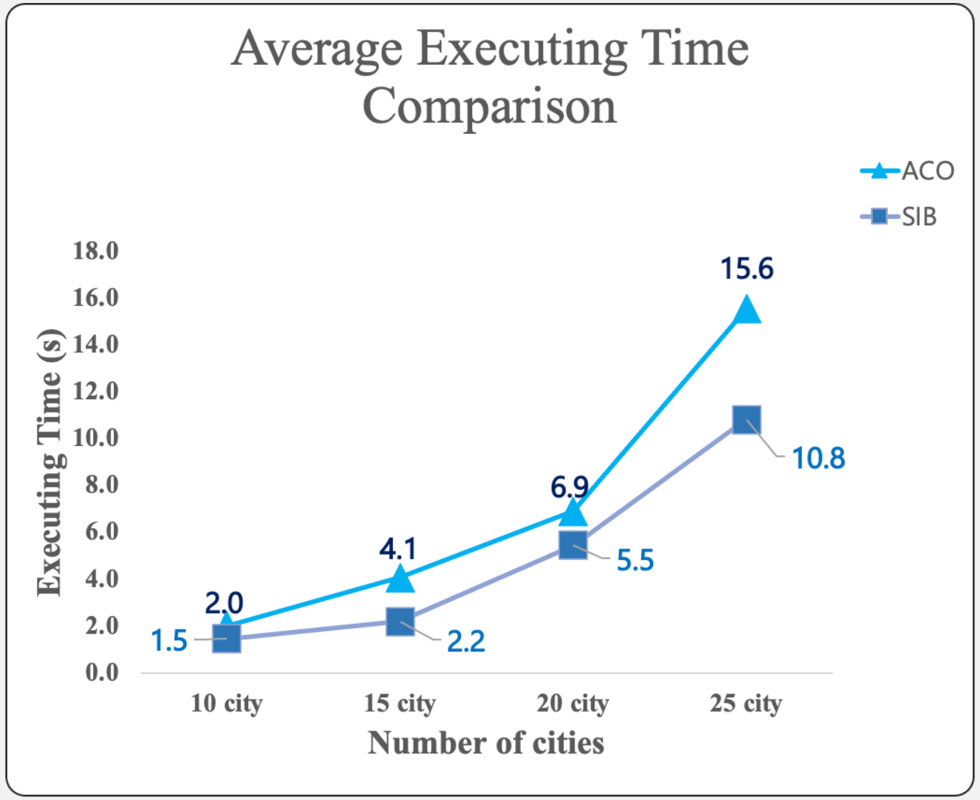
- ✔ Station Name
- ✔ Station Location

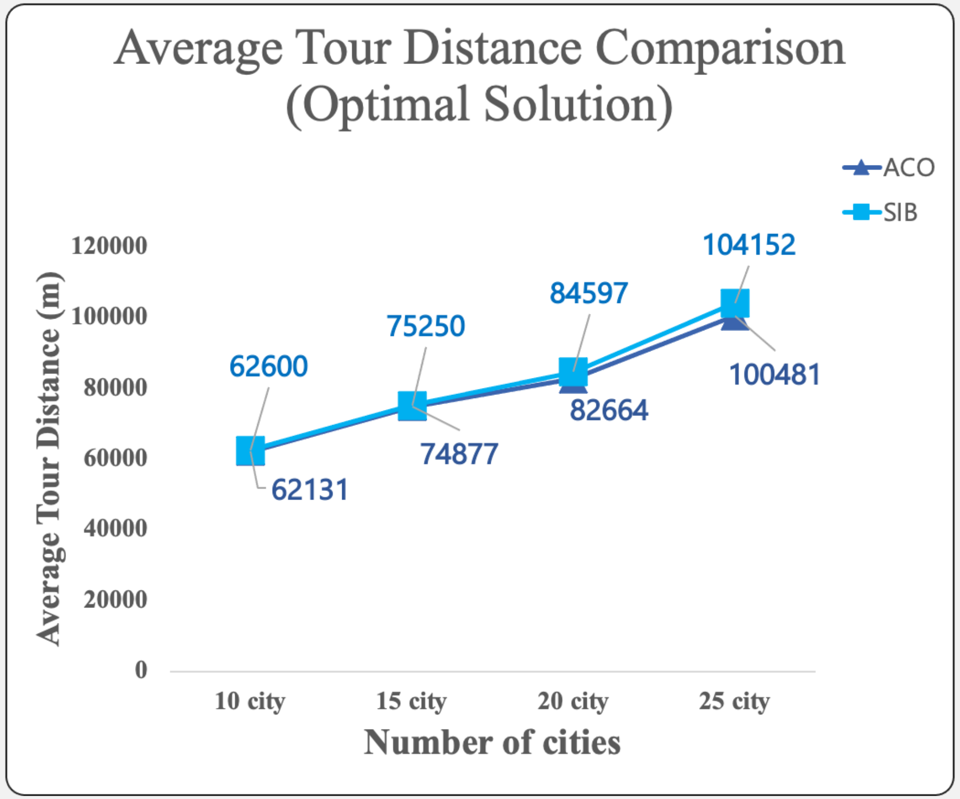
Iteration Parameters

Number of Cities	ACO*	SIB
10 cities	ant=30 N_loop=200	particle=100 n_loop =200
15 cities		
20 cities		particle=200, n_loop =200
25 cities	ant=30 N_loop=300	particle=200, n_loop =300

*alpha = 1, beta = 2, rho = 0.1, Q = 1 for all test

Implementation





- The SIB method performs significantly efficient in terms of executing time.
- The SIB method obtains a slightly longer distance then the ACO.

Overall, the ACO can find a better optimal solution, but it takes longer time and more memories. On the other hand, the SIB can obtain close solution with shorter execution time and less requirements.

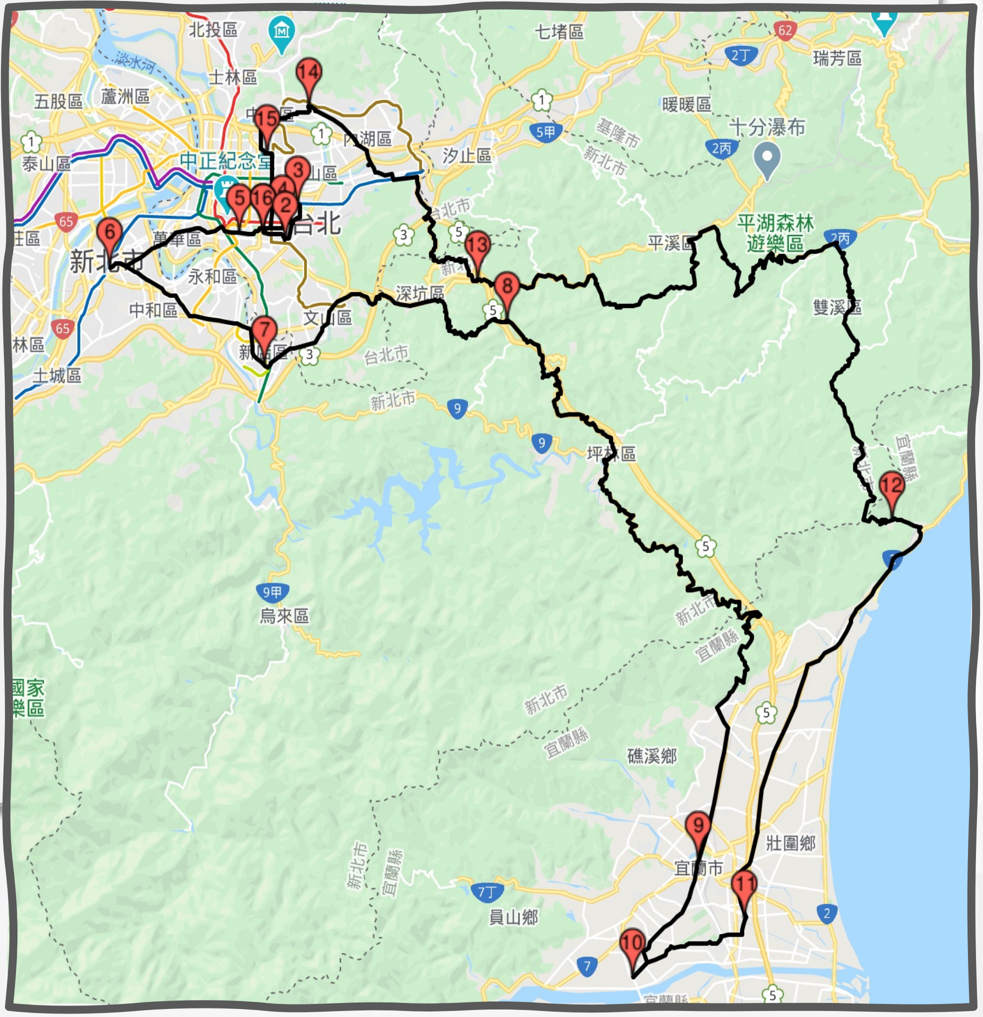
Conclusion

Good performance for TSP

- ✔ Practical in use for Real Logistic Problems

Aim for Practical Problems

- ✔ Efficiency
- ✔ Use less parameters
- ✔ Common city size in 10 to 25 for real-world application



THANKS!

Lorna Pei-Chen Yen

- Research Assistant in Frederick Phoa's Research Group
- Institute of Statistical Science, Academia Sinica, Taiwan
- lornayen@webmail.stat.sinica.edu.tw