2024 IEEE Conference on Artificial Intelligence

VICTORIA UNIVERSITY OF WELLINGTON
TE HERENGA WAKA
Evolutionary Computation Research Group

Automated Design of State Transition Rules in Ant Colony Optimization by Genetic Programming: A Comprehensive Investigation

Bocheng Lin, Yi Mei*, Mengjie Zhang

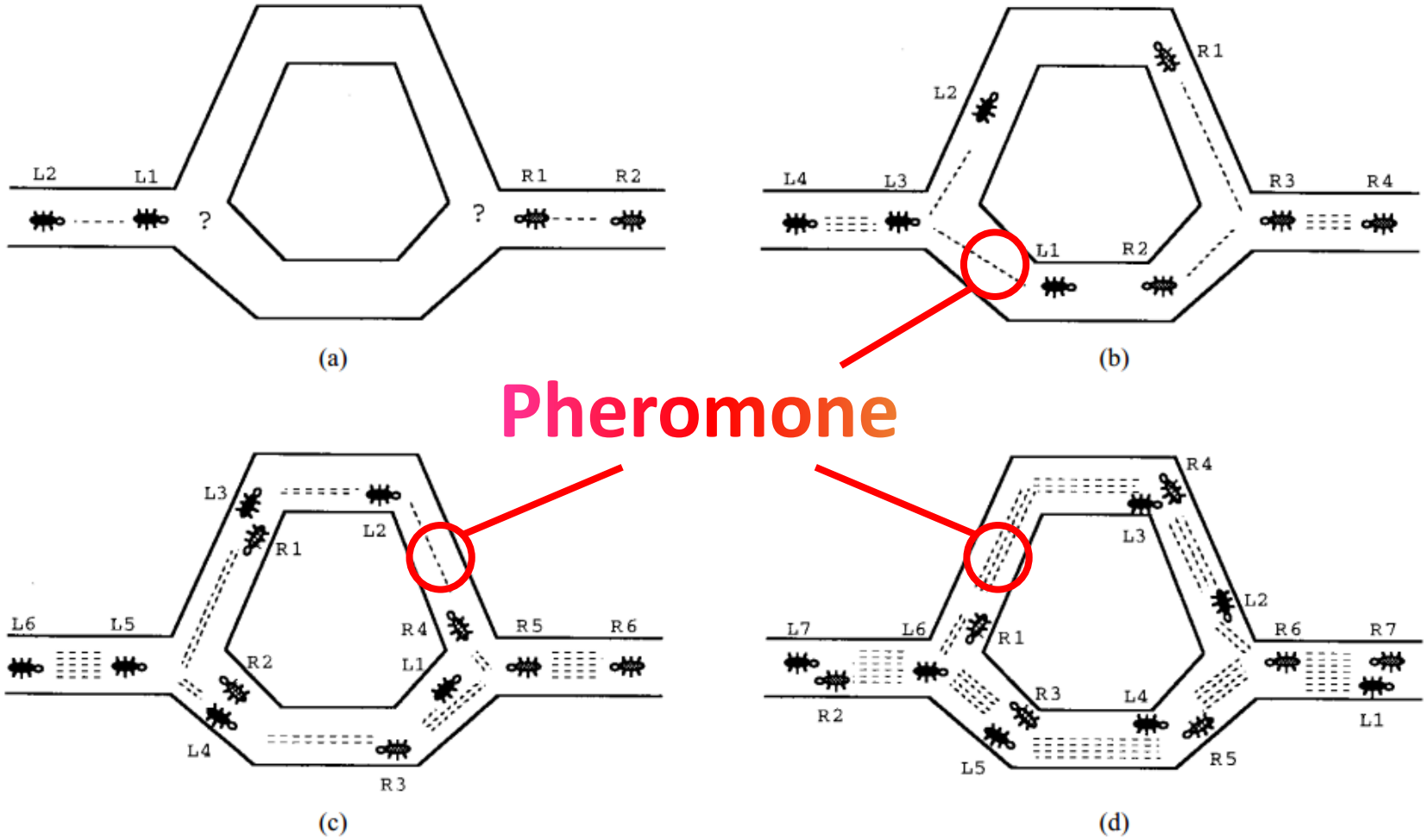
2024.6.21

2024 IEEE Conference on Artificial Intelligence (IEEE CAI 2024) Workshop
Advances in Computational Intelligence and Machine Learning: Applications and Innovations
Special Issue of Memetic Computing Journal

PART I

What is ACO?

This is how ants find shorter path.

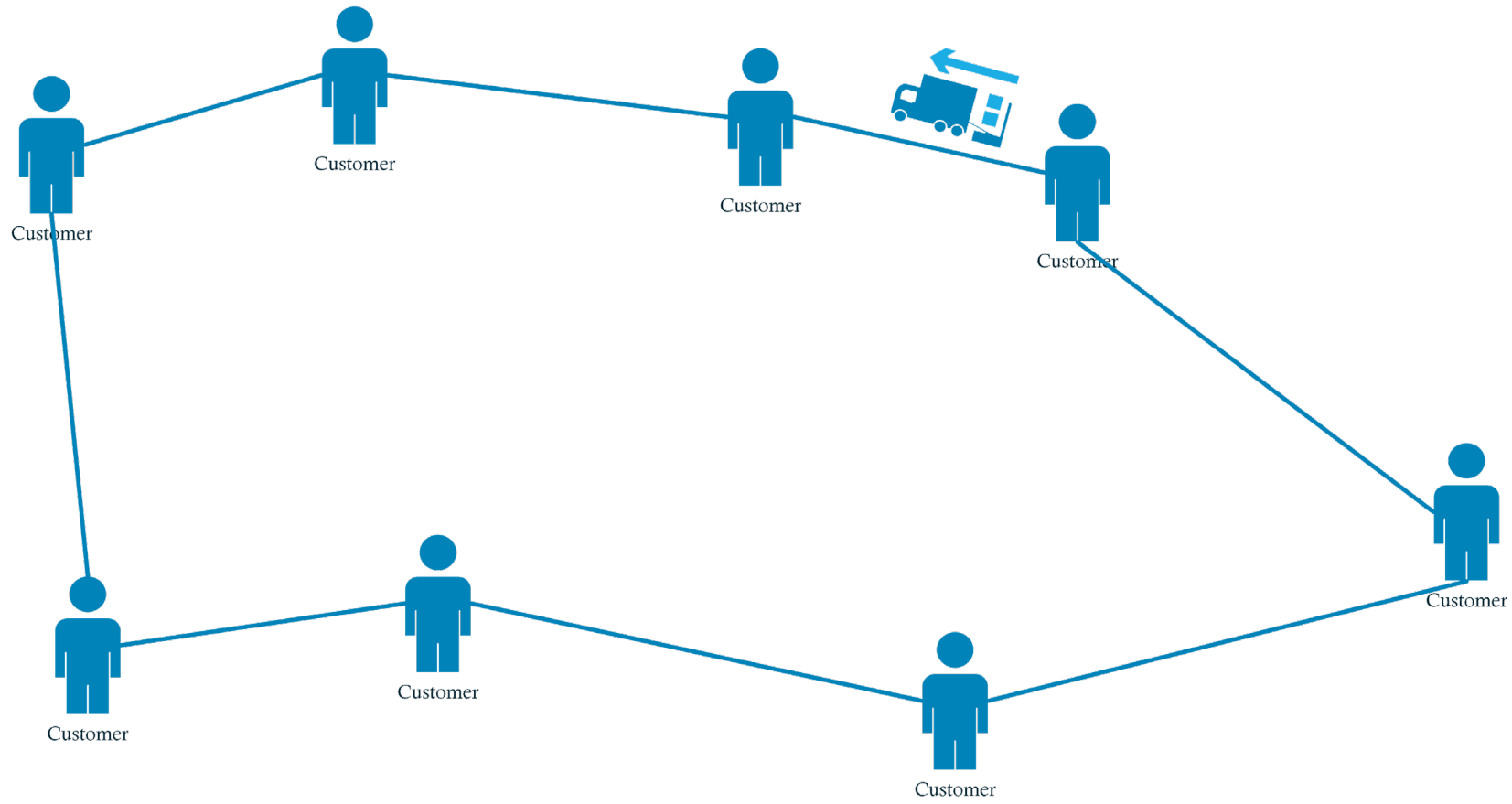


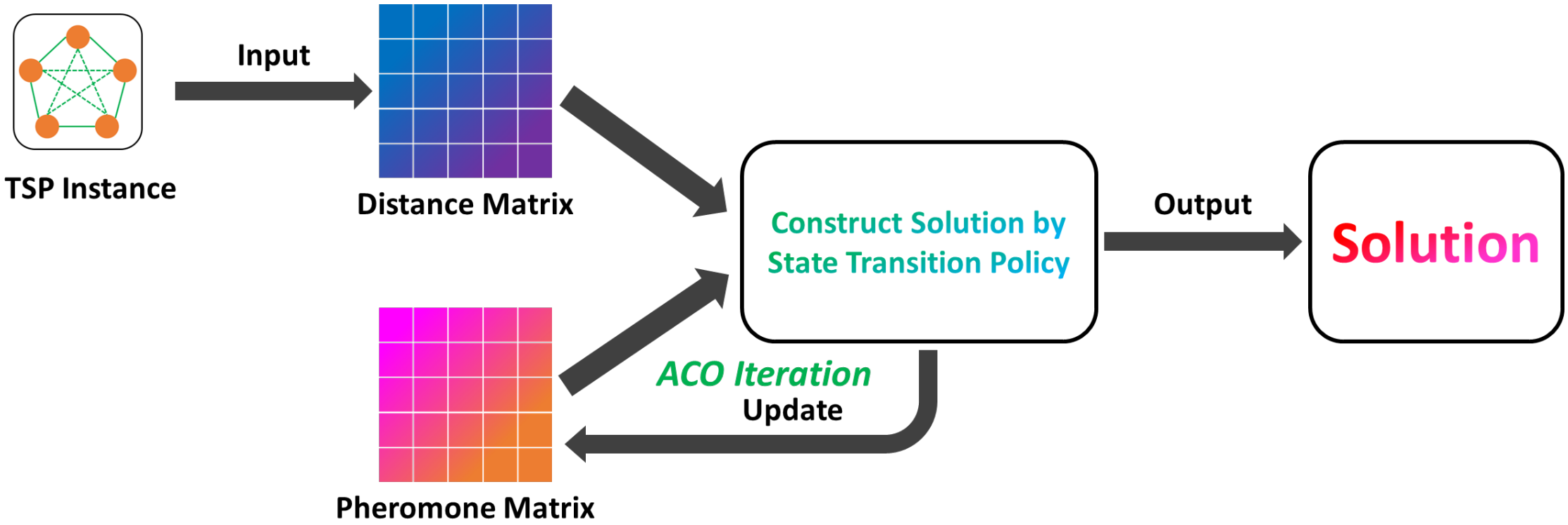
[1] M. Dorigo and L. M. Gambardella, “Ant colony system: a cooperative learning approach to the traveling salesman problem,” IEEE Transactions on Evolutionary Computation, vol. 1, no. 1, pp. 53–66, Apr. 1997, doi: 10.1109/4235.585892.

How ants find shortest path on more complex graph?

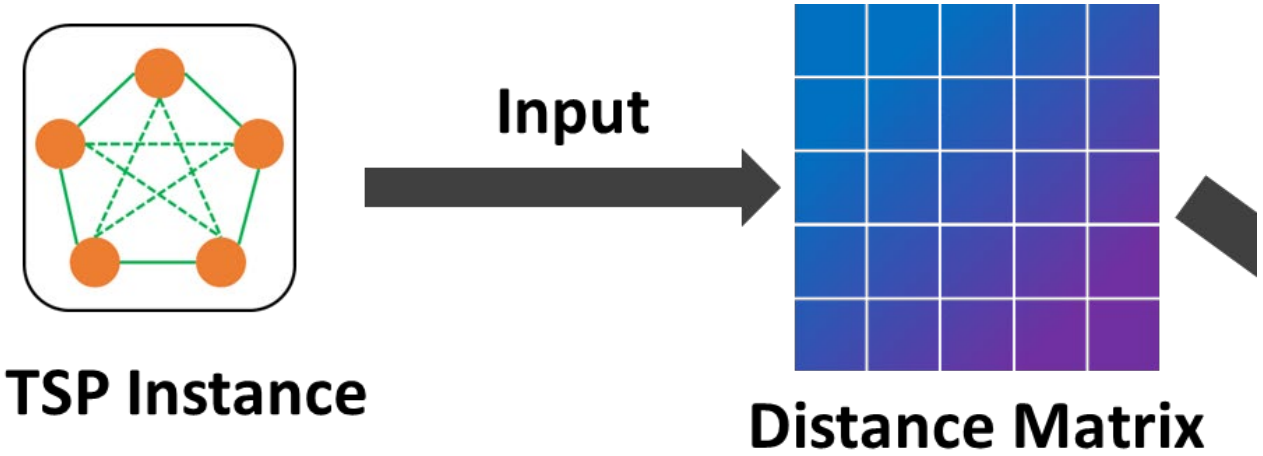
Travelling Salesman Problem

TSP

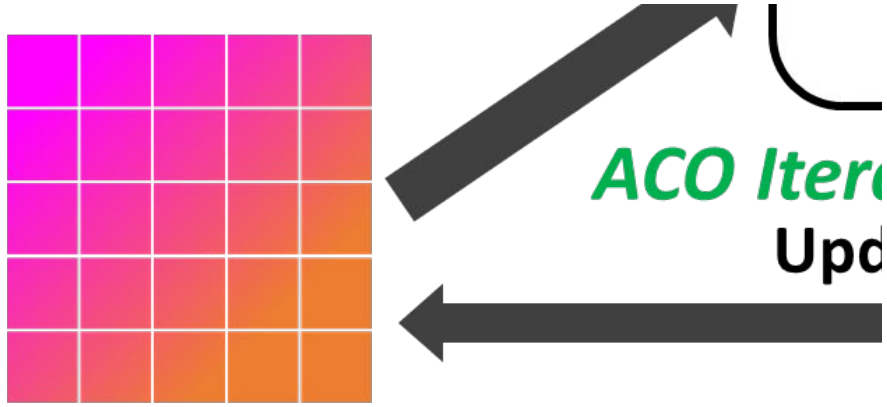



Input
TSP Instance
Distance Matrix
Construct Solution by State Transition Policy
Output
Solution
ACO Iteration
Update
Pheromone Matrix

A graph contains n nodes

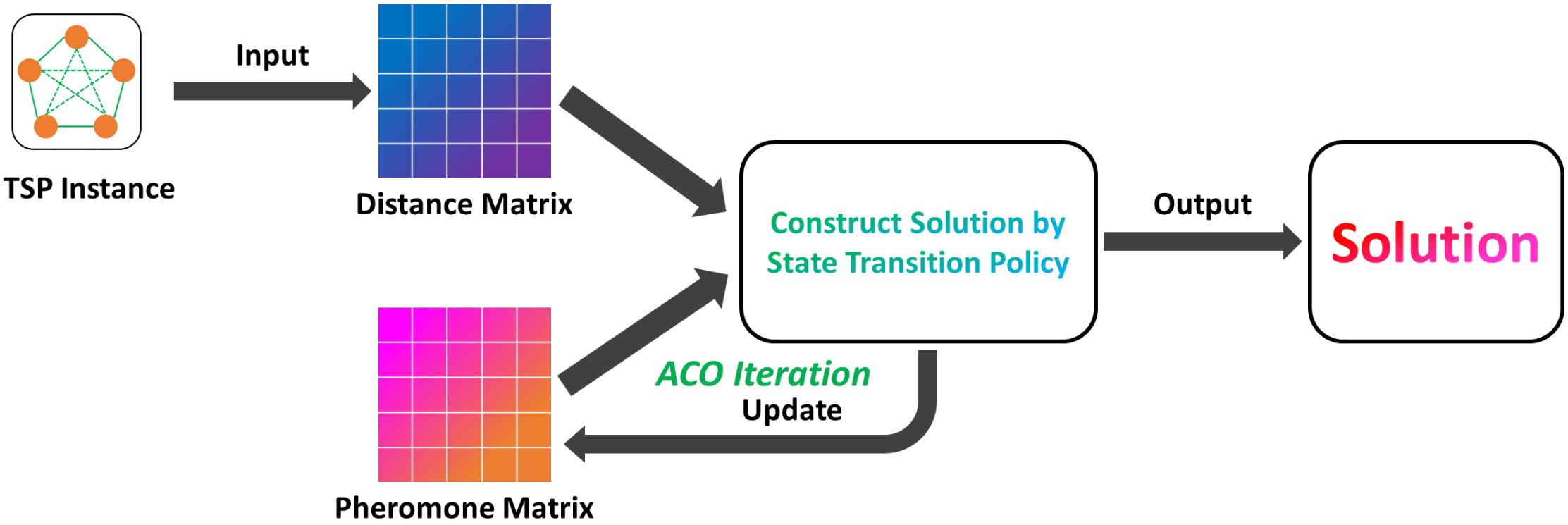


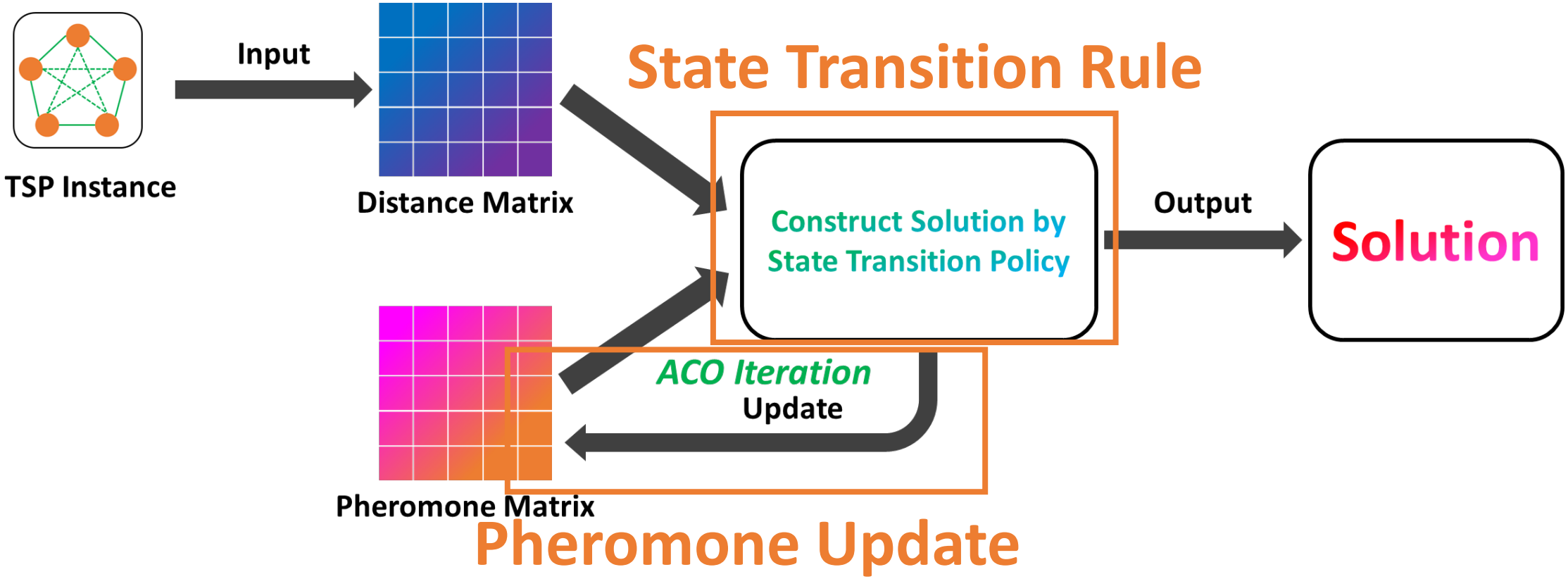
A $n \times n$ matrix records the distance d_{ij} between nodes



Pheromone Matrix

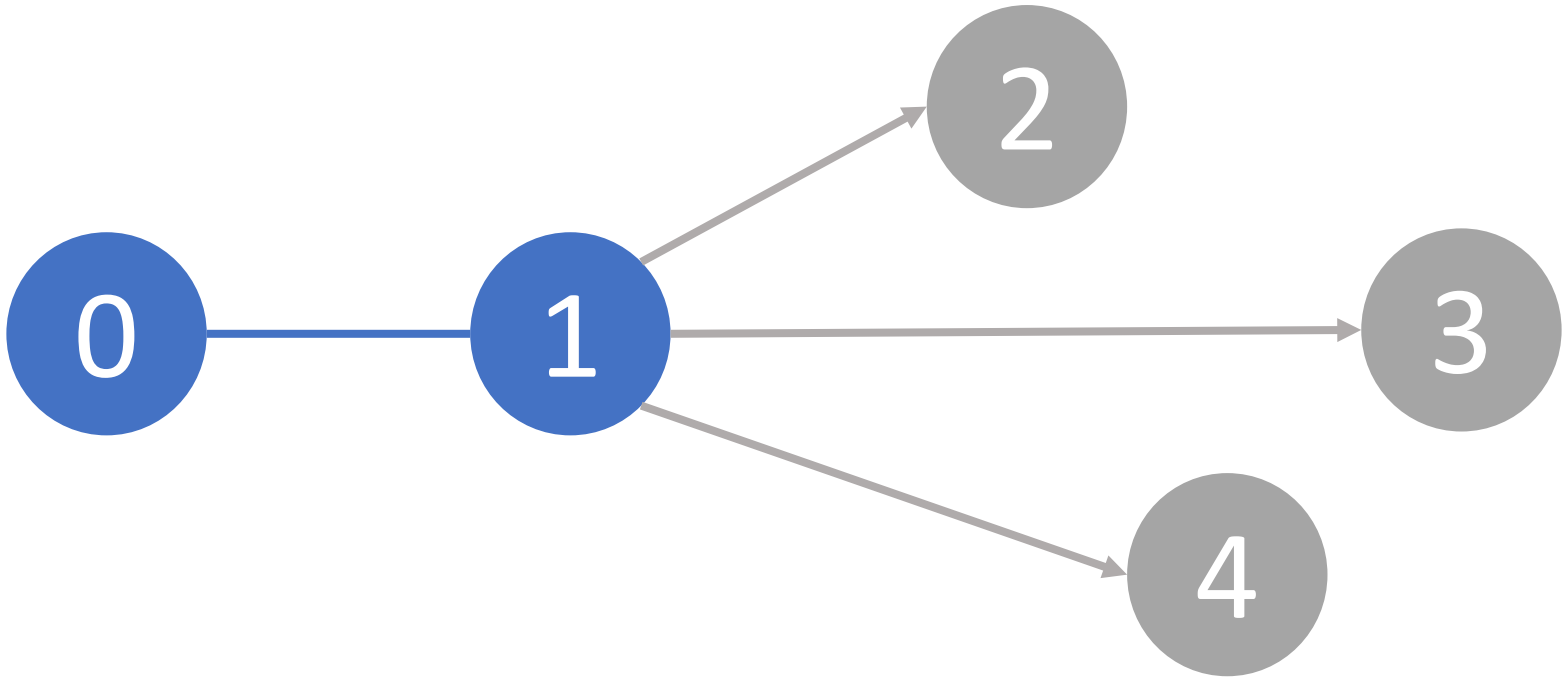
A $n \times n$ matrix records the pheromone φ_{ij} between nodes


Input
TSP Instance
Distance Matrix
Construct Solution by State Transition Policy
Output
Solution
ACO Iteration
Update
Pheromone Matrix

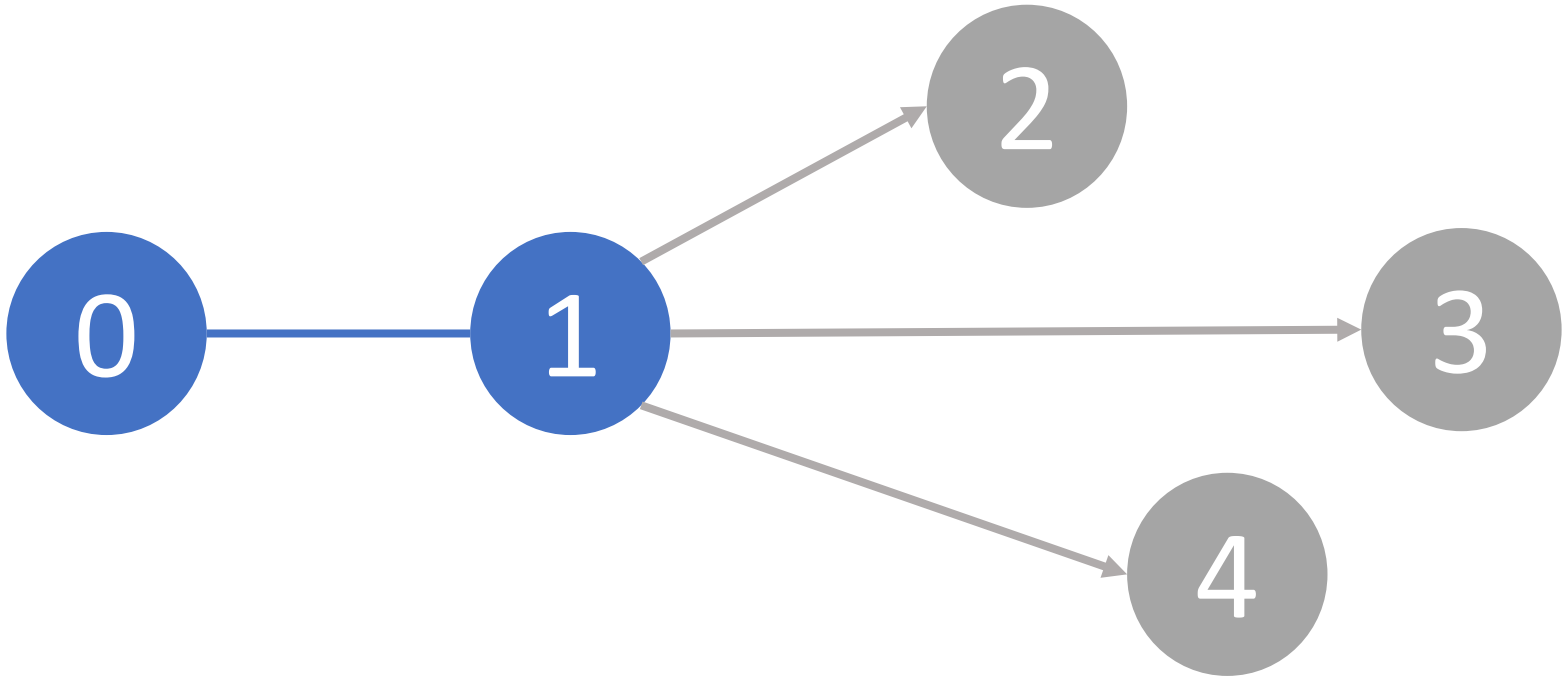


The core components in ACO

Which way to go?



Calculate a score of each candidate node



Score 1→2

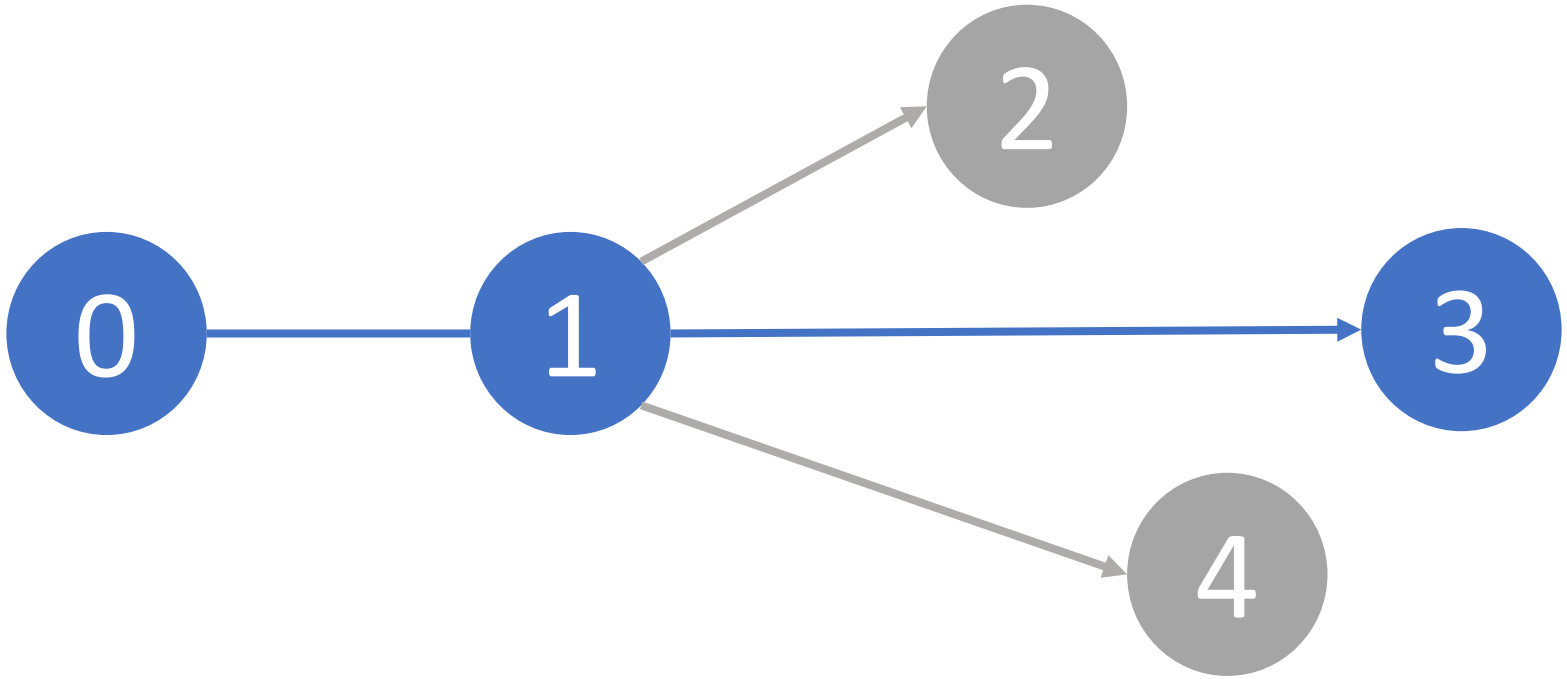
Score 1→3

Score 1→4

$$p_{ij} = \frac{\varphi_{ij}^{\alpha} / d_{ij}^{\beta}}{\sum_{l \in I_s} \varphi_{il}^{\alpha} / d_{il}^{\beta}}$$

Score

Calculate a score of each candidate node



Score 1→2

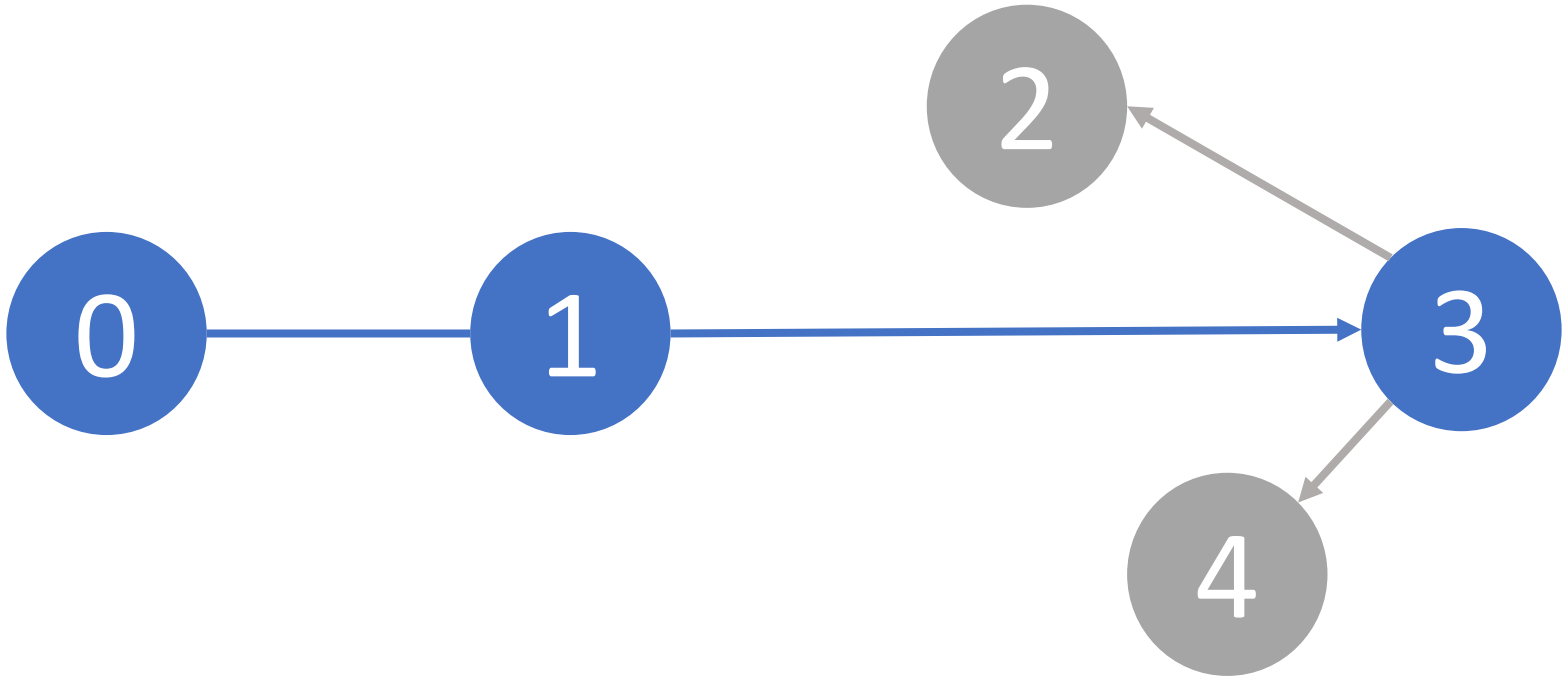
Score 1→3

Score 1→4

$$p_{ij} = \frac{\varphi_{ij}^{\alpha} / d_{ij}^{\beta}}{\sum_{l \in I_s} \varphi_{il}^{\alpha} / d_{il}^{\beta}}$$

Score

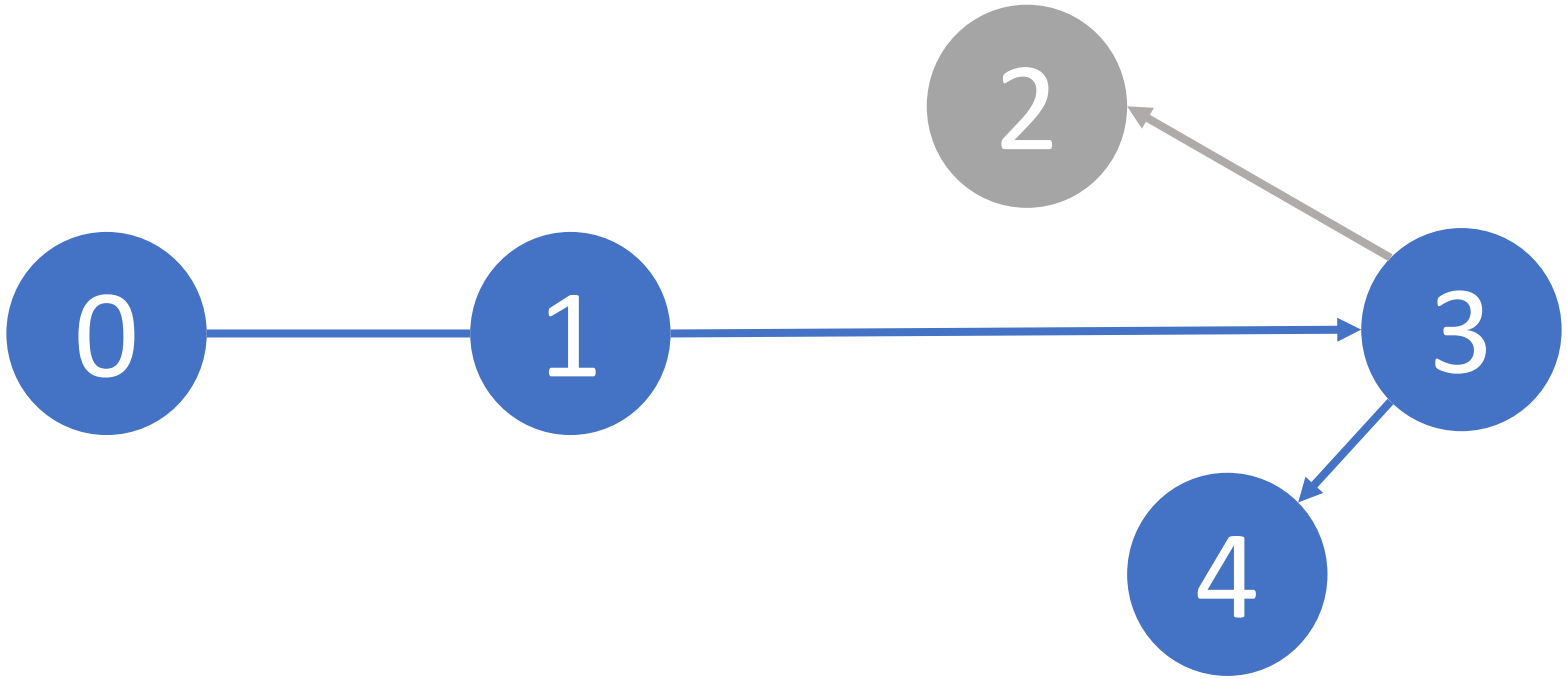
Calculate a score of each candidate node



Score 3→2

Score 3→4

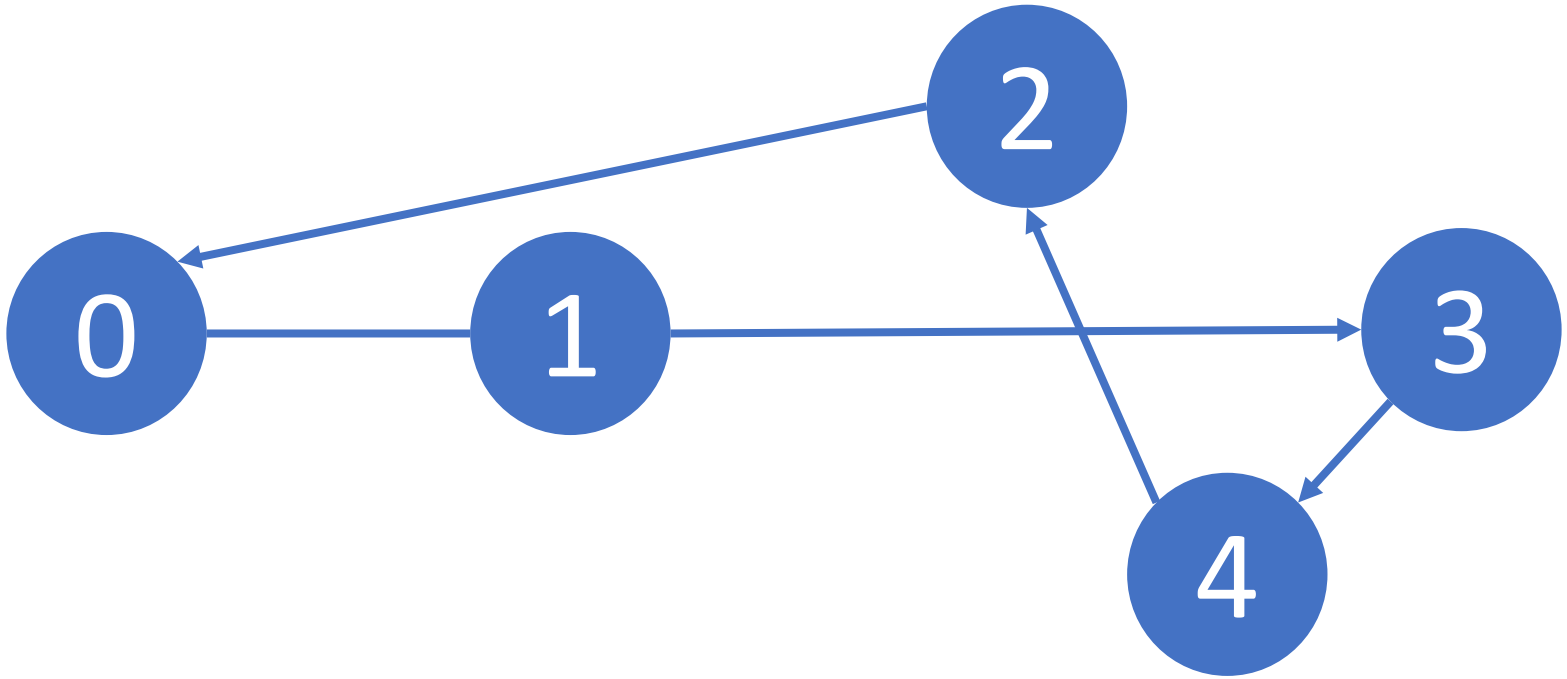
Calculate a score of each candidate node



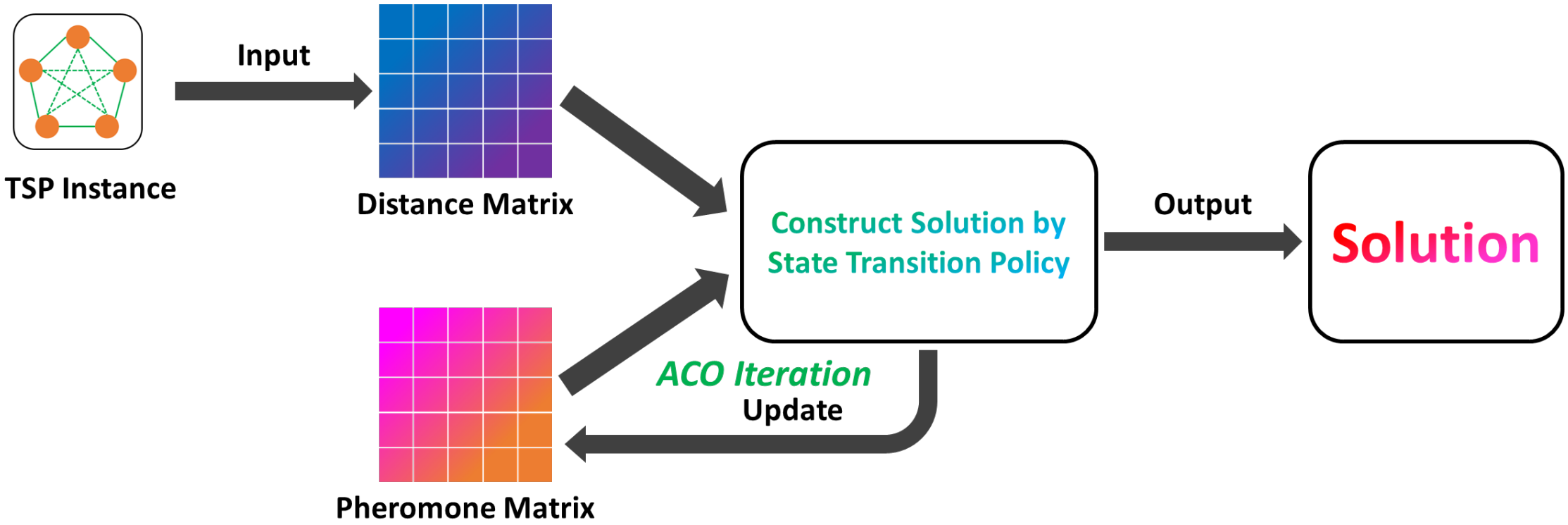
Score 3→2

Score 3→4

Complete the route



Update the pheromone after all the ants complete their routes


Input
TSP Instance
Distance Matrix
Construct Solution by
State Transition Policy
Output
Solution
ACO Iteration
Update
Pheromone Matrix

PART II

Why automated design the State Transition Rules in ACO?

Why automated design ACO?

- Heuristic Information
- Manually designed State Transition Rules
- Combinatorial Optimization Problems

Why automated design ACO?

- **Heuristic Information**
  - ***need expert experience***
- **Manually designed State Transition Rules**

- **Combinatorial Optimization Problems**

Why automated design ACO?

- Heuristic Information

  need expert experience

- Manually designed State Transition Rules

  cannot fully capture all the features

- Combinatorial Optimization Problems

Why automated design ACO?

- **Heuristic Information**
  - *need expert experience*
- **Manually designed State Transition Rules**
  - *cannot fully capture all the features*
- **Combinatorial Optimization Problems**
  - *infinite*

Why State Transition Rules?

- Pheromone Update

- State Transition Rules

Why State Transition Rules?

- Pheromone Update

  found new patterns but cannot performs better than manually designed rules

- State Transition Rules

Why State Transition Rules?

- Pheromone Update

  found new patterns but cannot performs better than manually designed rules

- State Transition Rules

  basically performs better than manual rule

Existing Works

- ~~Parameters Tuning~~
- Models Evaluation
- Model Representation

Existing Works

- **Models Evaluation**
  - Simulation-based evaluation
  - Supervised-learning based evaluation
- **Model Representation**

Existing Works

- **Models Evaluation**
  - Simulation-based evaluation
  - Supervised-learning based evaluation
- **Model Representation**
  - Tree Structure
  - Neural Networks
  - Code-level

Existing Works

- Models Evaluation

Simulation-based evaluation

Hyper-Heuristics, Large Language Models(LLMs)

Supervised-learning based evaluation

Neural-Networks, SVM

Existing Works

- Models Evaluation

Simulation-based evaluation

Hyper-Heuristics, Large Language Models(LLMs)

no need for pre-solved data

time consuming

Supervised-learning based evaluation

Neural-Networks, SVM

no need for entire ACO run to evaluate

mainly focused on heuristic information

need pre-solved data for training

Existing Works

- Models Representation

Tree Structure

Genetic Programming

Neural Networks

Neural-Networks

Code-level

LLMs, Linear GP

Existing Works

- Models Representation
  - **Tree Structure**
    - Genetic Programming
      - *very good interpretability*
  - **Neural Networks**
    - Neural-Networks
      - very *weak interpretability*
  - **Code-level**
    - LLMs, Linear GP
      - *very good interpretability*

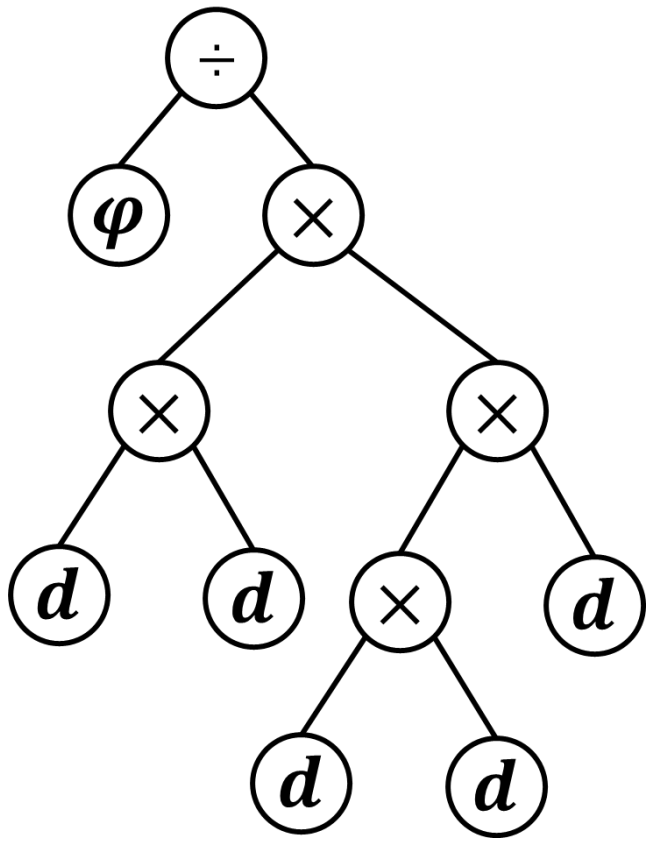
Existing Works

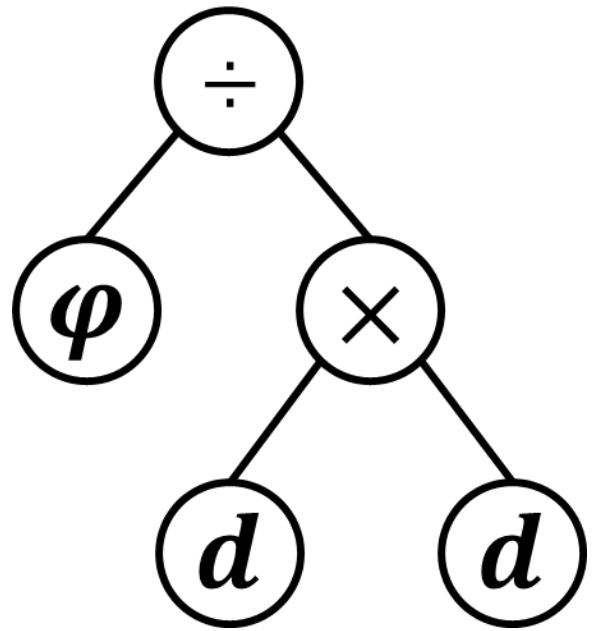
- Models Evaluation
  - Simulation-based evaluation
  - Supervised-learning based evaluation
- Model Representation
  - Tree Structure
  - Neural Networks
  - Code-level

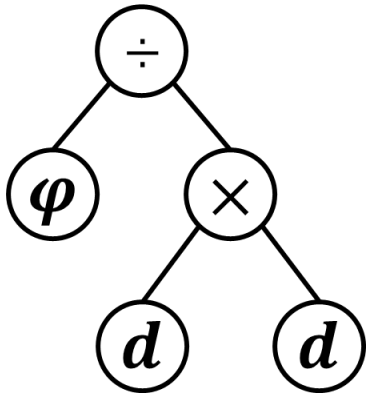
PART III

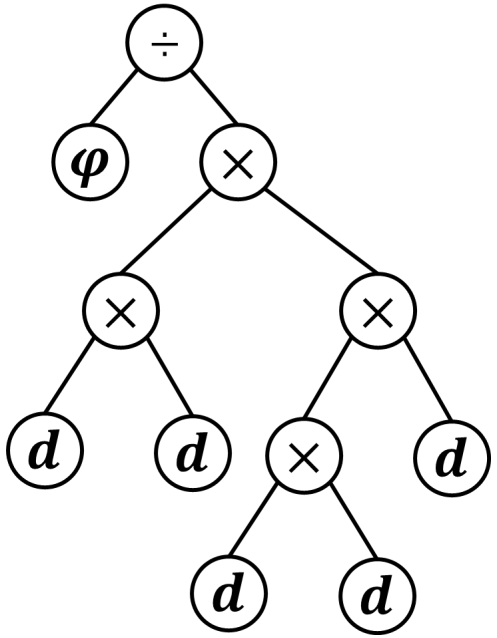
Automated Design of ACO by GP

$$p_{ij} = \frac{\varphi_{ij}^{\alpha} / d_{ij}^{\beta}}{\sum_{l \in I_s} \varphi_{il}^{\alpha} / d_{il}^{\beta}}$$









Function Set

÷ × − +

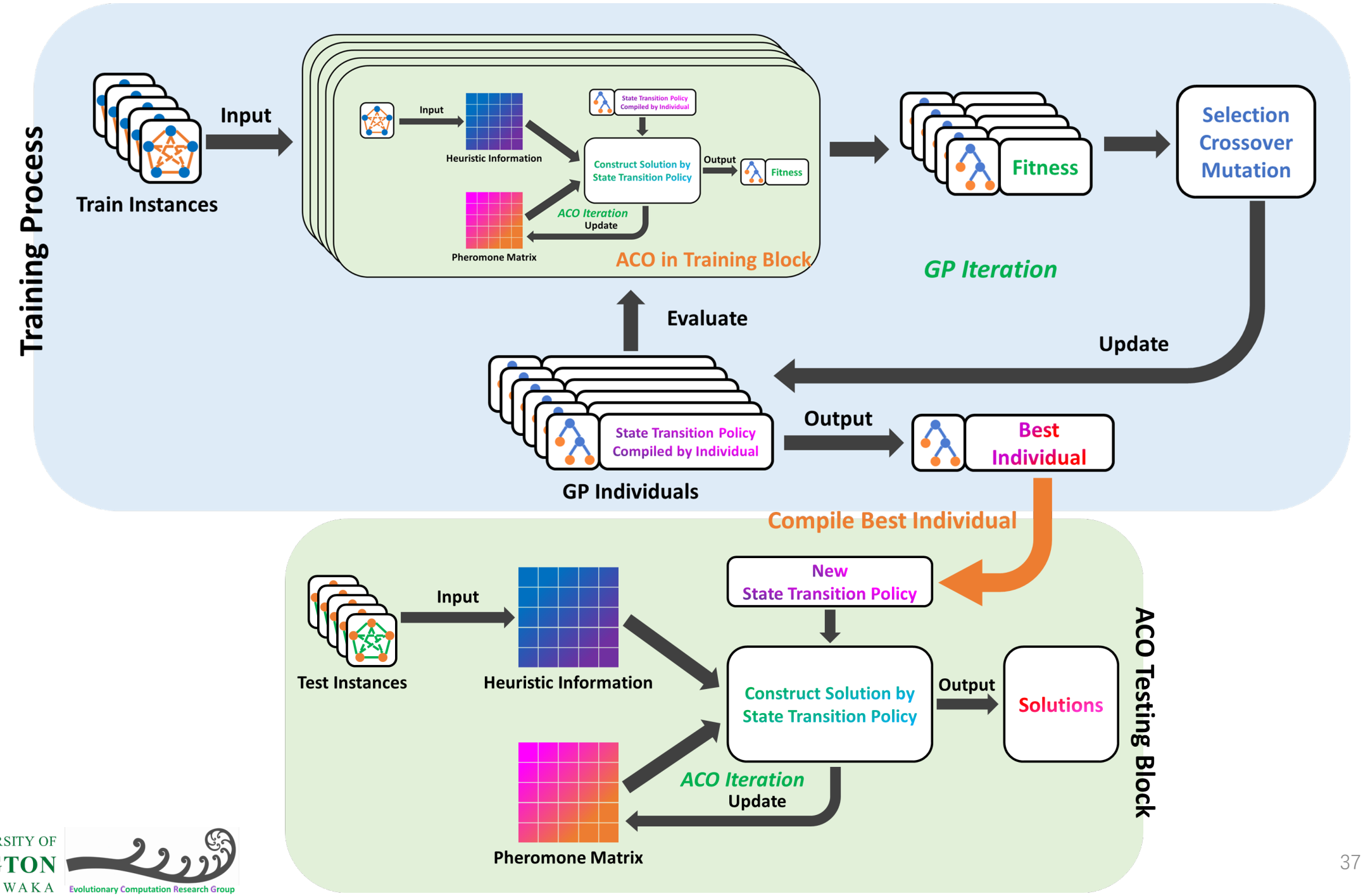
Terminal Set

φ d

Input

New
State Transition Policy

[2] Runka A. Evolving an edge selection formula for ant colony optimization[C]//Proceedings of the 11th Annual conference on Genetic and evolutionary computation. 2009: 1075-1082..


Training Process
Input
Train Instances
Input
Heuristic Information
State Transition Policy Compiled by Individual
Construct Solution by State Transition Policy
Output
Fitness
ACO Iteration
Update
Pheromone Matrix
ACO in Training Block
Fitness
Selection Crossover Mutation
GP Iteration
Update
Evaluate
State Transition Policy Compiled by Individual
GP Individuals
Output
Best Individual
Compile Best Individual
New State Transition Policy
Input
Test Instances
Heuristic Information
Construct Solution by State Transition Policy
Output
Solutions
ACO Iteration
Update
Pheromone Matrix
ACO Testing Block

Runka's Work in 2009

- Only trained on 1 instance and test on 2 instance
- Only Ant System
- No local search
- No global information

[2] Runka A. Evolving an edge selection formula for ant colony optimization[C]//Proceedings of the 11th Annual conference on Genetic and evolutionary computation. 2009: 1075-1082..

Questions:

1. Does GP-ACO exhibit generality when the distributions of training datasets and testing datasets are the same?
2. How do ACO variants influence the learning capabilities of GP-ACO?
3. Does the incorporation of local search weaken GP-ACO‘s learning effectiveness?
4. Would incorporating additional global information enhance GP-ACO’s learning capability?
5. Does GP-ACO remain effective when the distributions of training datasets and testing datasets are different?
6. How interpretable is GP-ACO?

Experiments to answer Questions:

Experiment 1: To answer Question 1 and Question 2

1. Does GP-ACO exhibit generality when the distributions of training datasets and testing datasets are the same?
2. How do ACO variants influence the learning capabilities of GP-ACO?

We apply the GP-ACO framework to AS and ACS and MMAS, without considering local search.

Train and test on three scales of TSP problems (TSP20, TSP20-100, TSP100)

Experiments to answer Questions:

Experiment 2: To answer Question 3

3. Does the incorporation of local search weaken GP-ACO's learning effectiveness?

We introduce local search to all the ACO variants in Experiment 1 while keeping the training and testing conditions unchanged.

Experiments to answer Questions:

Experiment 3: To answer Question 4

4. Would incorporating additional global information enhance GP-ACO's learning capability?

Building on Experiments 1 and 2, add four terminals containing global information to GP-ACO, which we refer to as xGP-ACO, while maintaining the same training and testing conditions.

Experiments to answer Questions:

Experiment 4: To answer Question 5

5. Does GP-ACO remain effective when the distributions of training datasets and testing datasets are different?

We directly use the state transition rules trained by xGP-ACO on the TSP100 dataset from Experiment 3, test on the publicly available TSPLIB dataset.

Datasets & Algorithm Settings

- **Datasets**

  Training: TSP20, TSP20-100, TSP100 *50 instances*

  Testing: TSP20, TSP20-100, TSP100 *10 instances*

  TSPLIB *15 instances*

- **Algorithm Settings**

GP

50 generations, 100 individuals,
0.8 crossover rate, 0.15 mutation rate,
0.05 reproduction rate,
10 elite individuals, 4 tournament size
5 maximum tree depth
30 independent training

ACO

20 (only TSP20) or 50 ants,
100 generations,
α = 1, β = 2 or 5 (only AS)
ρ = 0.9 or 0.98 (only MMAS)
q_0 = 0.9 (only ACS)
2-opt steps = cities/4

Each instance is generated using the portgen generator, which was previously used to create testbeds for the 8th DIMACS Implementation Challenge.
The portgen generator produces TSP instances, known as *rue* instances, by uniformly and randomly placing points on a two-dimensional plane.

Runka's Work in 2009

- Only trained on 1 instance and test on 2 instance
- Only Ant System
- No local search
- No global information

Before experiment, thinking:

Why Runka did not do these?

Why only Runka did this work without anyone following this?

[2] Runka A. Evolving an edge selection formula for ant colony optimization[C]//Proceedings of the 11th Annual conference on Genetic and evolutionary computation. 2009: 1075-1082..

Runka's Work in 2009

- Only trained on 1 instance and test on 2 instance
- Only Ant System
- No local search
- No global information

Execution Time

More than 50 hours

Table 1: GP parameters settings

Parameter	Value
Max Tree Depth	15
Generations	50
Population Size	200
Elitism	1 individual
Initialization	Ramped Half-and-half [2,6]
Selection	Tournament (size 4)
Crossover Type	Subtree Crossover
Crossover Percent	80%
Mutation Type	Subtree Mutation
Mutation Percent	20%
Reproduction	0%
Terminal Node Selection	10%

Table 2: AS parameters settings

Parameter	Value
Duration	100
Evaporation Rate (ρ)	0.1
Number of ants	n (problem size)
α (for default formula)	2
β (for default formula)	1
Tournament Size (when needed)	7

[2] Runka A. Evolving an edge selection formula for ant colony optimization[C]//Proceedings of the 11th Annual conference on Genetic and evolutionary computation. 2009: 1075-1082..

Accelerate Python

for GP-ACO

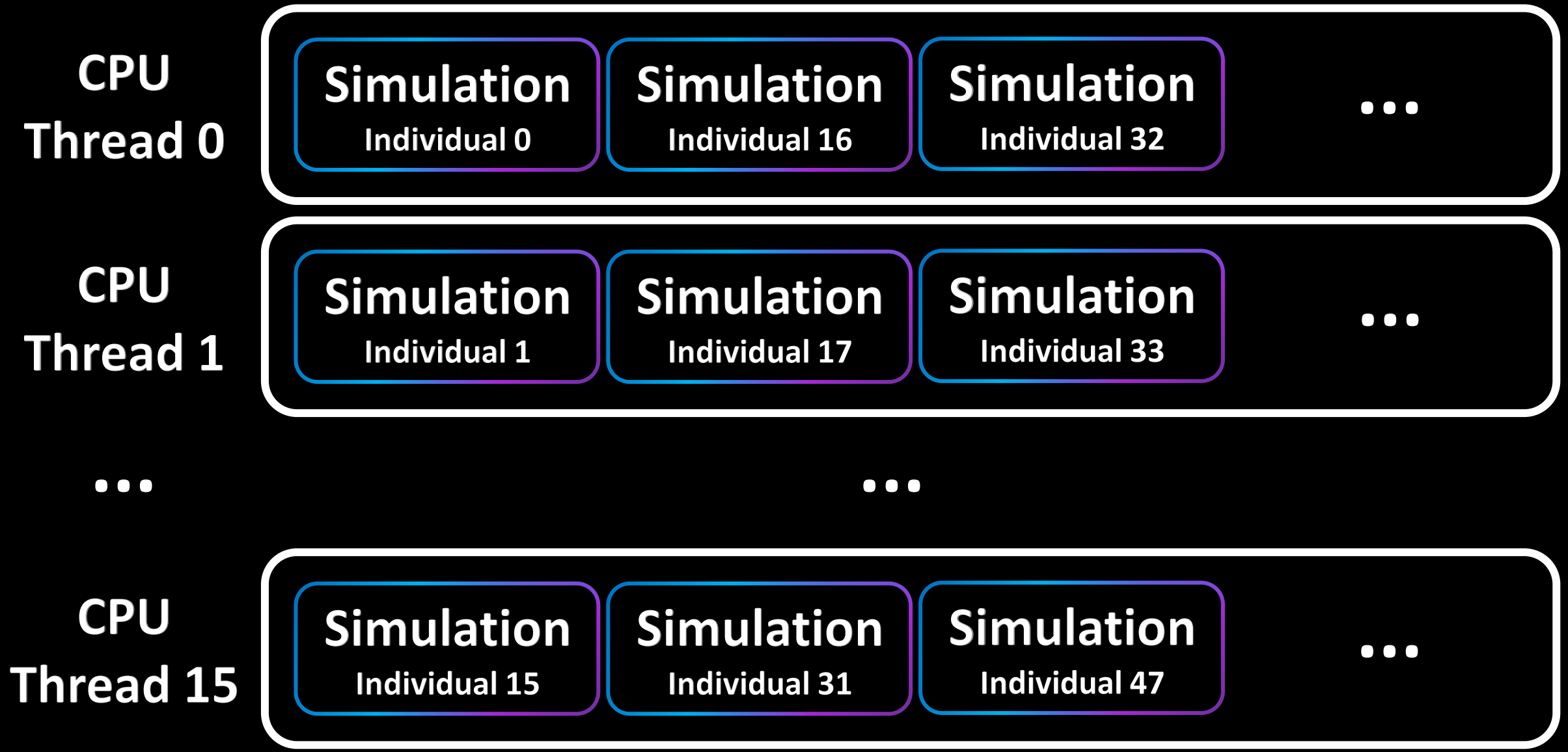
How to accelerate your python Genetic Programming Codes?

```
for individual in population:
        simulation(individual);
        offspring;
end for
```

```
for individual in population:
        simulation(individual);
        offspring;
end for
```

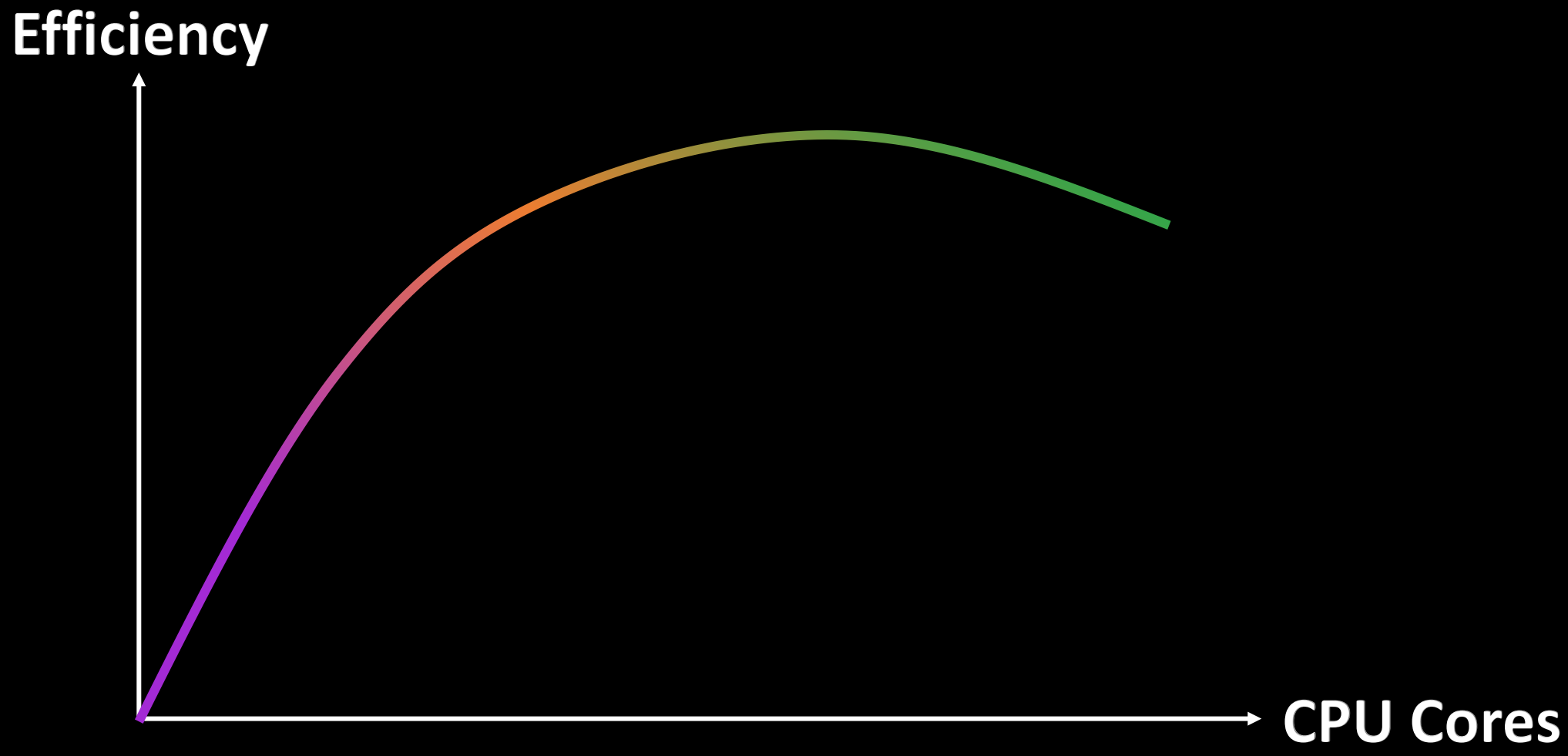
Parallel

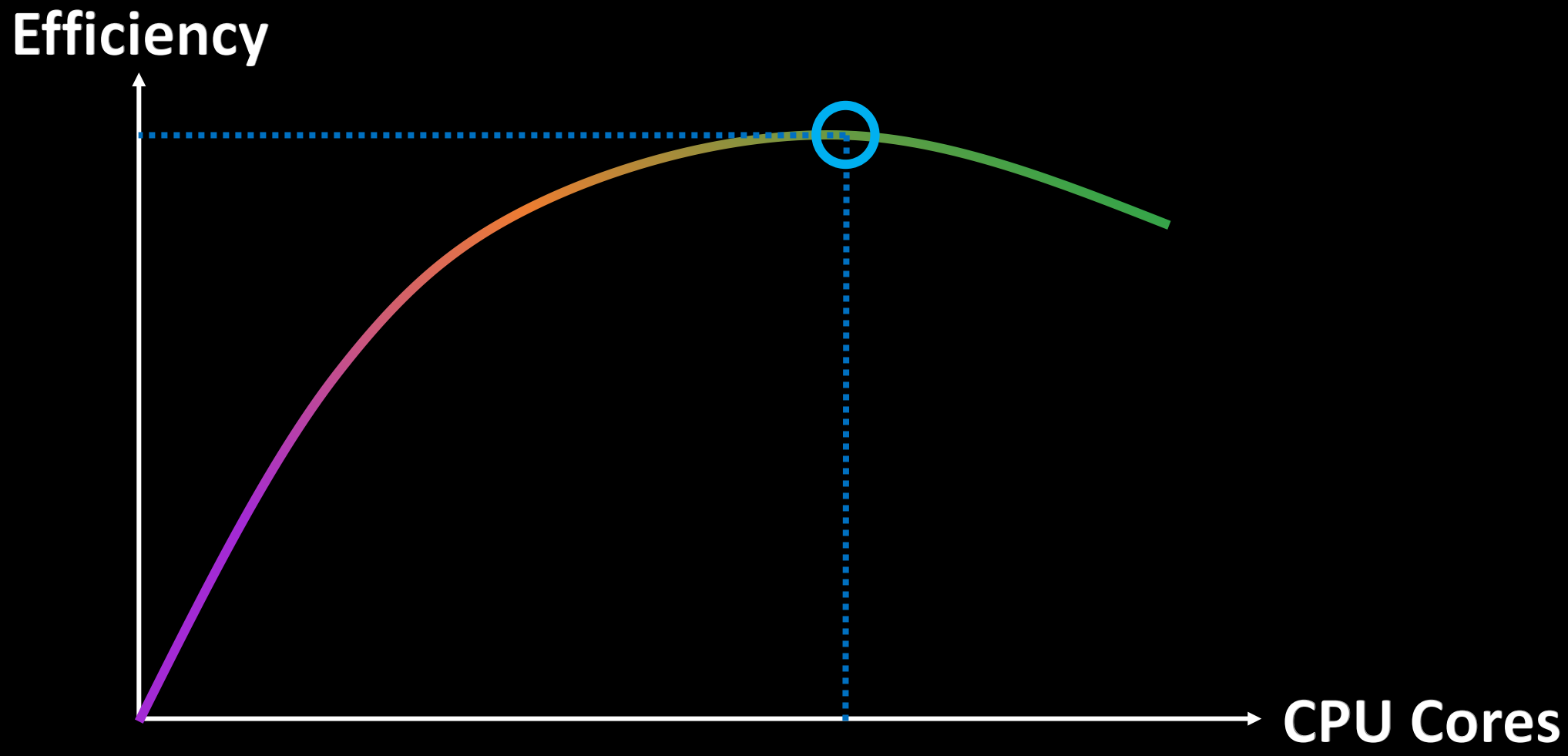
Parallel Simulation/Evaluation

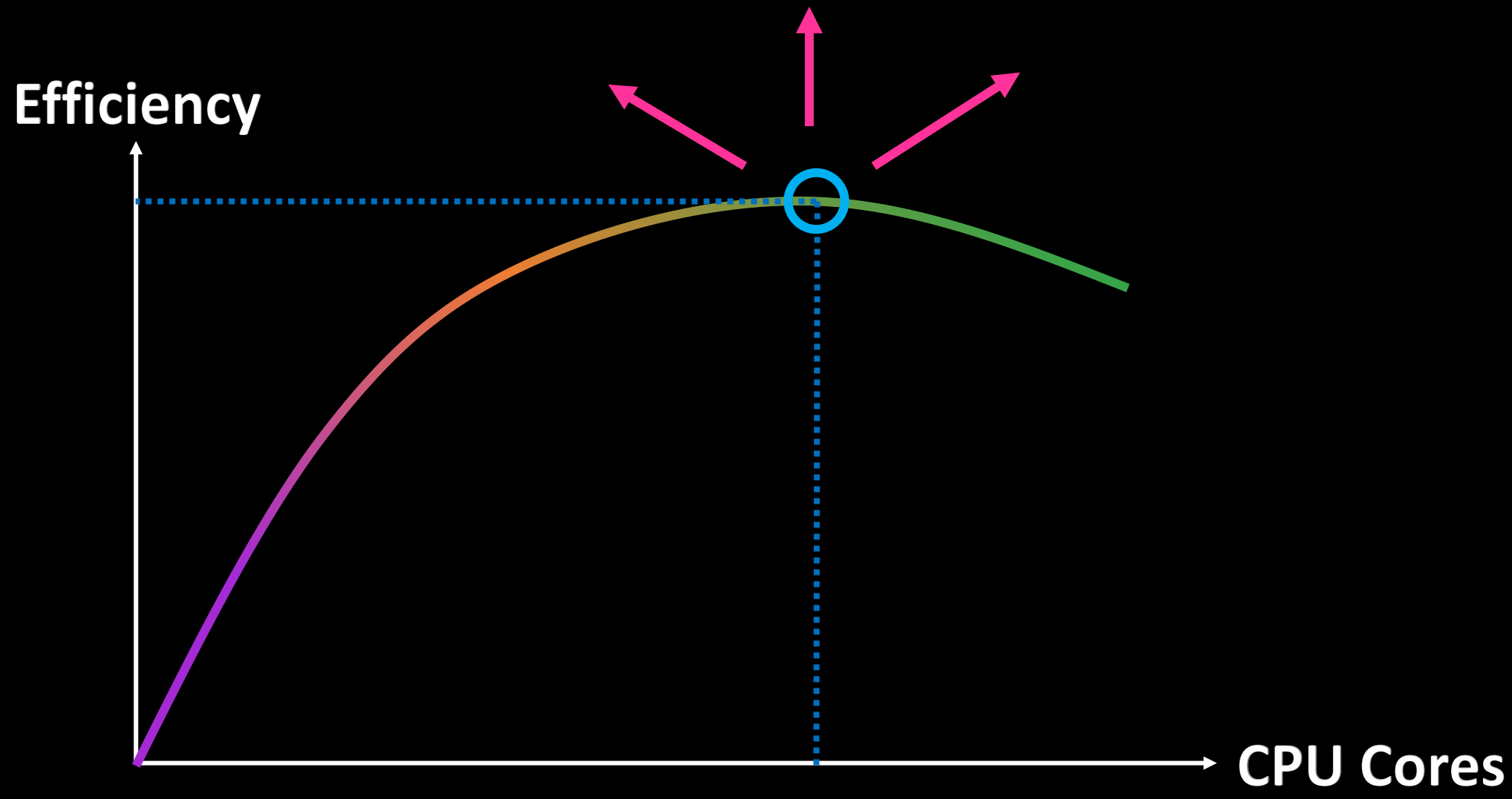


```
# regist
executor = concurrent.futures.ProcessPoolExecutor ( max_workers = cores )
toolbox.register ( "multiprocessing", executor.map )

# parallel evaluation
invalid_ind = [ ind for ind in population ]
fitnesses = toolbox.multiprocessing ( toolbox.evaluate, invalid_ind )
for ind, fit in zip ( invalid_ind, fitnesses ) :
    ind.fitness.values = ( fit, )
```


Efficiency
CPU Cores


Efficiency
CPU Cores


Efficiency
CPU Cores

Go into Simulation

Loops to Matrix

(Matrix A × Matrix B).sum (ms)

Matrix_size	Loops	Numpy	PyTorch_CPU	Numba_Loops
1000	232.2309	1.9934	2.9869	0.9961
2000	817.2810	9.9669	5.9803	1.9991
3000	1793.0334	22.9242	12.9590	1.9944
4000	3219.2883	42.8576	20.9301	4.9837
5000	4919.6362	66.7777	31.8937	6.9737
6000	7170.1484	92.6917	43.8540	10.9632
7000	9691.7591	130.5678	61.7921	15.9428
8000	12531.3115	165.4499	77.7411	17.9372
9000	16748.4260	204.3159	96.6794	21.9247
10000	19626.6966	267.1080	119.6053	27.9043

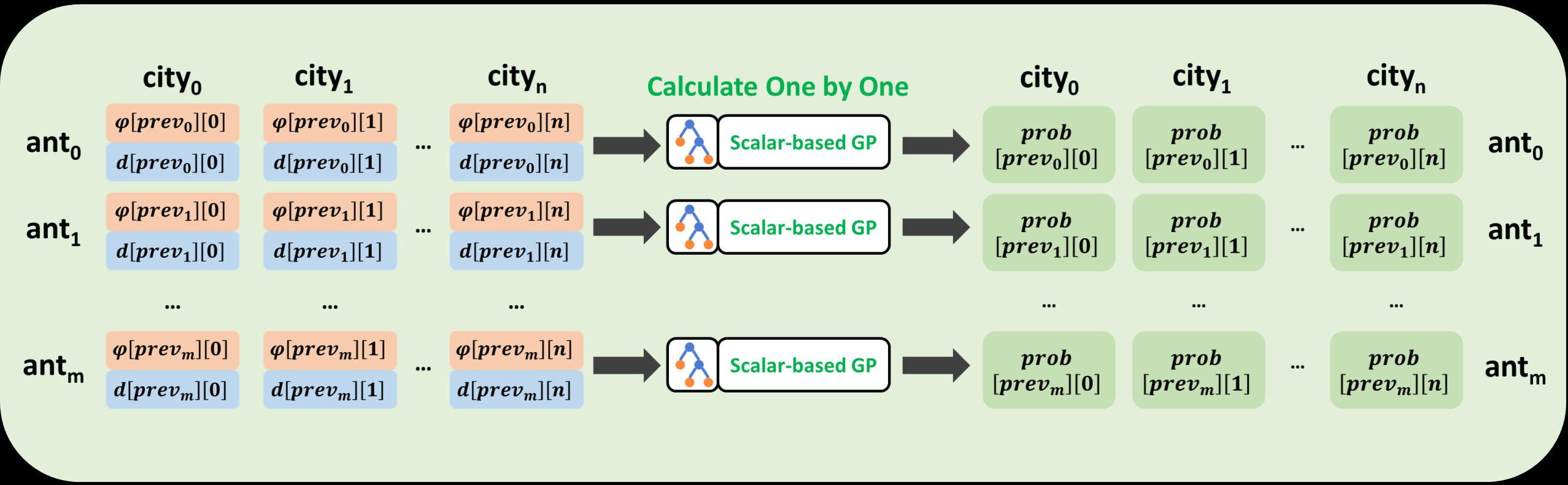
Intel i7-13700, Python3.8, PyTorch1.7

```
for individual in population:
        simulation(individual);
        offspring;
end for
```

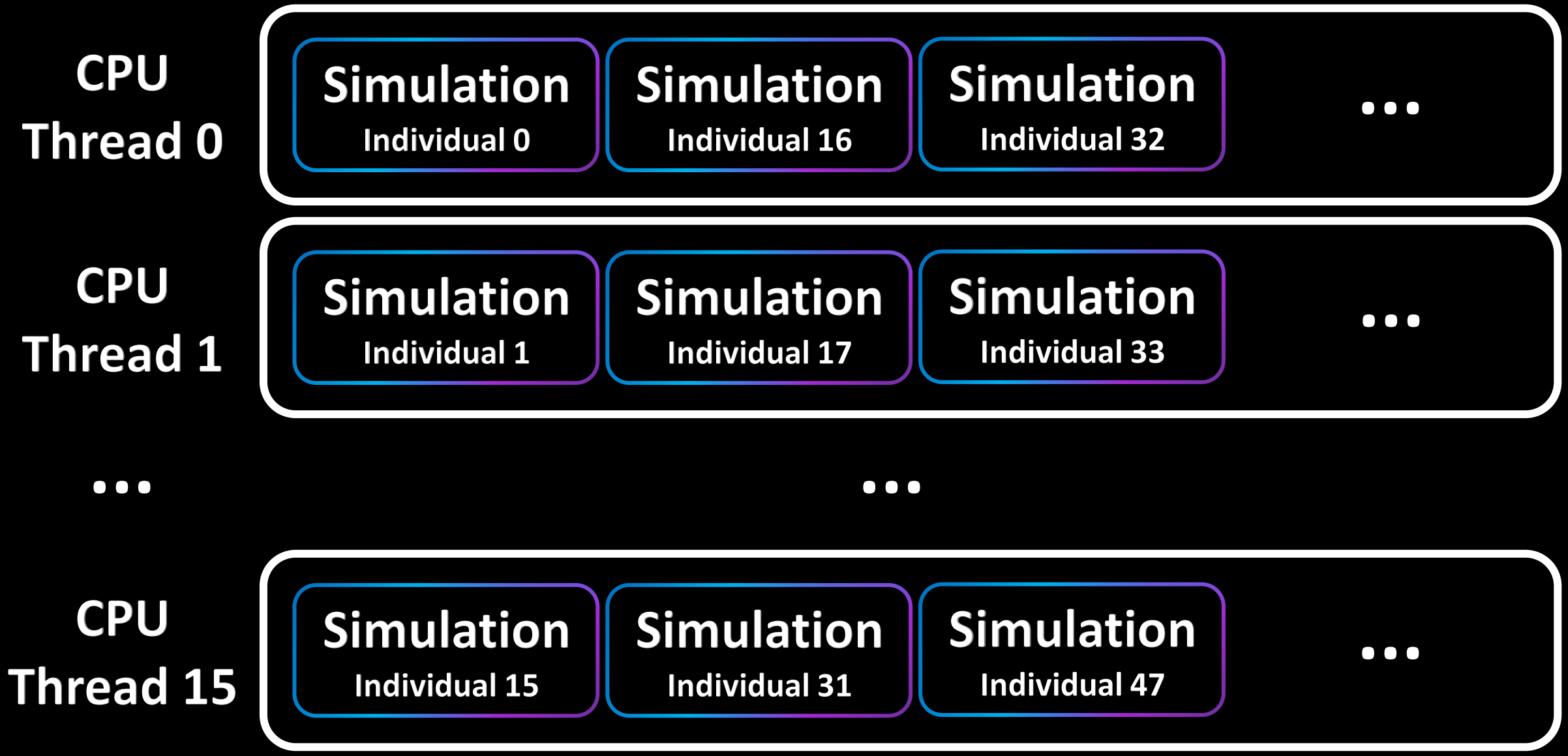
```
for individual in population:
    #ACO simulation
    for ant in all_ants:
        while unvisited_cities is not empty:
            for cities in unvisted_cities:
                calculate_priority by individual;
            end for;
            choose next_city;
        end while;
    end for;
    offspring;
end for
```

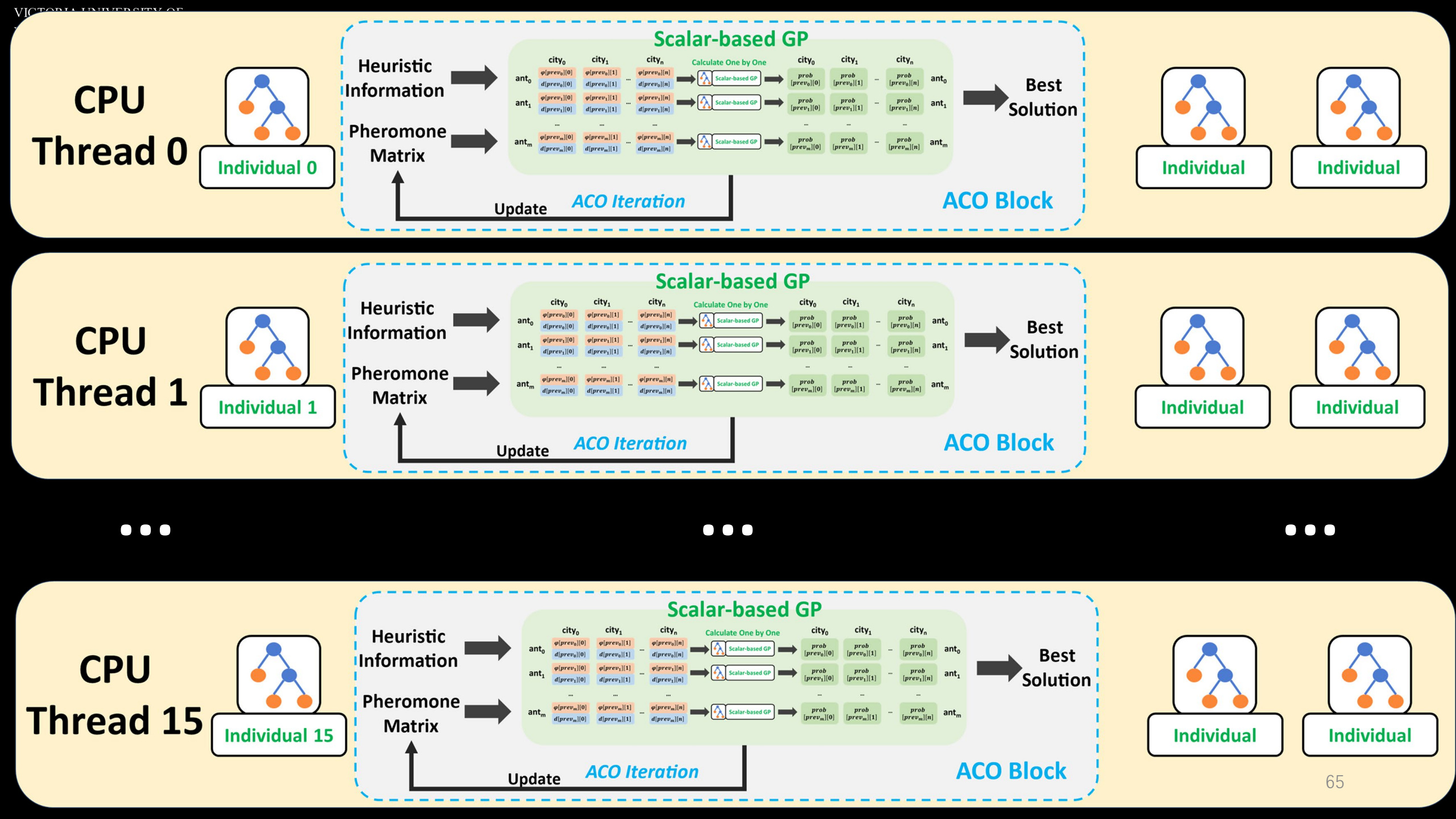
```
for individual in population:
    #ACO simulation
    for ant in all_ants:
        while unvisited_cities is not empty:
            for cities in unvisted_cities:
                calculate_priority by individual;
            end for;
            choose next_city;
        end while;
    end for;
    offspring;
end for
```

for *ant* **in** *all_ants*:
while *unvisited_cities* **is not empty:**
for *cities* **in** *unvisted_cities*:



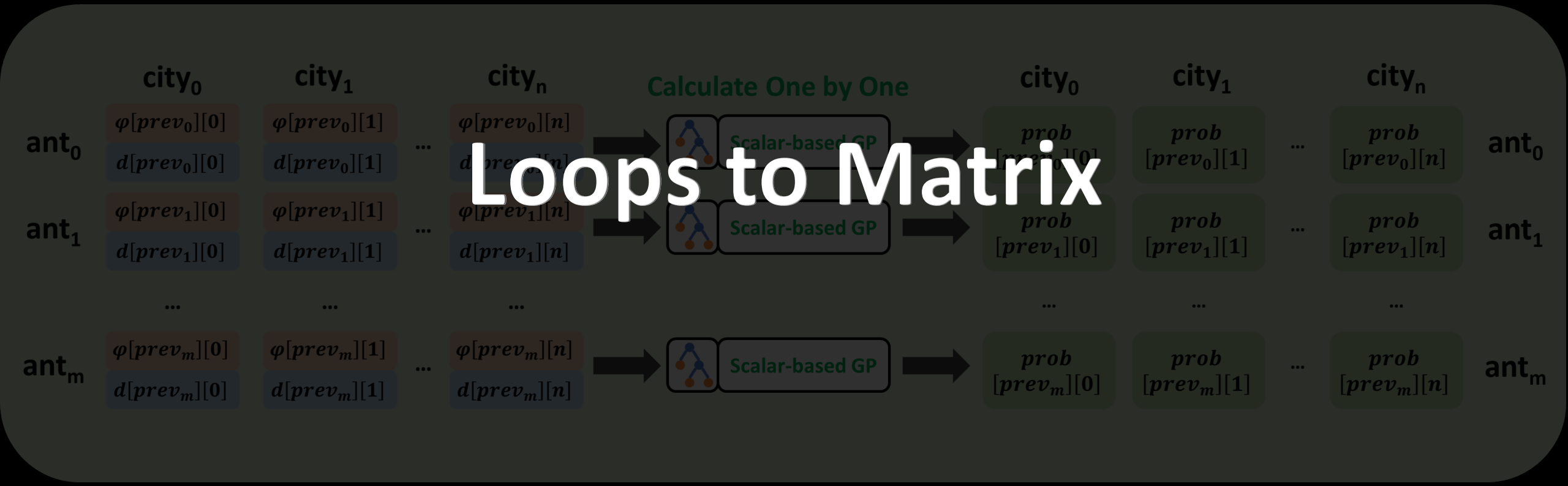
Parallel Simulation/Evaluation



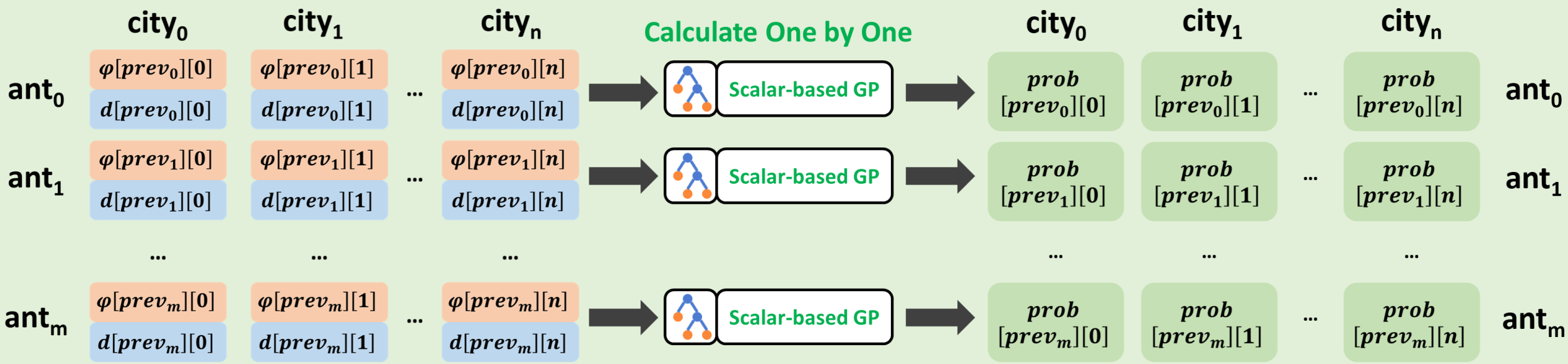

CPU Thread 0
Individual 0
Heuristic Information
Pheromone Matrix
Scalar-based GP
Calculate One by One
Scalar-based GP
Best Solution
Update
ACO Iteration
ACO Block
Individual
Individual
CPU Thread 1
Individual 1
Heuristic Information
Pheromone Matrix
Scalar-based GP
Calculate One by One
Scalar-based GP
Best Solution
Update
ACO Iteration
ACO Block
Individual
Individual
CPU Thread 15
Individual 15
Heuristic Information
Pheromone Matrix
Scalar-based GP
Calculate One by One
Scalar-based GP
Best Solution
Update
ACO Iteration
ACO Block
Individual
Individual

VICTORIA UNIVERSITY OF WELLINGTON
TE HERENGA WAKA

```
for ant in all_ants:
    while unvisited_cities is not empty:
        for cities in unvisted_cities:
```

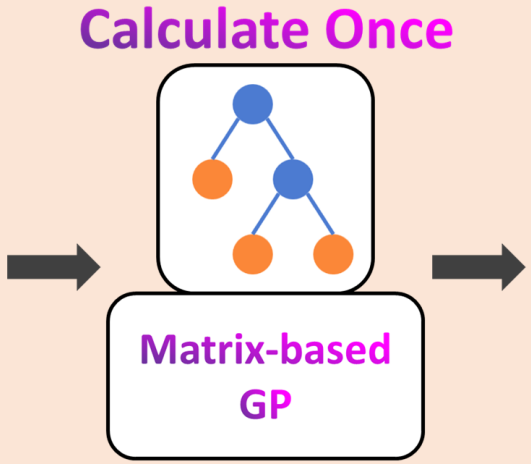


Loops to Matrix



$$\begin{bmatrix} \varphi[prev_0][0] & \cdots & \varphi[prev_0][n] \\ \vdots & \ddots & \vdots \\ \varphi[prev_m][0] & \cdots & \varphi[prev_0][n] \end{bmatrix}$$

$$\begin{bmatrix} d[prev_0][0] & \cdots & d[prev_0][n] \\ \vdots & \ddots & \vdots \\ d[prev_m][0] & \cdots & d[prev_0][n] \end{bmatrix}$$



$$\begin{bmatrix} prob[prev_0][0] & \cdots & prob[prev_0][n] \\ \vdots & \ddots & \vdots \\ prob[prev_m][0] & \cdots & prob[prev_0][n] \end{bmatrix}$$

```
for individual in population:
    #ACO simulation
    for ant in all_ants:
        while unvisited_cities is not empty:
            for cities in unvisted_cities:
                calculate_priority by individual;
                choose next_city;
            end for;
        end while;
    end for;
    offspring;
end for
```

```
for individual in population:
    #ACO simulation
    for ant in all_ants:
        while unvisited_cities is not empty:
            for cities in unvisted_cities:
                calculate_priority by individual;
            choose next_city;
        end for;
    end while;
end for;
offspring;
end for
```

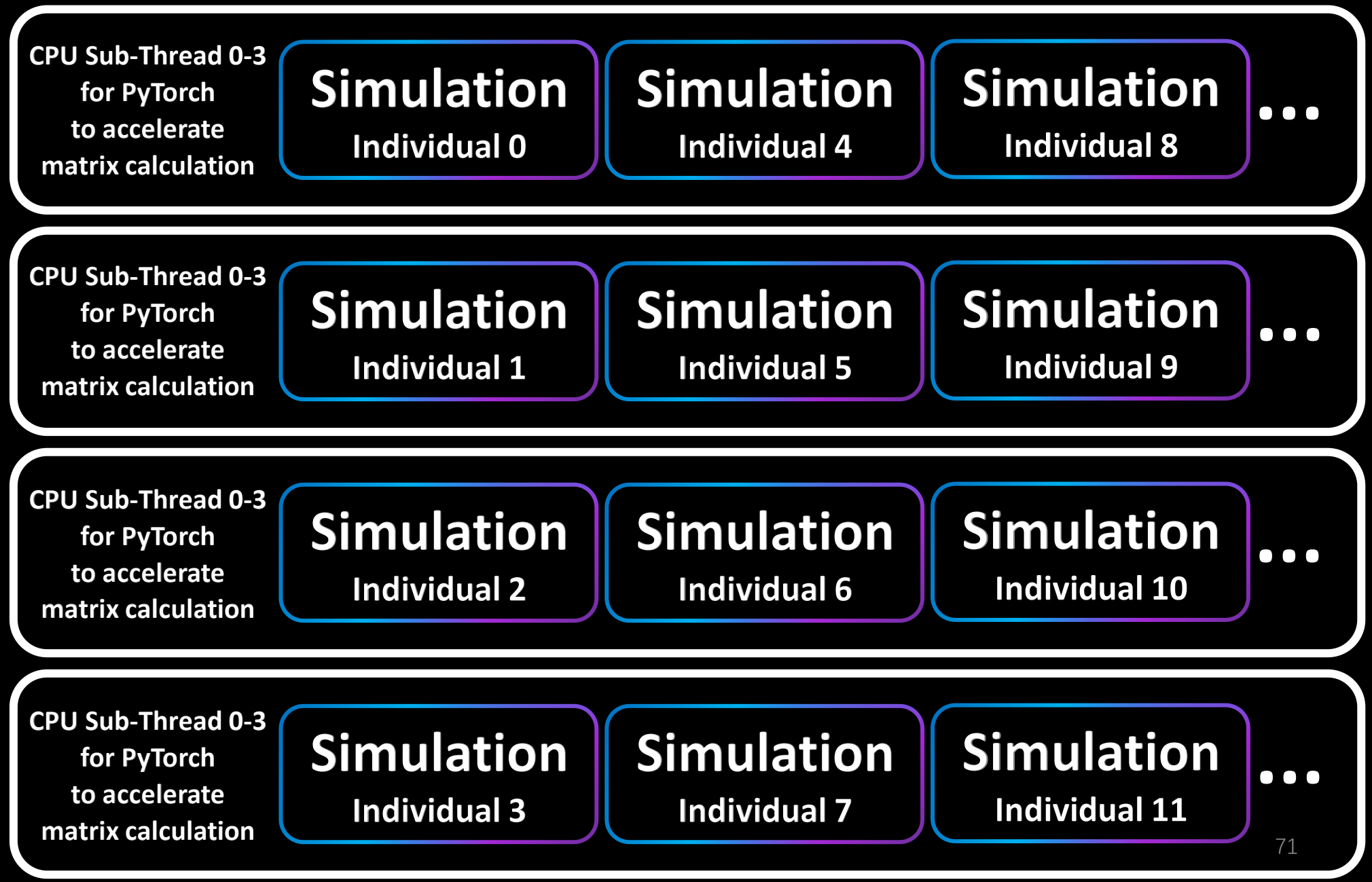
How to deal with *unvisited_cities* ?

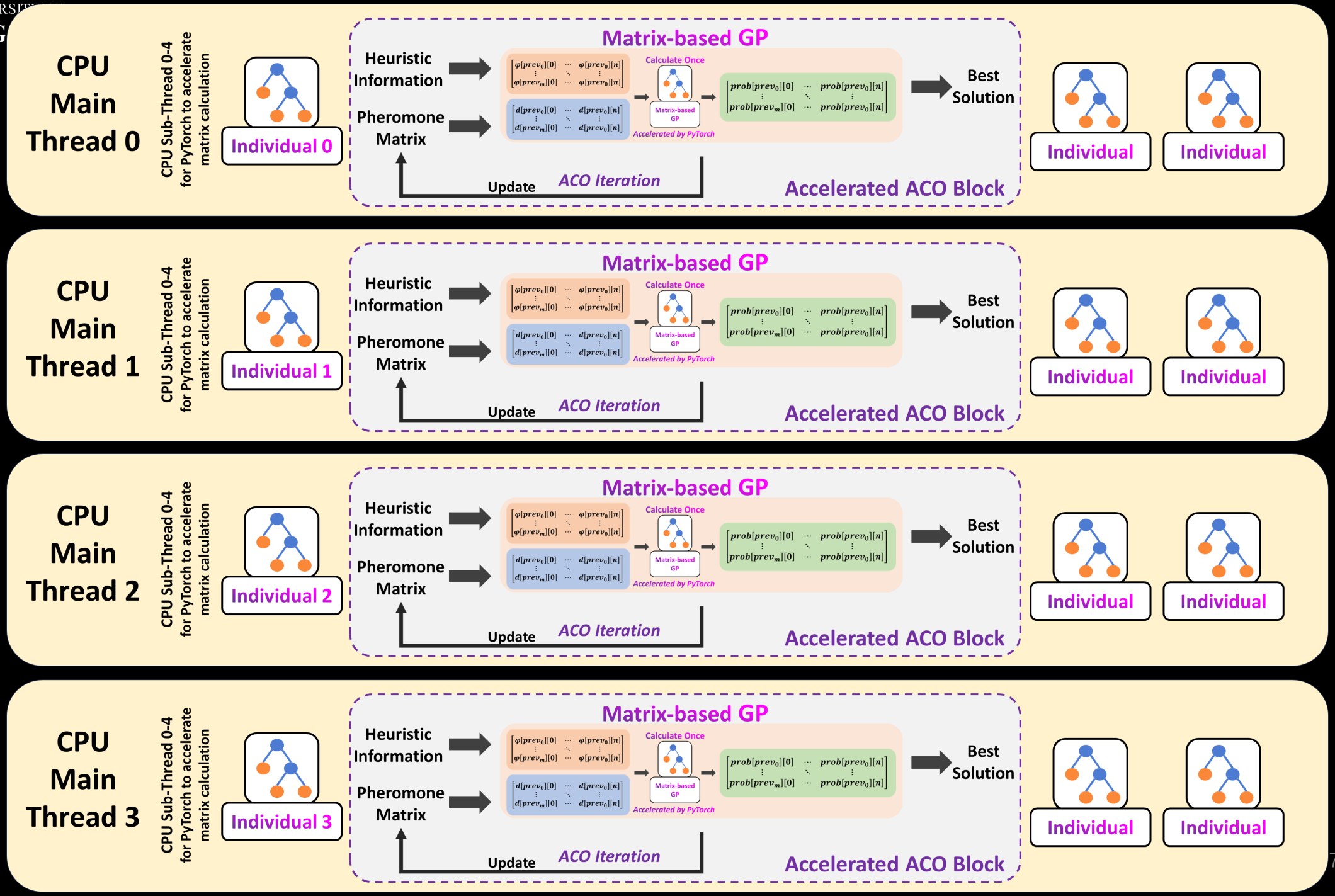
```
for individual in population:
    #ACO simulation
    for ant in all_ants:
        while unvisited_cities is not empty:
            for cities in unvisted_cities:
                calculate_priority by individual;
                choose next_city;
            end for;
        end while;
    end for;
    offspring;
end for
```

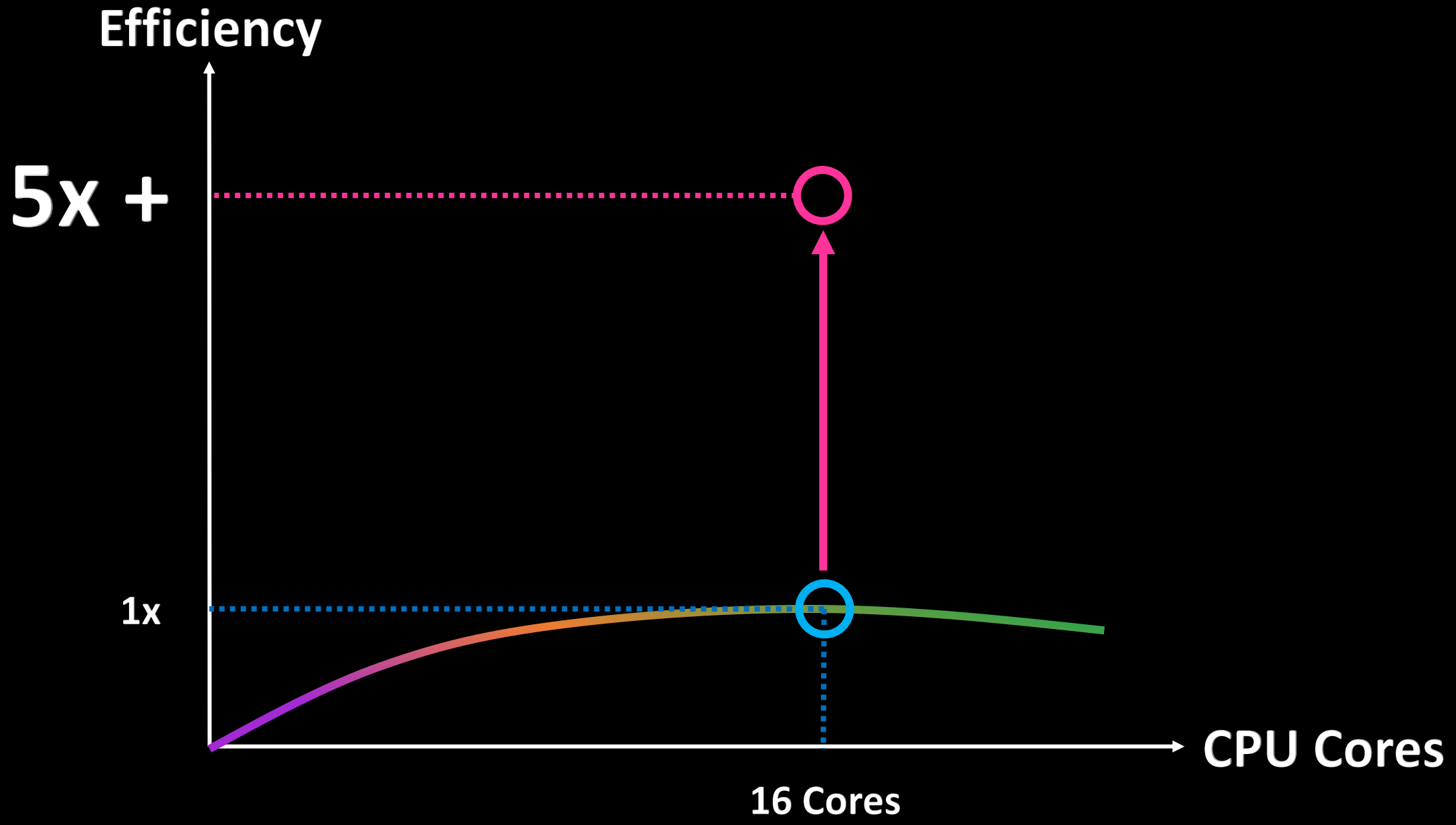
Matrix A × Matrix B × Mask

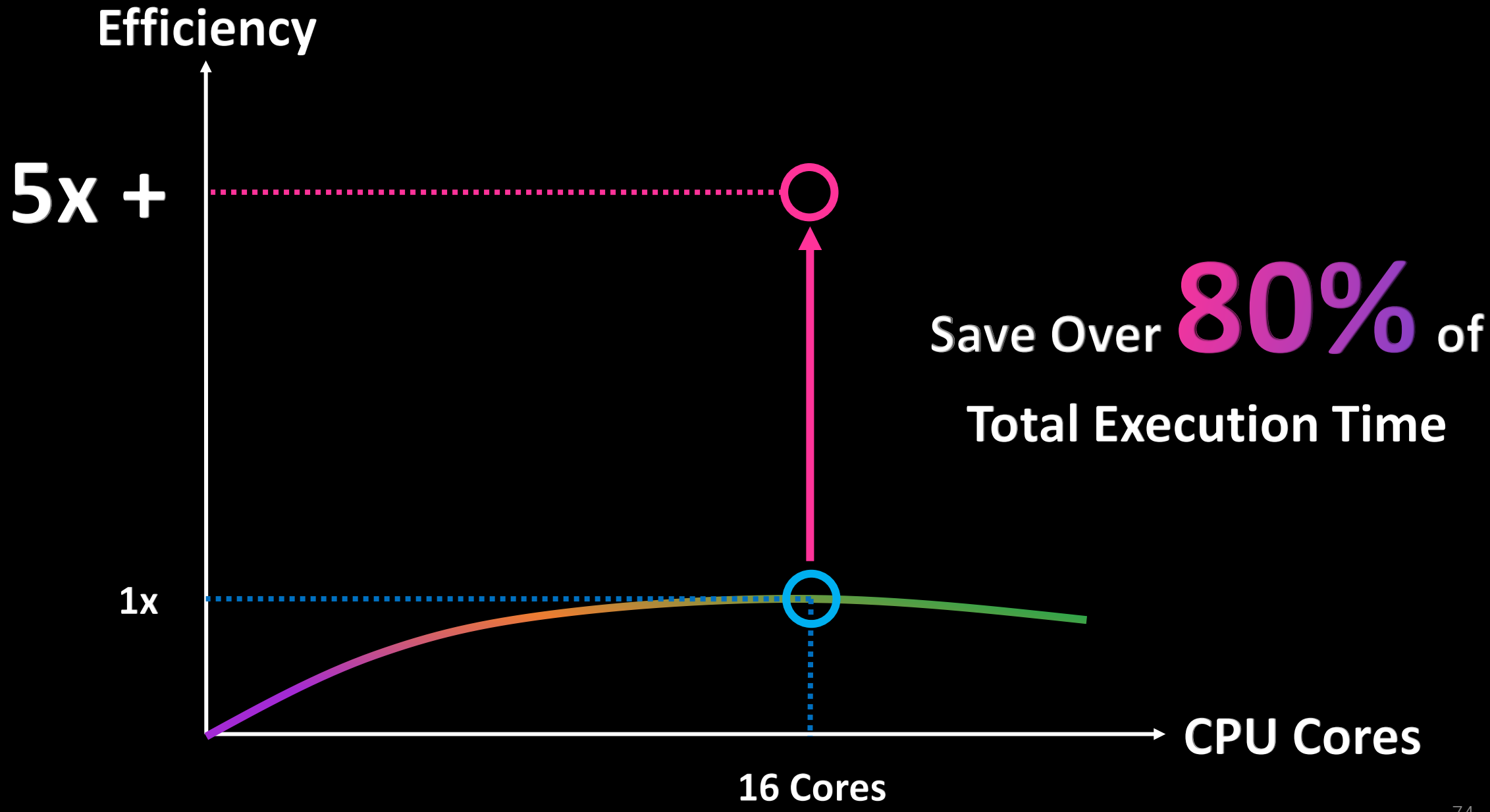
VICTORIA UNIVERSITY OF WELLINGTON
TE HERENGA WAKA

Parallel inside Simulation/Evaluation




CPU Main Thread 0
CPU Sub-Thread 0-4 for PyTorch to accelerate matrix calculation
Individual 0
Matrix-based GP
Heuristic Information
Pheromone Matrix
Calculate Once
Matrix-based GP
Accelerated by PyTorch
Best Solution
Update
ACO Iteration
Accelerated ACO Block
Individual
Individual
CPU Main Thread 1
CPU Sub-Thread 0-4 for PyTorch to accelerate matrix calculation
Individual 1
Matrix-based GP
Heuristic Information
Pheromone Matrix
Calculate Once
Matrix-based GP
Accelerated by PyTorch
Best Solution
Update
ACO Iteration
Accelerated ACO Block
Individual
Individual
CPU Main Thread 2
CPU Sub-Thread 0-4 for PyTorch to accelerate matrix calculation
Individual 2
Matrix-based GP
Heuristic Information
Pheromone Matrix
Calculate Once
Matrix-based GP
Accelerated by PyTorch
Best Solution
Update
ACO Iteration
Accelerated ACO Block
Individual
Individual
CPU Main Thread 3
CPU Sub-Thread 0-4 for PyTorch to accelerate matrix calculation
Individual 3
Matrix-based GP
Heuristic Information
Pheromone Matrix
Calculate Once
Matrix-based GP
Accelerated by PyTorch
Best Solution
Update
ACO Iteration
Accelerated ACO Block
Individual
Individual


Efficiency
5x +
1x
CPU Cores
16 Cores



Save Over **80%** of

Total Execution Time

Experiment 1: To answer Question 1 and Question 2

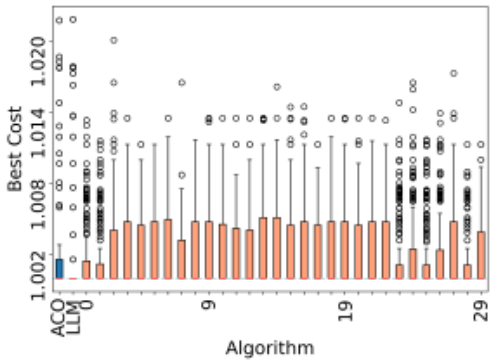
1. Does GP-ACO exhibit generality when the distributions of training datasets and testing datasets are the same?
2. How do ACO variants influence the learning capabilities of GP-ACO?

We apply the GP-ACO framework to AS and ACS and MMAS, without considering local search.

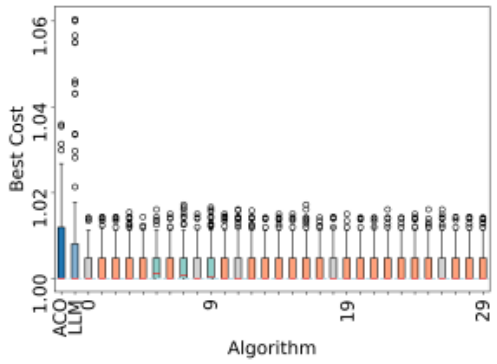
Train and test on three scales of TSP problems (TSP20, TSP20-100, TSP100)

Table 1: The normalized costs of GP-ACO, ACO baselines, and LLM-ACO on the test datasets.

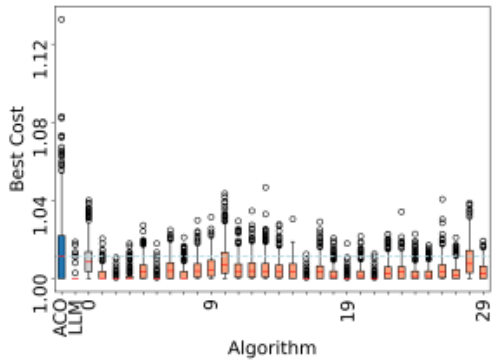
TSP		AS			ACS			MMAS		
		baseline	LLM	GP	baseline	LLM	GP	baseline	LLM	GP
20	mean	1.0028	1.0020	1.0019	1.0063	1.0074	**1.0025**	1.0164	**1.0031**	1.0039
		+	+		+	+		+	-	
	std	0	0	0.0001	0	0	0.0002	0	0	0.0002
	min	1	1	1	1	1	1	1	1	1
	max	1.0217	1.0218	1.0201	1.0359	1.0604	1.0172	1.1327	1.0186	1.0467
20-100	mean	1.0364	1.0271	**1.0265**	1.0499	1.0418	**1.0306**	1.0632	1.0076	1.0254
		+	+		+	+		+	-	
	std	0	0	0.0004	0	0	0.0004	0	0	0.0001
	min	1	1	1	1.0006	1.0016	1	1	1	1
	max	1.1042	1.0747	1.3813	1.1445	1.0831	1.1660	1.2101	1.0440	1.1212
100	mean	1.0784	1.0533	1.0105	1.0867	1.0721	**1.0446**	1.1248	**1.0144**	1.0400
		+	+		+	+		+	-	
	std	0	0	0.0001	0	0	0.0005	0	0	0.0001
	min	1.0359	1.0127	1.0005	1.0310	1.0148	1.0162	1.0481	1	1.0071
	max	1.1346	1.0962	1.0274	1.1451	1.1154	1.0799	1.2604	1.0680	1.0852



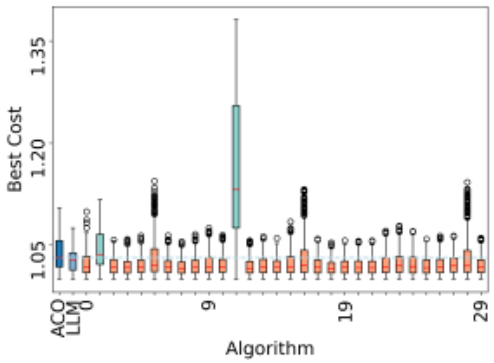
(a) TSP20 - GP-AS



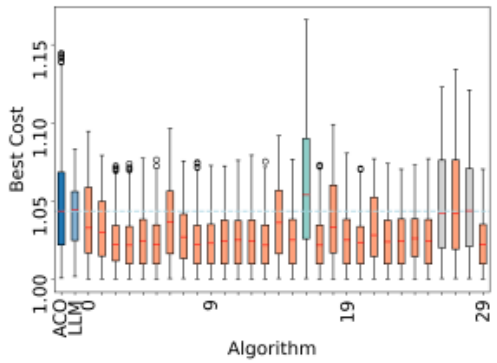
(b) TSP20 - GP-ACS



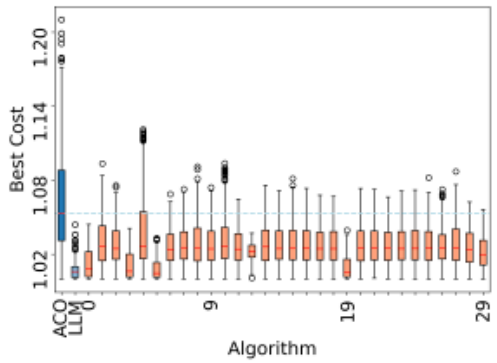
(c) TSP20 - GP-MMAS



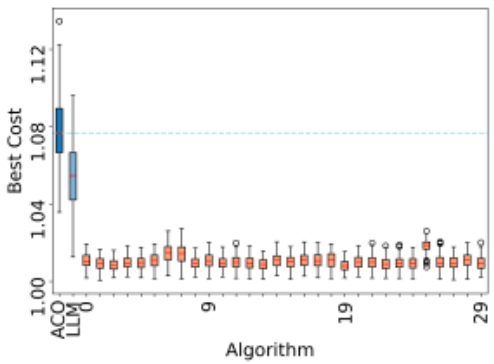
(d) TSP20-100 - GP-AS



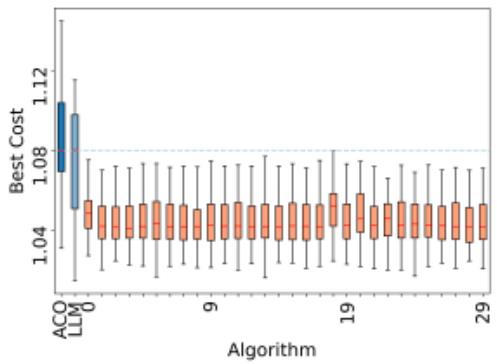
(e) TSP20-100 - GP-ACS



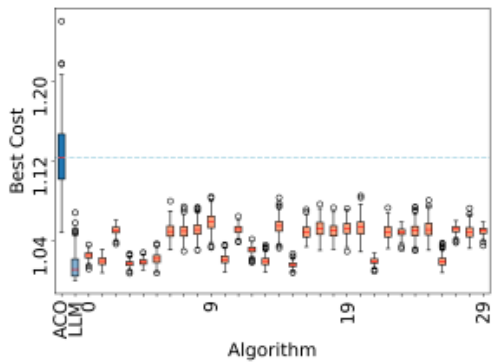
(f) TSP20-100 - GP-MMAS



(g) TSP100 - GP-AS



(h) TSP100 - GP-ACS



(i) TSP100 - GP-MMAS

Experiment 1: To answer Question 1 and Question 2

1. Does GP-ACO exhibit generality when the distributions of training datasets and testing datasets are the same?
2. How do ACO variants influence the learning capabilities of GP-ACO?

Answer 1: The GP-ACO demonstrates generality.

Across three scales of the TSP problem, the trained GP-ACO consistently outperforms the ACO baseline, and this advantage becomes more pronounced as the problem size increases.

Answer 2: Different ACO frameworks do not affect the learning capability of the GP-ACO.

In three variants of the ACO, the GP-ACO significantly surpasses the ACO baseline in performance. Additionally, we discovered that the state transition rules designed in GP-AS and GP-ACS notably outperform those in LLM-ACO, whereas the reverse is true for GP-MMAS.

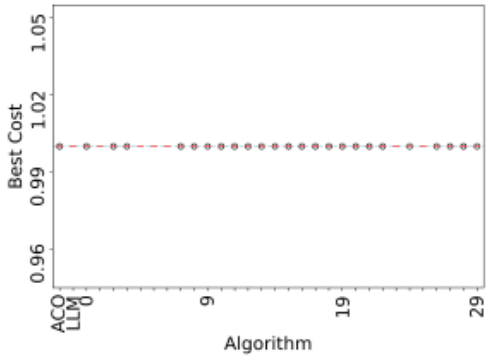
Experiment 2: To answer Question 3

3. Does the incorporation of local search weaken GP-ACO's learning effectiveness?

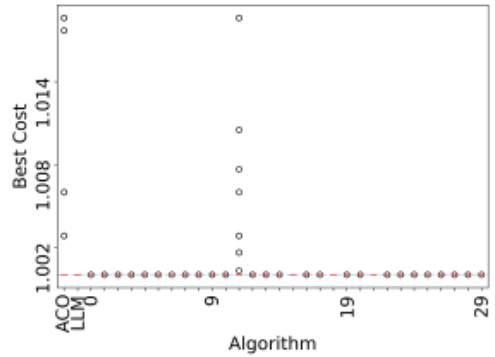
We introduce local search to all the ACO variants in Experiment 1 while keeping the training and testing conditions unchanged.

Table 2: The normalized costs of GP-ACO+2-opt, ACO+2-opt baselines, and LLM-ACO+2-opt on the test datasets.

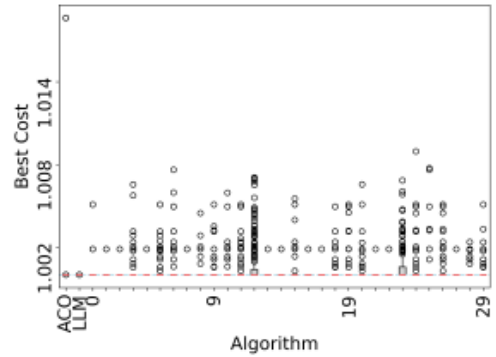
TSP		AS+2-opt			ACS+2-opt			MMAS+2-opt		
		baseline	LLM	GP	baseline	LLM	GP	baseline	LLM	GP
20	mean	1	1	1	1.0003	1	1	1.0002	1	1.0001
		=	=		+	-		+	-	
	std	0	0	0	0	0	0	0	0	0
	min	1	1	1	1	1	1	1	1	1
	max	1	1	1	1.0186	1.0000	1.0186	1.0186	1	1.0090
20-100	mean	1.0018	1.0037	1.0008	1.0021	1.0046	1.0008	1.0007	1.0008	1.0008
		+	+		+	+		-	+	
	std	0	0	0	0	0	0.0001	0	0	0.0001
	min	1	1	1	1	1	1	1	1	1
	max	1.0101	1.0162	1.0452	1.0097	1.0173	1.0123	1.0102	1.0084	1.0495
100	mean	1.0045	1.0091	1.0021	1.0045	1.0096	1.0017	1.0017	1.0011	1.0013
		+	+		+	+		=	-	
	std	0	0	0.0001	0	0	0.0001	0	0	0.0001
	min	1	1	1	1	1	1	1	1	1
	max	1.0184	1.0269	1.0136	1.0211	1.0281	1.0117	1.0112	1.0185	1.0276



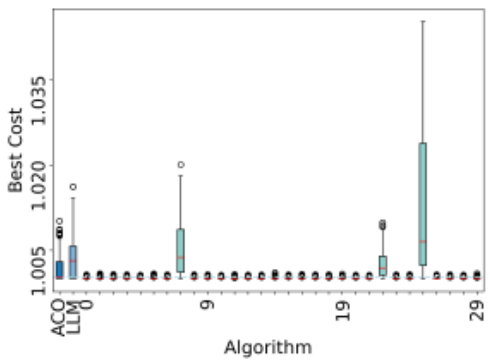
(a) TSP20 - GP-AS+2-opt



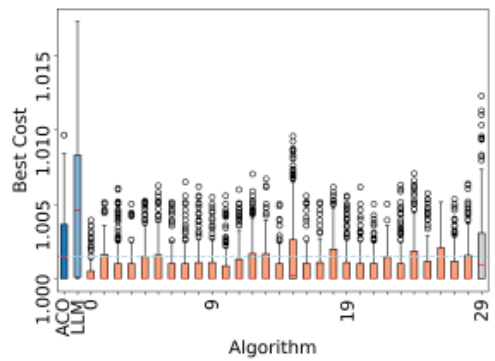
(b) TSP20 - GP-ACS+2-opt



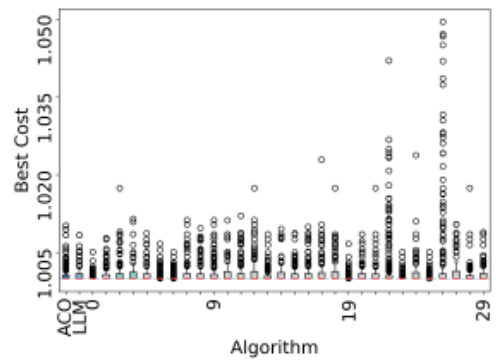
(c) TSP20 - GP-MMAS+2-opt



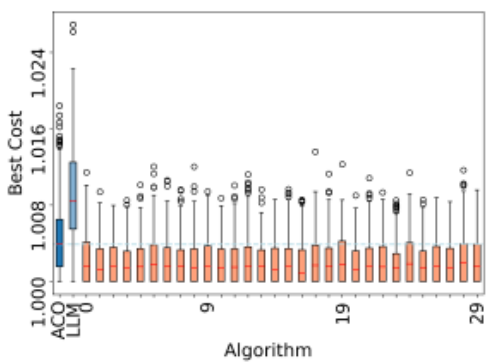
(d) TSP20-100 - GP-AS+2-opt



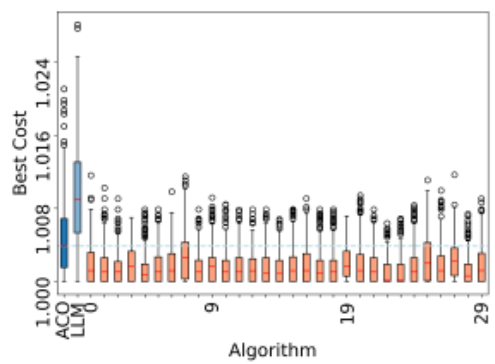
(e) TSP20-100 - GP-ACS+2-opt



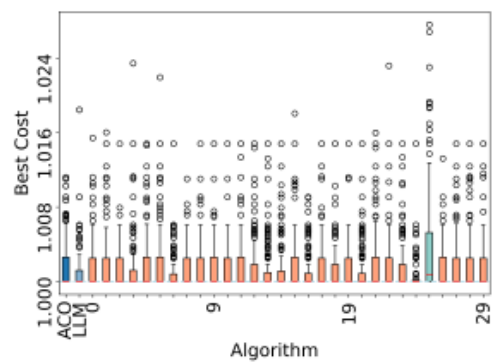
(f) TSP20-100 - GP-MMAS+2-opt



(g) TSP100 - GP-AS+2-opt



(h) TSP100 - GP-ACS+2-opt



(i) TSP100 - GP-MMAS+2-opt

Experiment 2: To answer Question 3

3. Does the incorporation of local search weaken GP-ACO's learning effectiveness?

Answer 3: The introduction of local search does not impact the learning effectiveness of GP-ACO in the GP-AS+2-opt and GP-ACS+2-opt scenarios,
but it diminishes the learning capabilities of GP-ACO in the GP-MMAS+2-opt scenario.

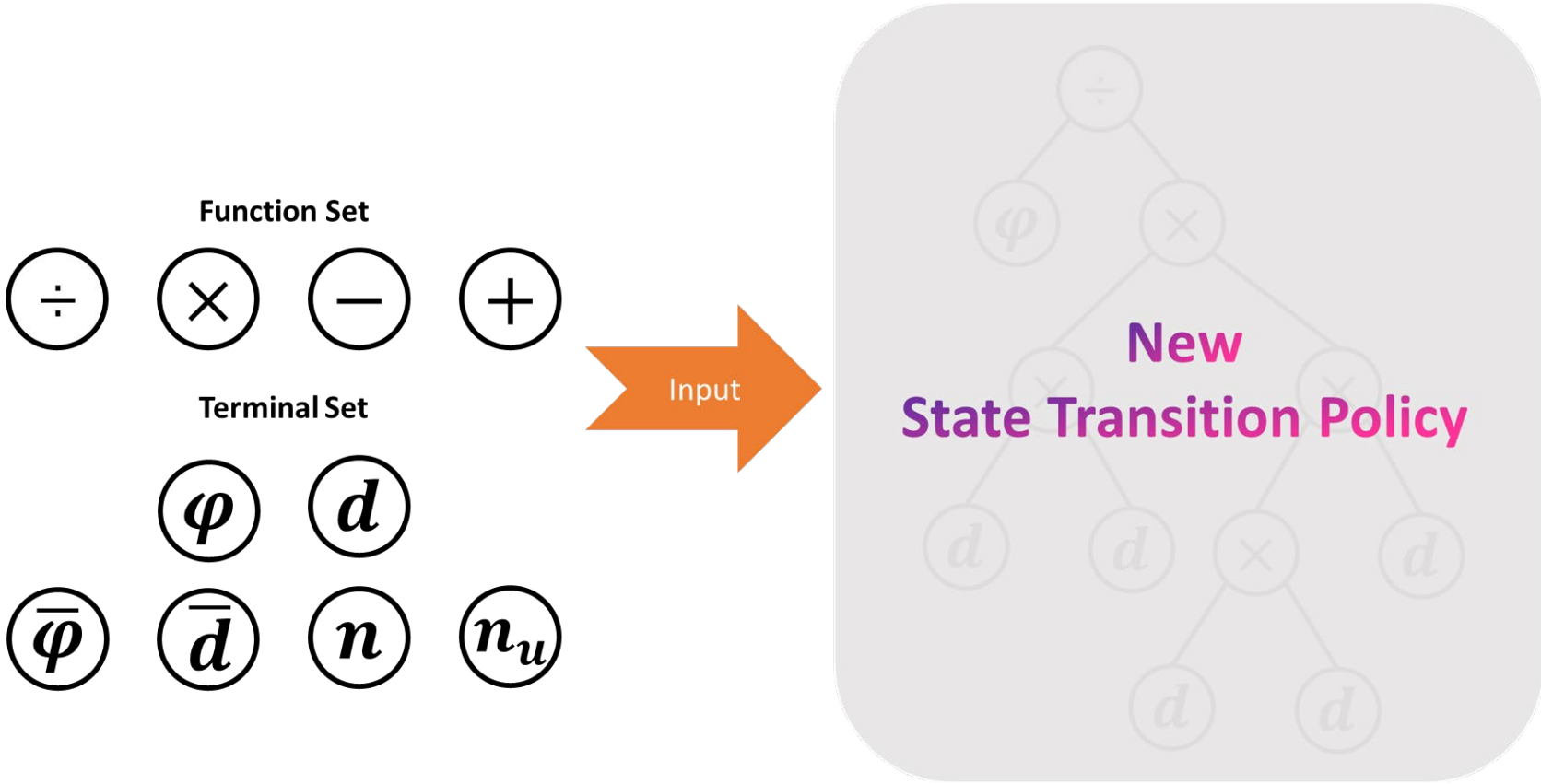
Enhancing Answer 1: GP-ACO+2-opt continues to exhibit generality.
However, in the case of GP-MMAS+2-opt, its performance in generality did not meet expectations.

Enhancing Answer 2: ACO variants that rely more on state transition rules are almost unaffected by the introduction of 2-opt local search regarding the learning capabilities of GP-ACO.
Conversely, ACO variants that depend more heavily on pheromone updates show a reduction in GP-ACO's learning abilities when local search is implemented.

Experiment 3: To answer Question 4

4. Would incorporating additional global information enhance GP-ACO's learning capability?

Building on Experiments 1 and 2, add four terminals containing global information to GP-ACO, which we refer to as xGP-ACO, while maintaining the same training and testing conditions.



average pheromone to the candidate node from the current node
average distance to the candidate node from the current node
total number of nodes
number of candidate nodes

TSP		AS				ACS				MMAS			
		baseline	LLM	GP	xGP	baseline	LLM	GP	xGP	baseline	LLM	GP	xGP
20	mean	1.0028 +	1.0020 +	1.0019 +	**1.0014**	1.0063 +	1.0074 +	**1.0025** +	1.0027	1.0164 +	1.0031 +	1.0039 +	**1.0015**
	std	0	0	0.0001	0.0001	0	0	0.0002	0.0002	0	0	0.0002	0.0002
	min	1	1	1	1	1	1	1	1	1	1	1	1
	max	1.0217	1.0218	1.0201	1.0159	1.0359	1.0604	1.0172	1.0172	1.1327	1.0186	1.0467	1.0403
20-100	mean	1.0364 +	1.0271 +	1.0265 +	**1.0235**	1.0499 +	1.0418 +	1.0306 +	**1.0293**	1.0632 +	1.0076 +	1.0254 +	**1.0029**
	std	0	0	0.0004	0.0004	0	0	0.0004	0.0004	0	0	0.0001	0.0001
	min	1	1	1	1	1.0006	1.0016	1	1	1	1	1	**0.9940**
	max	1.1042	1.0747	1.3813	1.1882	1.1445	1.0831	1.1660	1.1177	1.2101	1.0440	1.1212	1.0579
100	mean	1.0784 +	1.0533 +	1.0105 +	**1.0079**	1.0867 +	1.0721 +	1.0446 +	**1.0440**	1.1248 +	1.0144 +	1.0400 +	**1.0039**
	std	0	0	0.0001	0.0001	0	0	0.0005	0.0005	0	0	0.0001	0.0001
	min	1.0359	1.0127	1.0005	1.0008	1.0310	1.0148	1.0162	1.0156	1.0481	1	1.0071	**0.9881**
	max	1.1346	1.0962	1.0274	1.0194	1.1451	1.1154	1.0799	1.0797	1.2604	1.0680	1.0852	1.0400

TSP		AS+2-opt								ACS+2-opt								MMAS+2-opt							
		baseline		LLM		GP		xGP		baseline		LLM		GP		xGP		baseline		LLM		GP		xGP	
20	mean	**1**	-	**1**	=	**1**	=	**1**		1.0003	+	**1**	=	**1**	=	**1**		1.0002	+	**1**	-	1.0001	-	**1**	
	std	0		0		0		0		0		0		0		0		0		0		0		0	
	min	1		1		1		1		1		1		1		1		1		1		1		1	
	max	1		1		1		1.0063		1.0186		1.0000		1.0186		1		1.0186		1		1.0090		1.0103	
20-100	mean	1.0018	+	1.0037	+	1.0008	+	**1.0002**		1.0021	+	1.0046	+	**1.0008**	+	**1.0008**		**1.0007**	=	1.0008	+	1.0008	+	**1.0007**	
	std	0		0		0		0		0		0		0.0001		0.0001		0		0		0.0001		0.0001	
	min	1		1		1		1		1		1		1		1		1		1		1		1	
	max	1.0101		1.0162		1.0452		1.0053		1.0097		1.0173		1.0123		1.0080		1.0102		1.0084		1.0495		1.0728	
100	mean	1.0045	+	1.0091	+	1.0021	+	**1.0017**		1.0045	+	1.0096	+	**1.0017**	-	1.0018		1.0017	+	**1.0011**	-	1.0013	=	1.0013	
	std	0		0		0.0001		0.0001		0		0		0.0001		0.0001		0		0		0.0001		0.0001	
	min	1		1		1		1		1		1		1		1		1		1		1		1	
	max	1.0184		1.0269		1.0136		1.0142		1.0211		1.0281		1.0117		1.0141		1.0112		1.0185		1.0276		1.0276	

Experiment 3: To answer Question 4

4. Would incorporating additional global information enhance GP-ACO's learning capability?

Answer 4: The introduction of additional global information in xGP-ACO without local search proves highly effective,
significantly enhancing learning capabilities compared to GP-ACO.
With local search, incorporating more global information significantly improves the learning capabilities of GP-AS+2-opt
but weakens those of xGP-MMAS+2-opt, which has less dependency on state transition rules.
Enhancing Answer 2 & 3: After introducing more global information, and excluding the consideration of local search, **variations in ACO frameworks do not reduce the learning capabilities of xGP-ACO.**
However, when the local search is introduced, **the lesser dependence on state transition rules in MMAS variants diminishes the learning capabilities of xGP-ACO+2-opt.**

Experiment 4: To answer Question 5

5. Does GP-ACO remain effective when the distributions of training datasets and testing datasets are different?

We directly use the state transition rules trained by xGP-ACO on the TSP100 dataset from Experiment 3, test on the publicly available TSPLIB dataset.

TSP instances	AS				ACS				MMAS			
	baseline	LLM	xGP	xGP-best	baseline	LLM	xGP	xGP-best	baseline	LLM	xGP	xGP-best
bier127	10.75	**5.52**	6.62	8.83	17.84	6.60	7.83	**6.56**	6.80	2.06	5.45	**0.54**
ch130	9.84	5.57	0.88	**0.75**	20.59	**6.19**	10.11	9.63	7.98	2.70	**0.71**	1.53
d493	21.23	11.31	11.20	**3.15**	32.53	**10.90**	16.25	13.97	11.22	8.42	**4.16**	4.98
eil51	7.57	4.01	1.11	**0.78**	13.20	**5.04**	10.47	9.29	5.21	1.71	1.11	**0.31**
fl417	16.90	14.46	4.26	**1.81**	26.12	15.11	12.68	**12.18**	9.98	9.29	**2.53**	4.83
kroA150	13.76	9.26	3.47	**2.23**	23.55	**10.02**	13.67	11.62	8.83	4.51	1.92	**1.05**
kroB100	7.03	5.46	**1.35**	1.36	16.36	7.41	**5.76**	6.30	7.43	1.42	1.46	**0.26**
kroC100	6.92	4.68	1.25	**1.08**	17.05	**5.49**	7.75	7.22	7.96	1.14	1.39	**0.18**
lin318	17.34	10.21	11.25	**6.40**	26.70	**9.65**	16.51	15.74	10.73	6.50	**3.87**	3.93
pr226	15.00	**7.28**	10.33	9.30	17.58	**9.89**	12.01	10.27	8.74	**2.20**	5.62	2.24
pr264	13.62	8.92	8.62	**7.52**	25.30	**9.30**	12.75	11.49	9.82	5.99	3.81	**3.30**
pr299	20.97	10.36	13.36	**7.92**	33.72	**11.21**	19.04	17.28	12.23	8.13	4.23	**3.69**
pr439	23.52	**10.94**	21.67	16.86	36.07	**13.35**	19.71	16.19	12.27	7.31	7.53	**4.45**
rat99	9.00	6.50	1.18	**1.01**	15.58	**8.60**	10.54	10.76	8.49	3.61	**1.85**	1.97
ts225	9.25	**3.70**	14.82	19.71	23.17	**3.74**	8.91	5.56	12.08	3.10	8.71	**0.70**
Avg. opt. gap	13.51	7.88	7.43	**5.91**	23.02	**8.83**	12.27	10.93	9.32	4.54	3.62	**2.26**

TSP instances	AS+2opt				ACS+2opt				MMAS+2opt			
	baseline	LLM	xGP	xGP-best	baseline	LLM	xGP	xGP-best	baseline	LLM	xGP	xGP-best
bier127	0.85	0.74	0.99	**0.51**	1.26	0.71	0.38	**0.26**	3.60	0.07	2.05	**0.03**
ch130	0.94	1.25	**0.03**	0.25	1.89	1.40	**0.94**	1.01	4.00	0.22	0.20	**0.14**
d493	2.30	2.60	1.82	**1.42**	3.30	2.25	2.07	**1.82**	7.96	1.12	4.41	**0.58**
eil51	0.95	0.90	**0.39**	0.68	1.50	0.94	0.72	**0.71**	4.38	0.70	**0.41**	0.69
fl417	1.13	1.67	**0.35**	0.83	1.74	1.62	1.07	**1.05**	2.60	0.59	5.49	**0.52**
kroA150	1.30	1.81	**0.52**	0.69	2.04	2.11	0.67	**0.56**	4.60	0.21	1.41	**0.04**
kroB100	0.52	0.65	0.27	**0.26**	0.81	0.40	0.18	**0.21**	3.06	0.09	0.25	**0.08**
kroC100	0.28	0.67	0.22	**0.008**	0.65	0.81	**0.04**	**0.04**	2.56	0.05	0.26	**0.03**
lin318	2.38	2.41	**1.26**	1.72	3.29	2.06	1.65	**1.03**	5.74	0.58	5.74	**0.45**
pr226	0.40	0.86	0.55	**0.26**	0.89	1.03	**0.51**	0.52	1.14	0.03	8.63	**0.007**
pr264	1.04	0.73	0.89	**0.21**	1.43	2.49	0.27	**0.16**	5.45	**0.007**	5.49	0.07
pr299	1.62	1.85	1.66	**1.22**	2.43	1.96	1.18	**1.02**	7.60	0.33	6.35	**0.18**
pr439	1.49	1.84	3.50	**1.26**	2.20	1.63	1.34	**1.00**	7.17	0.55	9.36	**0.22**
rat99	1.04	1.44	**0.38**	0.71	2.09	1.16	**0.74**	0.75	7.35	0.72	0.46	**0.69**
ts225	0.54	0.74	3.29	**0.48**	0.72	1.37	0.32	**0.29**	3.34	0.33	7.61	**0.002**
Avg. opt. gap	1.12	1.34	1.08	**0.70**	1.75	1.46	0.80	**0.70**	4.70	0.37	3.87	**0.25**

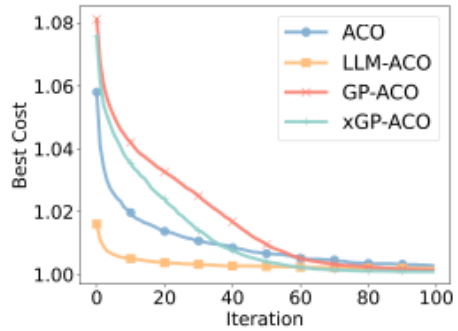
Experiment 4: To answer Question 5

5. Does GP-ACO remain effective when the distributions of training datasets and testing datasets are different?

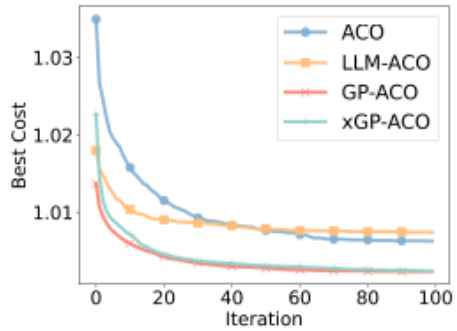
Answer 5: **The state transition rules learned by xGP-ACO and xGP-ACO+2-opt demonstrate strong performance across maps of varying scales and distributions.** **This indicates that the generality capabilities of xGP-ACO extend beyond training scenarios, adapting effectively to diverse and larger-scale environments.**

PART IV

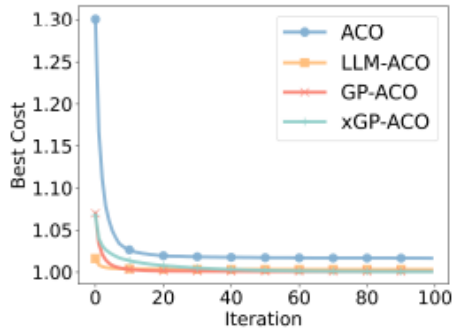
Further Analysis



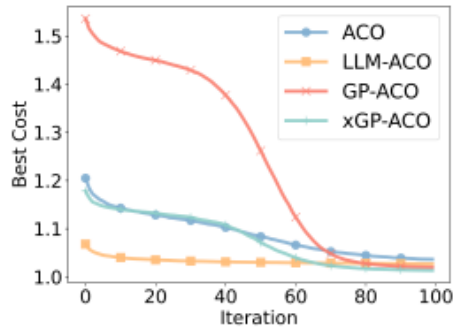
(a) TSP20-AS



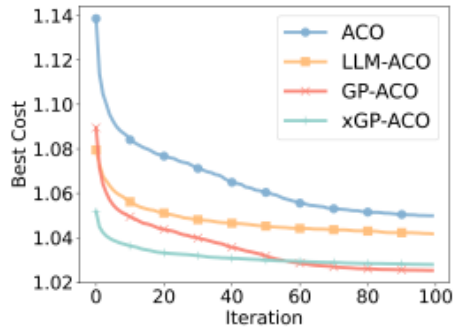
(b) TSP20-ACS



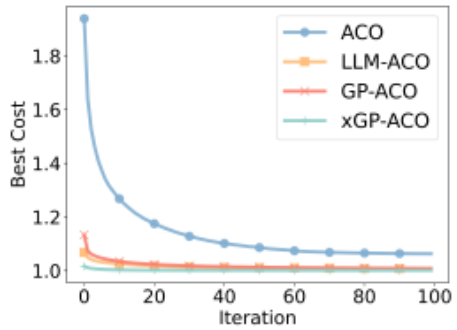
(c) TSP20-MMAS



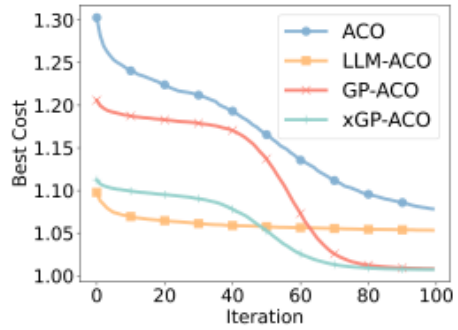
(d) TSP20-100-AS



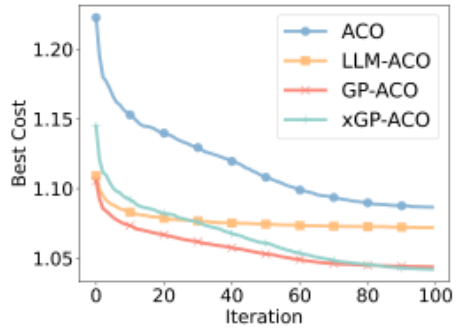
(e) TSP20-100-ACS



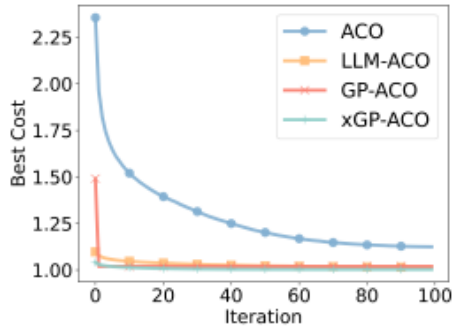
(f) TSP20-100-MMAS



(g) TSP100-AS

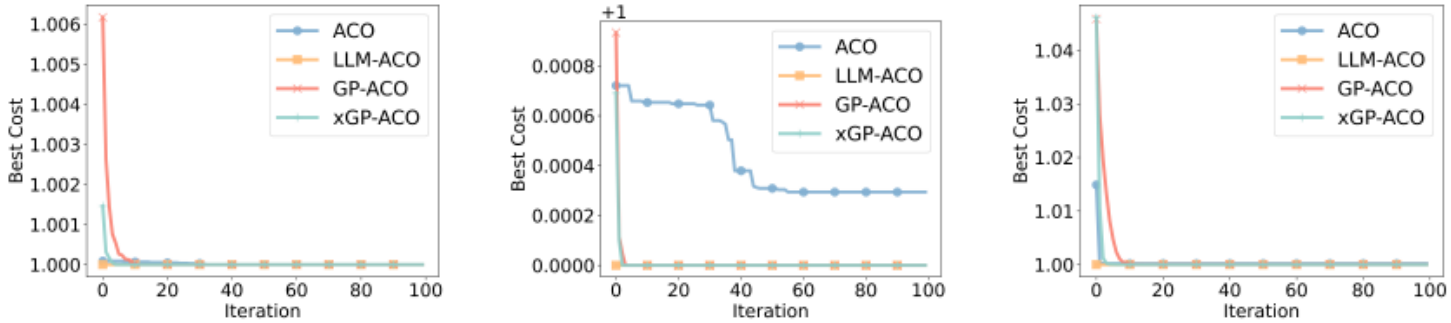


(h) TSP100-ACS



(i) TSP100-MMAS

Fig. 7: Convergence of ACO baseline, LLM-ACO, GP-ACO and xGP-ACO



(a) TSP20-AS+2-opt (b) TSP20-ACS+2-opt (c) TSP20-MMAS+2-opt

(d) TSP20-100-AS+2-opt (e) TSP20-100-ACS+2-opt (f) TSP20-100-MMAS+2-opt

(g) TSP100-AS+2-opt (h) TSP100-ACS+2-opt (i) TSP100-MMAS+2-opt

Fig. 8: Convergence of ACO+2-opt baseline, LLM-ACO+2-opt, GP-ACO+2-opt and xGP-ACO+2-opt

Table 7: New State Transition rules learned by GP-ACO.

Algorithm	TSP20	TSP20-100	TSP100
GP-AS	$\frac{\varphi}{(d-2\varphi)d^2} - \varphi$	$\frac{\varphi}{(\varphi+d^4)d^5}$	$\frac{2\varphi}{(2\varphi+d^2)(2\varphi+d^4)d^3}$
GP-ACS	$\varphi d^3 - \frac{d^3}{\varphi} + 1$	$\frac{(d^3-1)\varphi}{(\varphi-d)d^4} - d$	$\frac{2\varphi}{(\varphi+d)^2 d} - 2d^2 + 1$
GP-MMAS	$-\varphi d - \frac{2d^2}{\varphi+d}$	$-\frac{2\varphi}{(\varphi+\varphi d+2d^2)d}$	$\frac{\varphi+\varphi(\varphi+d)d^2}{(\varphi+d)^2 d^4}$

Table 8: New State Transition rules learned by xGP-ACO.

Local Search	**xGP-AS**	**xGP-ACS**	**xGP-MMAS**
None	$\frac{n^2\varphi^2\bar{d}^2}{\varphi d\bar{d}+d^2}$	$\left(\frac{\bar{d}}{d}\right)^3 - \frac{\varphi}{d^2}$	$\frac{2\bar{\varphi}+\varphi d}{\bar{\varphi}^4+d^8}$
2-opt	$\frac{n^3\varphi^3\bar{d}^2}{(n\varphi+1)(n+\varphi)d}$	$\frac{\varphi^2\bar{d}}{d^2}$	$-\varphi^2 + (n - n_u - 2)\varphi - n_u - \bar{\varphi}$

To answer Question 6

6. How interpretable is GP-ACO?

Answer 6: GP-ACO and xGP-ACO exhibit excellent interpretability.

PART IV

Conclusion & Future Works

The primary goals of this paper are to answer four research questions that were formulated concerning

1. **generality,**
2. **the impact of different ACO variants on GP-ACO,**
3. **the influence of the 2-opt local search on GP-ACO,**
4. **and the effects of incorporating more global information into GP-ACO.**

The primary goals of this paper are to answer four research questions that were formulated concerning

1. generality,
2. the impact of different ACO variants on GP-ACO,
3. the influence of the 2-opt local search on GP-ACO,
4. and the effects of incorporating more global information into GP-ACO.

From the experimental results, we observe that:

1. GP-ACO exhibits robust generality and performs well across problems of different scales and distributions;
2. Different ACO variants do not affect the learning capability of GP-ACO;
3. The introduction of 2-opt local search diminishes the learning capabilities of GP-ACO in the MMAS+2-opt variant, which has a reduced dependency on state transition rules;
4. Incorporating additional global information (xGP-ACO) significantly enhances the learning capabilities of GP-ACO;
5. GP-ACO and xGP-ACO show excellent interpretability.

Add
Matrix Operators
into function sets

Thank you for your listening!

Bocheng Lin 2024.6.21

VICTORIA UNIVERSITY OF WELLINGTON
TE HERENGA WAKA
Evolutionary Computation Research Group