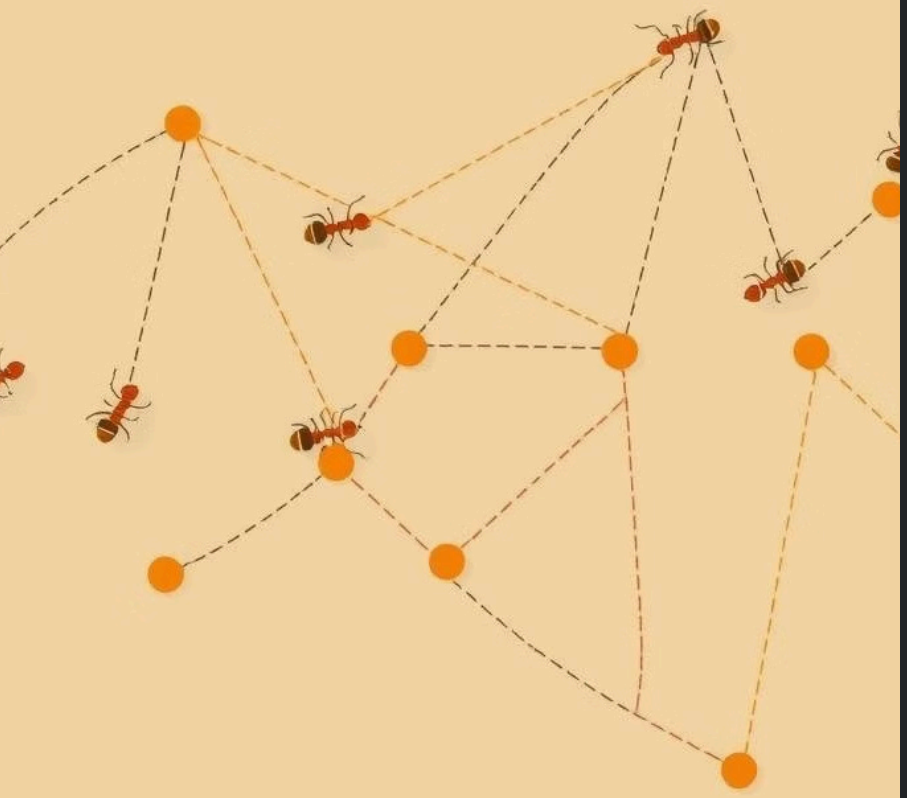


# Ant Colony Optimization

This presentation explores the implementation and analysis of an Ant Colony Optimization (ACO) algorithm applied to the Traveling Salesperson Problem (TSP). The TSP is a classic NP-hard combinatorial optimization challenge with both theoretical and practical significance. Our study investigates ACO's performance on randomly generated symmetric TSP instances of 10 and 20 cities, focusing on how key parameters affect solution quality and convergence.

Using Python, we simulate probabilistic tour construction, pheromone evaporation, and deposition, highlighting the balance between exploration and exploitation in finding near-optimal routes.

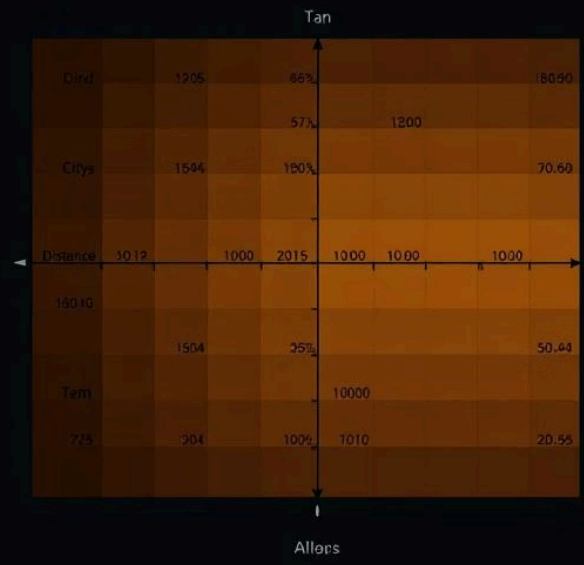
# Problem Representation and Initialization

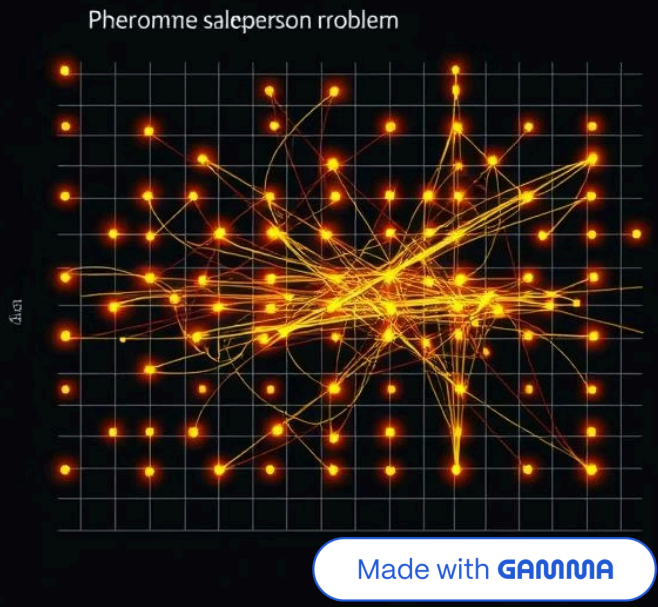
## Distance Matrix (D)

A symmetric N x N matrix representing distances between cities, with diagonal elements zero and off-diagonal random integers between 3 and 50. A fixed seed ensures reproducibility.

## Pheromone Matrix (T)

Stores pheromone levels on edges between cities, initialized with small positive values for non-diagonal elements. It evolves during iterations to guide ants.



# Ant Colony Optimization Algorithm

1

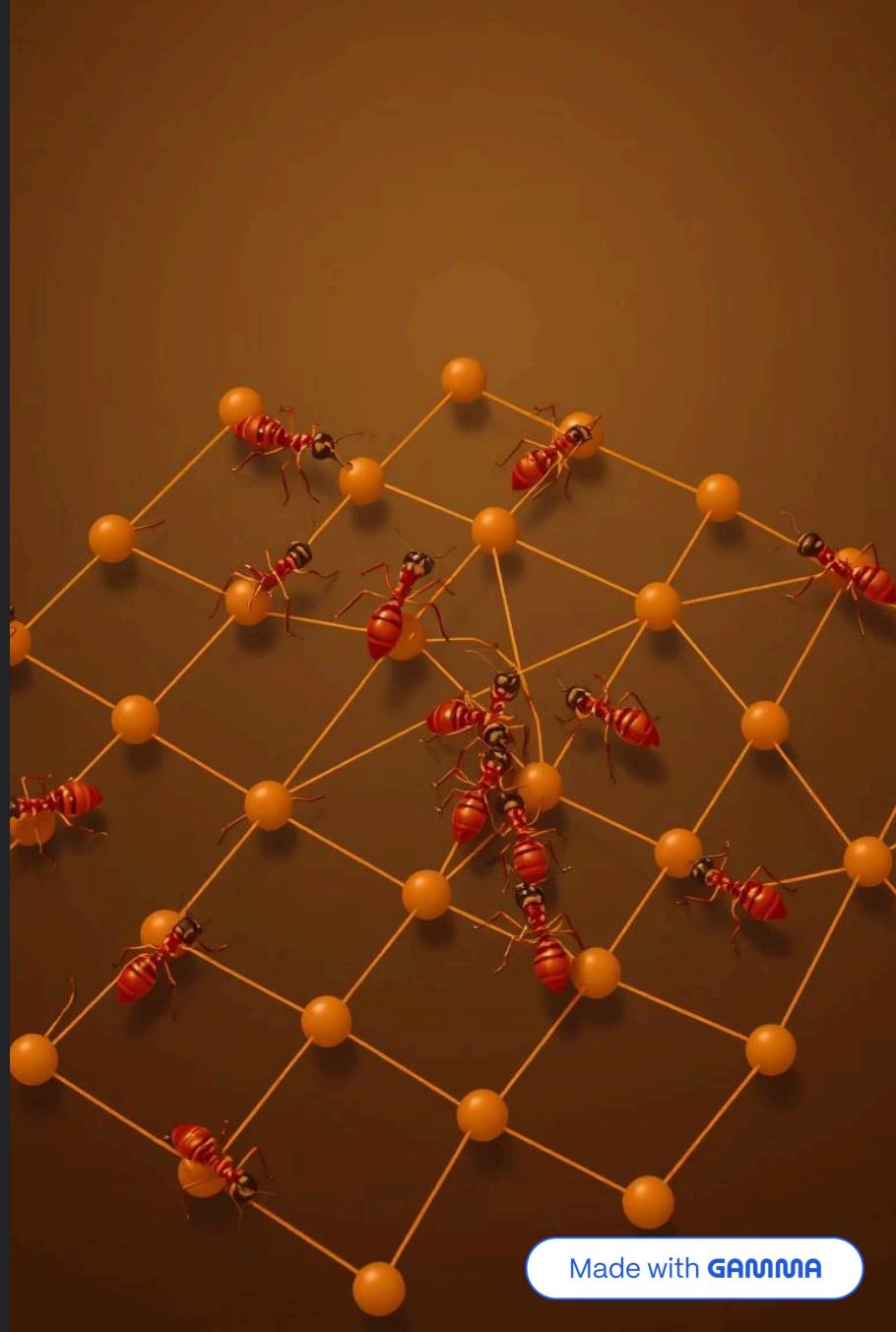
## Ant Solution Construction

Each ant starts at city 0 and probabilistically selects the next city based on pheromone intensity and heuristic information (inverse distance), building a complete tour.

2

## Pheromone Update

After tours, pheromone trails evaporate to avoid premature convergence, then ants deposit pheromones proportional to solution quality, reinforcing shorter paths.



# Parameter Settings and Experimental Design

## Adaptive Parameters

- Small ants (≤3): Higher heuristic influence (β), favors greedy choices.
- Moderate ants (4-10): Balanced pheromone (α) and heuristic (β) influence.
- Large ants (>10): Increased pheromone influence to leverage collective intelligence.

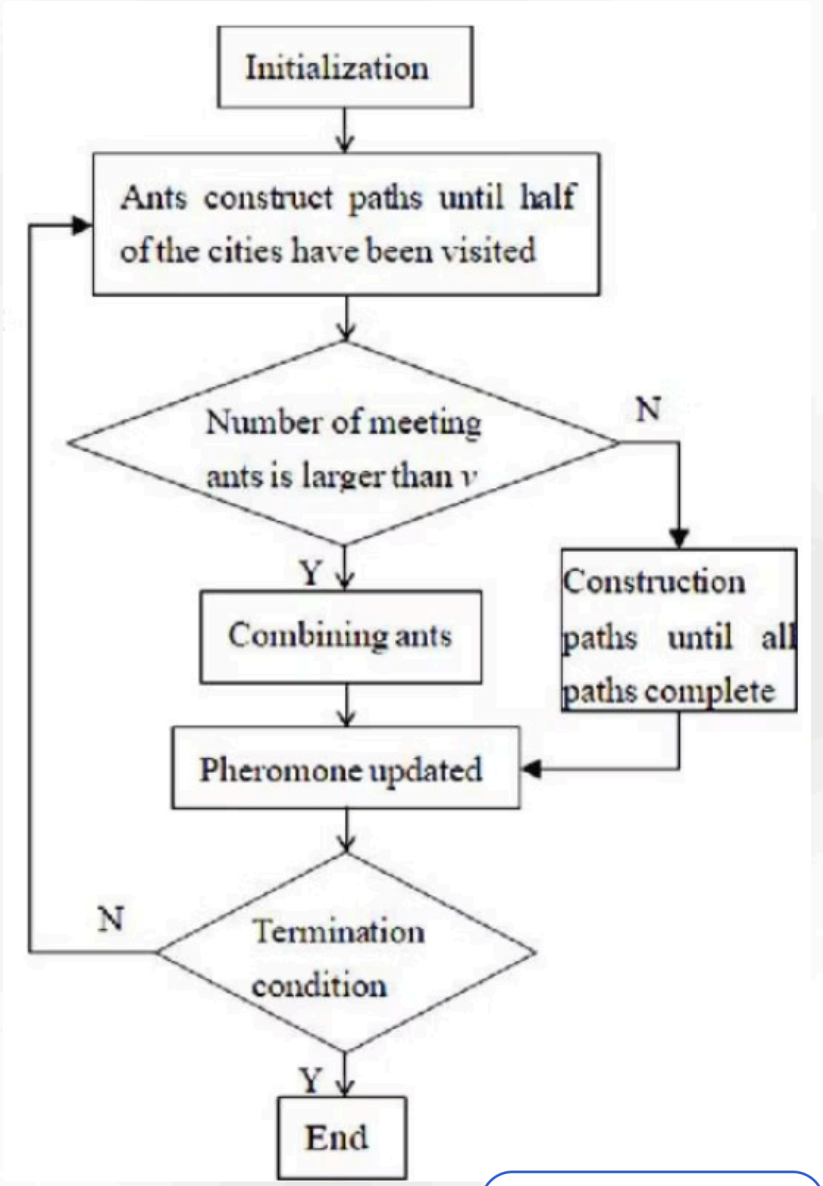
## Experiments

- 10 cities: Ant counts of 1, 5, 10, 20; 8 iterations each.
- 20 cities: Same ant counts; 15 iterations each.

# Matrix Visualization and Debugging

Pheromone matrix in a formatted manner, aiding in understanding matrix states during algorithm execution. This visualization supports debugging and analysis of pheromone evolution and distance relationships.

Such matrix views help track how pheromone intensities change over iterations, reflecting the algorithm's learning and convergence process.



# Results: Output Structure and Data

1. ## Per-Iteration Details

   Tables show paths and distances for each ant per iteration, highlighting solution progress.

2. ## Best This Iteration

   Shortest path and distance found in each iteration are recorded.

3. ## Overall Best Path

   The globally best path and distance across all iterations are tracked for each ant count configuration.

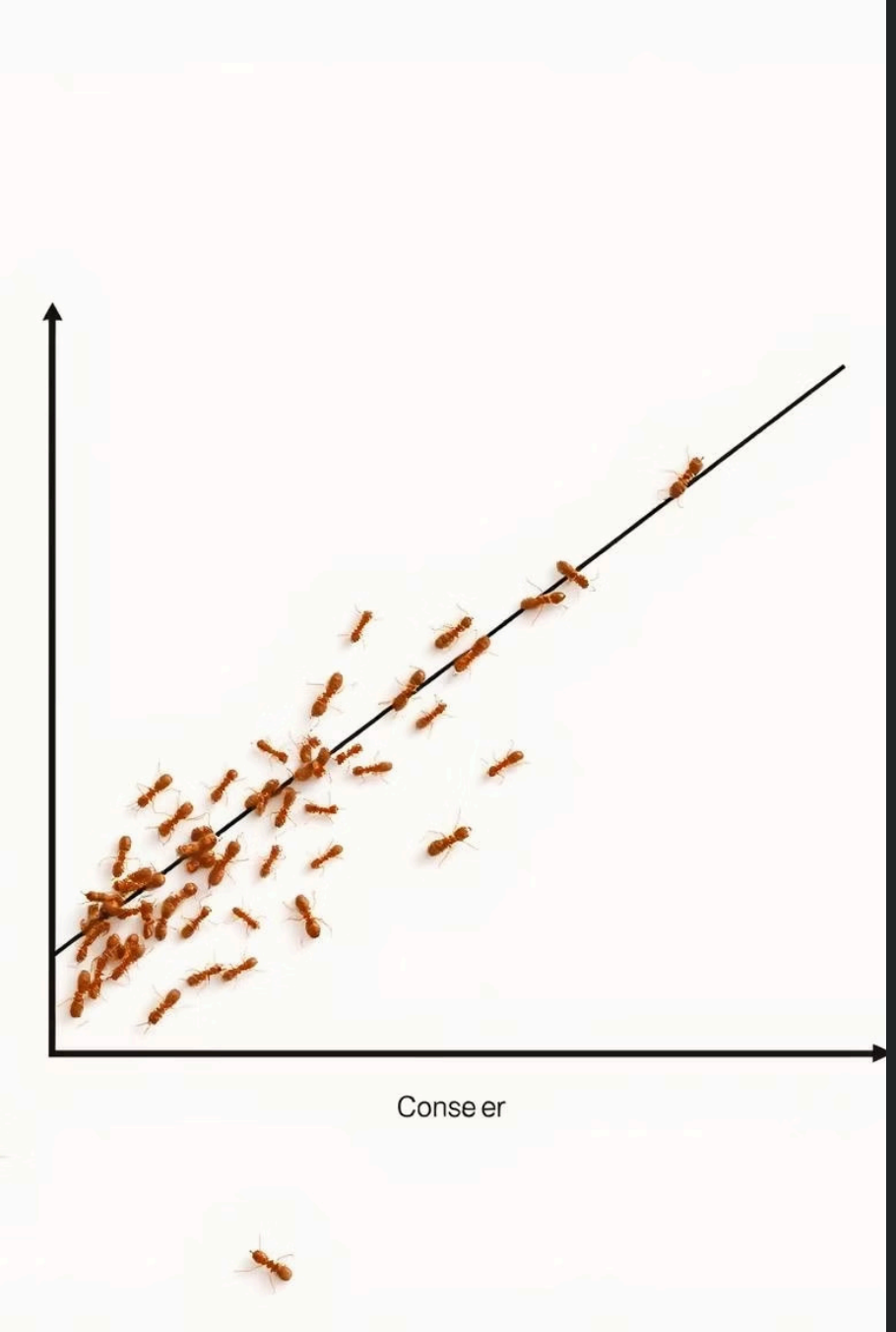
# Qualitative Observations: Number of Ants and Problem Size

## Number of Ants

- Few ants limit exploration, rely on heuristics, may converge slowly.
- Moderate ants improve path diversity and solution quality.
- Large ants refine solutions but increase computational cost.

## Problem Size

- 20-city problem has a much larger search space than 10-city.
- More iterations needed for convergence on larger problems.
- Quality relative to optimum is harder to achieve with fixed budget.




# Convergence and Adaptive Parameter Strategy

## Convergence Behavior

Early iterations show diverse paths and varying distances; pheromone reinforcement guides ants to shorter tours over time.

## Adaptive α and β

Adjusting pheromone and heuristic influence balances exploration and exploitation, improving solution quality depending on ant population size.

This adaptive approach ensures the algorithm remains effective across different problem sizes and colony configurations, leveraging collective intelligence while maintaining sufficient exploration.