This Jupyter notebook implements an **Ant Colony Optimization (ACO)** algorithm to solve a **knapsack problem** for backpack packing. Here's a comprehensive explanation:

**Problem Overview**

The notebook solves a backpack optimization problem where you need to select items to pack within a 20kg weight limit while maximizing value.

**PEAS Framework**

The selected line "PEAS" refers to the AI agent design framework:

* **Performance Measure**: Maximize total value without exceeding 20kg
* **Environment**: Backpack with 20kg capacity, 8 item types with weights/values
* **Actuator**: Packing items into the backpack
* **Sensor**: Detecting item weights, values, and current bag weight

A screenshot of a computer

AI-generated content may be incorrect.

**Key Components**

**1. Initial Setup**

* Defines 8 items with weights and values (Tent, Sleeping Bag, etc.)
* Sets backpack capacity to 20kg
* Imports necessary libraries (numpy, matplotlib, sys, time)

A screenshot of a computer code

AI-generated content may be incorrect.

**ACO Algorithm Implementation**

**Cost Matrix Function:**

A white background with text

AI-generated content may be incorrect.

**Transition Matrix:**

* Calculates ant movement probabilities using pheromone levels and heuristic information
* Uses alpha (pheromone importance) and beta (heuristic importance) parameters

A computer screen shot of a computer code

AI-generated content may be incorrect.

**Pheromone Update:**

* Evaporates existing pheromones
* Reinforces paths taken by ants based on solution quality
* Prints pheromone levels for tracking

A screen shot of a computer code

AI-generated content may be incorrect.

**3. Main ACO Algorithm**

The aco\_algorithm function:

* Initializes pheromone levels
* Runs multiple iterations with multiple ants
* Each ant constructs a solution by selecting items based on transition probabilities
* Updates pheromones based on solution quality
* Tracks best solution and convergence

**4. Two Scenarios**

The notebook runs ACO with different parameters:

* **Scenario 1**: 3 ants, α=0.5, β=3
* **Scenario 2**: 5 ants, α=0.5, β=7

**5. Dynamic Input**

Allows users to:

* Use default items
* Enter custom items manually

A screenshot of a computer program

AI-generated content may be incorrect.

**### 4.  Calling the search algorithms**

**Algorithm Flow**

1. Initialize pheromone trails
2. For each iteration:
   * Each ant constructs a solution probabilistically
   * Evaluate solution quality
   * Update pheromones based on results
3. Track best solution found
4. Compare different parameter settings

A screenshot of a computer program

AI-generated content may be incorrect.

Final Pheromones levels after doing 100 iteration and selecting the best solution items

A screenshot of a computer program

AI-generated content may be incorrect.

**Output**

A screenshot of a computer program

AI-generated content may be incorrect.

**6. Visualization**

* **Convergence plots**: Show how solution quality improves over iterations
* **Pheromone comparison**: Bar charts comparing final pheromone levels
* **Pheromone evolution**: Heatmaps showing how pheromones change over time

A graph with numbers and lines

AI-generated content may be incorrect.

A screenshot of a graph

AI-generated content may be incorrect.

**7. Performance Analysis**

Measures and compares:

* Execution time
* Memory usage
* Time complexity: O(iterations × ants × items²)
* Space complexity: O(items + ants × items)

A screenshot of a computer code

AI-generated content may be incorrect.