Exp-2.13

Title:

Assignment Problem Using Exhaustive Search

Aim:

To find the optimal assignment of workers to tasks minimizing the total cost by generating all permutations of assignments and evaluating their total costs.

Procedure:

1. Define a function total_cost(assignment, cost_matrix) to compute total cost for a specific assignment.
2. Define a function assignment_problem(cost_matrix) that:
    - Generates all permutations of task indices representing all possible worker-task assignments.
    - For each permutation, calculate the total cost using total_cost.
    - Track the minimum total cost and the corresponding assignment.
3. Return the optimal assignment and its total cost.
4. Test on given cost matrices and print the optimal assignment and cost.

Algorithm:

1. For each permutation of tasks (one task per worker):
    - Compute total cost summing cost[i][assignment[i]] for all workers i.
2. Maintain minimum cost and track assignment.
3. After processing all permutations, return optimal assignment and minimum cost.

Input:

[3, 10, 7], [8, 5, 12], [4, 6, 9]

[15, 9, 4], [8, 7, 18], [6, 12, 11]

Output:

Test Case 1:

Optimal Assignment: [(1, 1), (2, 2), (3, 3)]

Total Cost: 17

Test Case 2:

Optimal Assignment: [(1, 3), (2, 1), (3, 2)]

Total Cost: 24

Program:

```
import itertools
def total_cost(assignment, cost_matrix):
    return sum(cost_matrix[i][assignment[i]] for i in range(len(assignment)))
def assignment_problem(cost_matrix):
    n = len(cost_matrix)
    tasks = list(range(n))
    min_cost = float('inf')
    optimal_assignment = []
    for perm in itertools.permutations(tasks):
        current_cost = total_cost(perm, cost_matrix)
        if current_cost < min_cost:
            min_cost = current_cost
            optimal_assignment = perm
```

```
    assignment = [(worker + 1, task + 1) for worker, task in
enumerate(optimal_assignment)]
    return assignment, min_cost
cost_matrix1 = [
    [3, 10, 7],
    [8, 5, 12],
    [4, 6, 9]
]
assignment1, cost1 = assignment_problem(cost_matrix1)
print("Test Case 1:")
print("Optimal Assignment:", assignment1)
print("Total Cost:", cost1)
cost_matrix2 = [
    [15, 9, 4],
    [8, 7, 18],
    [6, 12, 11]
]
assignment2, cost2 = assignment_problem(cost_matrix2)
print("\nTest Case 2:")
print("Optimal Assignment:", assignment2)
print("Total Cost:", cost2)
```

Performance Analysis:

Time Complexity: O(n!)

Space Complexity: O(n)

Program Output:

exp2.13.py - C:/Users/KABILAN/Documents/daa docs/exp2.13.py (3.11.8)

File Edit Format Run Options Window Help

```python
import itertools

def total_cost(assignment, cost_matrix):
    return sum(cost_matrix[i][assignment[i]] for i in range(len(assignment)))

def assignment_problem(cost_matrix):
    n = len(cost_matrix)
    tasks = list(range(n))
    min_cost = float('inf')
    optimal_assignment = []

    for perm in itertools.permutations(tasks):
        current_cost = total_cost(perm, cost_matrix)
        if current_cost < min_cost:
            min_cost = current_cost
            optimal_assignment = perm


    assignment = [(worker + 1, task + 1) for worker, task in enumerate(optimal_assignment)]
    return assignment, min_cost

cost_matrix1 = [
    [3, 10, 7],
    [8, 5, 12],
    [4, 6, 9]
]

assignment1, cost1 = assignment_problem(cost_matrix1)
print("Test Case 1:")
print("Optimal Assignment:", assignment1)
print("Total Cost:", cost1)

cost_matrix2 = [
    [15, 9, 4],
    [8, 7, 18],
    [6, 12, 11]
]

assignment2, cost2 = assignment_problem(cost_matrix2)
print("\nTest Case 2:")
print("Optimal Assignment:", assignment2)
print("Total Cost:", cost2)
```

IDLE Shell 3.11.8

File Edit Shell Debug Options Window Help

```
Python 3.11.8 (tags/v3.11.8:db85d51, Feb  6 2024, 22:03:32) [MSC v.1937 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/KABILAN/Documents/daa docs/exp2.13.py
Test Case 1:
Optimal Assignment: [(1, 3), (2, 2), (3, 1)]
Total Cost: 16

Test Case 2:
Optimal Assignment: [(1, 3), (2, 2), (3, 1)]
Total Cost: 17
>>>
```

Ln: 12 Col: 0

Result:

The exhaustive search assignment program runs successfully and finds the optimal assignments with minimal costs.