



ANT COLONY



OPTIMIZATION

(ACO)







Description

**ANTS LEAVE PHEROMONES ON THE PATH
THEY TRAVELS. MAKING EACH PATH HAS 2
VALUES : PHEROMONE, COST (DISTANCE)**



Description (2)

- 1.** Ants travel to every node, changing pheromone of the paths they take
 - 2.** Best paths end up having the highest pheromones
- 
- 

Formulas

$$\Delta\tau_{i,j}^k = \begin{cases} \frac{1}{L_k} & k^{\text{th}} \text{ ant travels on the edge } i,j \\ 0 & \text{otherwise} \end{cases}$$



$$\tau_{i,j}^k = (1 - \rho) \tau_{i,j} + \sum_{k=1}^m \Delta\tau_{i,j}^k \quad \text{With vaporization}$$

Sequential (Travel loop)




```
void travel(){
    // Loop for each ant
    for(int i = 0; i < totalAnts; i++){
        // current node (starts at node 0)
        int currNode = 0;
        bool nextNodeExist = true;
        // visited nodes
        vector<bool> visited(totalNodes, 0);
        float totalCost = 0; // Lk
        vector<pair<int, int>> traveledThrough;
        visited[0] = true;

        // loop until traveled every node
        for(int j = 0; j < totalNodes-1; j++){
            nextNodeExist = true;
            int nextNode = getNextNode(visited, currNode);
            if(nextNode < 0){
                nextNodeExist = false;
                break;
            }
        }
    }
}
```

**Loop for n ant,
each ant travels m nodes**

Parallel (Travel loop)

An illustration of a large black ant with four smaller black ants nearby, all on a light green background with stylized leaves.

```
// Ant kth travel thorough every node
if(gidx < totalAnts){
    for(int i = 0; i < totalNodes - 1; i++){
        // Calculate for next node
        // Getting totalProbs
        nextNodeExist = false;
        float probs[TOTAL_NODE];
        float totalProbs = 0;
        for(int l = 0; l < totalNodes; l++) probs[l] = 0;

        for(int j = 0; j < totalNodes; j++){
            if(!visited[j] && map[currNode*totalNodes+j].cost > 0){ // connects
                nextNodeExist = true; // probs is not all 0
                float Tij = map[currNode*totalNodes+j].pheromone;
                float nij = 1.0 / map[currNode*totalNodes+j].cost;
                probs[j] = Tij * nij;
                totalProbs += Tij * nij;
            }
        }
    }
}
```

**Call totalAnts number of threads,
each thread represents kth ant**

Sequential (Updating pheromones)

```
// update pheromone (every update old pheromone evaporates by evaRate)
void updatePheromones(){
    for(int i = 0; i < totalNodes; i++){
        for(int j = 0; j < totalNodes; j++){
            float oldPheromone = map[i][j].pheromone;
            map[i][j].pheromone = ((1-evaRate) * oldPheromone) + delta[i][j];
        }
    }
}
```



Sequential matrix operation

Parallel (Updating pheromones)

```
__global__ void updatePheromones(int totalNodes, float evaRate, edges* map, float* delta){  
    const int col = blockDim.x * blockIdx.x + threadIdx.x;  
    const int row = blockDim.y * blockIdx.y + threadIdx.y;  
    const int gidx = row*totalNodes + col;  
  
    if(col < totalNodes && row < totalNodes){  
        map[gidx].pheromone = (1-evaRate) * map[gidx].pheromone + delta[gidx];  
    }  
}
```



Parallelized matrix operation

Time comparison

(100 nodes, 60 ants, 20 epochs)



1. Sequential time

```
$ nvcc acoSequential.cu -o acoSequential && ./acoSequential.exe  
DONE EPOCH 11  
DONE EPOCH 12  
DONE EPOCH 13  
DONE EPOCH 14  
DONE EPOCH 15  
DONE EPOCH 16  
DONE EPOCH 17  
DONE EPOCH 18  
DONE EPOCH 19  
Time taken by function: 854405 microseconds
```

2. Parallel time

```
$ nvcc acoParallel.cu -o acoParallel && ./acoParallel  
DONE EPOCH 11  
DONE EPOCH 12  
DONE EPOCH 13  
DONE EPOCH 14  
DONE EPOCH 15  
DONE EPOCH 16  
DONE EPOCH 17  
DONE EPOCH 18  
DONE EPOCH 19  
Time taken by function: 226995 microseconds
```

Formulas



$$S = \frac{1}{(1 - P) + \frac{P}{N}}$$

(NVIDIA, 2020a)

(Travel)

$$S = 1 / ((1 - 0.98) + (0.98 / 60))$$

$$= 27.5$$

(Pheromone update)

$$S = 1 / ((1 - 0.02) + (0.02 / 100 * 100))$$

$$= 1.02$$

Possible improvements

- ***Each thread (ant) use a lot of memory***
- ***Getting next node might be parallelizable***
- ***Can be overall faster, based on the formula***

The background is a light green watercolor wash. In the four corners, there are detailed illustrations of green leaves and branches, some with fine outlines and others with more blended, painterly edges.

Thank You