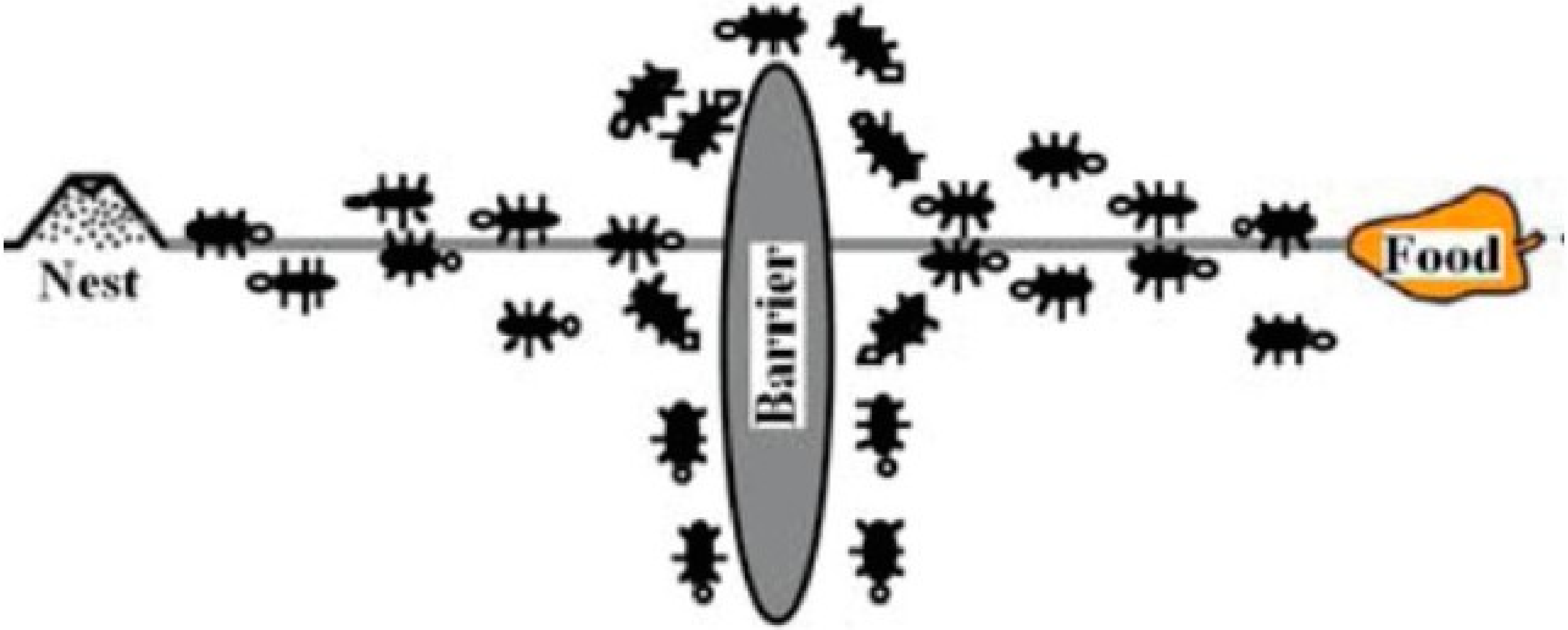
Ant Algorithm


Nest
Barrier
Food

Initialize the pheromone matrix τ for each pair of cities
Place the m ants on n random cities
for $t = 1$ to nc **do**
    for $i = 1$ to n **do**
        for $k = 1$ to m **do**
            Choose next city j according to the transition rule
    for $k = 1$ to m **do**
        Calculate tour distance L_k for ant k
        if an improved tour is found **then**
            Update T^* and L^*
    Update the pheromone matrix τ

Initialize T^*_{Global} {this data is shared, everything else is private}
parallel region with $nColonies$ **threads**
    Initialize the pheromone matrix τ for each pair of cities
    Place the m ants on n random cities
    for $t = 1$ to nc **do**
        for $i = 1$ to n **do**
            for $k = 1$ to m **do**
                Choose next city j according to the transition rule
        for $k = 1$ to m **do**
            Calculate tour distance L_k for ant k
            if an improved tour is found **then**
                Update T^* and L^*
            if this is an exchange cycle **then**
                if $L^* < L^*_{Global}$ **then**
                    *****Critical section*****
                    if $L^* < L^*_{Global}$ **then**
                        $T^*_{Global} = T^*$
                    *****End critical section*****
                *****Synchronization barrier*****
                $T^* = T^*_{Global}$
        Update the pheromone matrix τ

visited using a stochastic mechanism. An ant k at city i has not visited set of cities S_p then P_{ij} be the probability to visit edge k after edge i.

$$P_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{j\epsilon S_p} \tau_{ij}^{\alpha}\eta_{ij}^{\beta}} & if\ j\epsilon S_p \\ 0 & \end{cases} \tag{1}$$

S_P represents the set of cities which has not been visited yet and to be visited again so that the probability of the ant visiting a city which has already visited becomes 0. Where τ_{ij} is the pheromone content on the edge joining node i to j . η_{ij} represents the heuristic value which is inverse of the distance between the city i to j, which is given by:

$$\eta_{ij} = \frac{1}{d_{ij}}$$

Where d_{ij} is the distance between the city i to j. α and β represents the dependency of probability on the pheromone content or the heuristic value respectively. Increasing the value of α and β may vary the convergence of ACO.

After solution construction we have to update the pheromone accordingly, as follows:

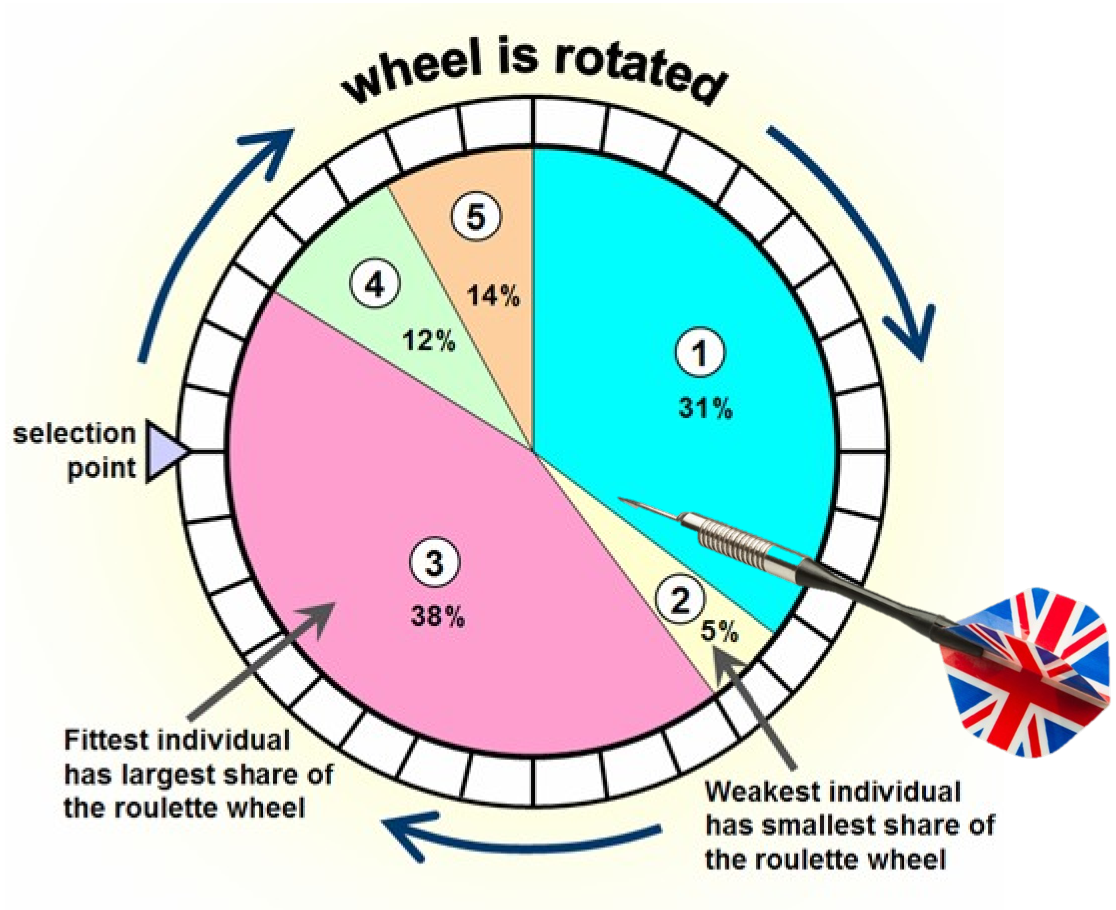
$$\tau_{ij} \leftarrow (1-\rho).\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$$

Where ρ is the evaporation rate, *m* is the number of ants, and $\Delta\tau_{ij}^{k}$ is the quantity of pheromone laid on edge(i,j) by an ant k:

$$\Delta\tau_{ij}^{k} = \begin{cases} Q/L_k & \text{if ant } k \text{ uses edge } (i,j) \text{ in its tour,} \\ 0 & \text{otherwise} \end{cases}$$

Where Q is a constant and L_k is the length of the tour constructed by an ant k.

Roulette Wheel Selection



Printing the best tour

```
struct {
   int cost;
   int rank;
} loc_data, global_data;

loc_data.cost = Tour_cost(loc_best_tour);
loc_data.rank = my_rank;

MPI_Allreduce(&loc_data, &global_data, 1, MPI_2INT, MPI_MINLOC, comm);
if (global_data.rank == 0) return;  /* 0 already has the best tour */
if (my_rank == 0)
   Receive best tour from process global_data.rank;
else if (my_rank == global_data.rank)
   Send best tour to process 0;
```

Homework 6

- 使用 MPI+OpenMP 實作，每一台電腦各啟動一個 process ，每個 process 再 fork 出 multi-thread