CCVRP optimization with ACO

* Problem Description:

The Clustered capacitated vehicle routing problem (CCVRP) consist of n-1 costumers with certain need and one depot with some vehicles with specific amount of capacity.

Each customer vi (i ∈ {1,…,n}) has a known nonnegative demand di to be delivered or collected and the depot has a fictitious demand d0 = 0. There exist m identical vehicles, each with a capacity Q and in order to ensure feasibility we assume that di ⩽ Q for each i ∈ {1,…,n}.

Problem assumption:

* + each route starts and ends at the depot vertex;
  + once a vehicle enters a cluster, it visits all the vertices within the cluster before leaving it;
  + the sum of the demands of the visited vertices by a route does not exceed the capacity of the vehicle, Q.
* Instances Description:

Instances are created based on CVRP instances form TSPLIB library with difference that we created new problem that is a clustered version of CVRP.

Each CVRP instance file consists of two part as **specification part** that contains information about the instance data and **data part**.

* Algorithm Description:

ACS differs from the previous AS because of two main aspects:

* Pheromone:

Change to the probabilistic function: drop alpha



in ACS only the best solution computed since the beginning of the computation is used to globally update the pheromone.



Whattt???????



* Update pheromone trail while building the solution
* Ants eat pheromone on the trail
* Local search added before pheromone update

*local updating* of the pheromone applied during the construction phase.

Each time an ant moves, the pheromone associated to the edge is modified



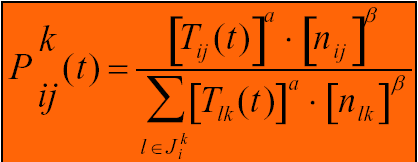




* where is the tour length produced by the execution of one ACS iteration without the pheromone component (this is equivalent to a probabilistic nearest neighbor heuristic)

for the edges which never belonged to a global best tour the pheromone remains  

* State Transition Rule



1. **Initialize** Trail
2. **Do While** (Stopping Criteria Not Satisfied) – Cycle Loop

**Do Until** (Each Ant Completes a Tour) – Tour Loop

Local Trail Update

**End Do**

Analyze Tours

Global Trail Update

**End Do**

**ACS-State Transition Rule**:



As rule (page15):

alpha = 1 beta = 2

