CCVRP optimization with Genetic

* 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:

The algorithm designed base on the related paper as *(A novel two-level optimization approach for clustered vehicle routing problem).*

Our approach is obtained by decomposing the problem into two logical and natural subproblems:

an upper-level (global) subproblem and a lower-level (local) subproblem. The first subproblem aims at determining the routes visiting the clusters, called global routes, using a genetic algorithm applied

to the corresponding global graph (see details in Section 3) while the aim of the second subproblem is to determine the visiting order within the clusters for the above-mentioned routes. The second subproblem is solved by transforming each global route into a TSP which then is

computed optimally using the Concorde TSP solver.

* Global subproblem:

As mentioned, we solve this subproblem using GA, so we describe out GA as follow parts:

* + Representation:

Global subproblem search space is consist of number of clusters Vi (i ∈ {0…,n}) witch depot node placed in V0 cluster .

So out representation would be a sequence of cluster numbers that shows out global routes toward clusters. Depot cluster would be seen repeatedly as finish each route from depot and back to it.

 

As images shows one feasible solution can be: (7 1 2 0 3 4 0 5 6)

For creating the chromosome, we list cluster needs in descending and trying to satisfy needs by minimum vehicle number.

* + Fitness function:

The fitness function of each individual chromosome in the population is given by the total length of the best corresponding clustered routes associated to the collection of global routes specified by the chromosome. This distance also takes into account the order in which the vertices within the clusters are visited. Our aim is to minimize this total distance.

* + CrossOver:

Our GA uses a custom version of the two-point crossover. The crossover function takes two parent candidate solutions as input and outputs two solutions.

* + Mutation:

we use a swapping inter-cluster mutation operator which acts as follows: we randomly select genes (i.e. clusters) and if the genes are from different global routes, their position is exchanged.

(5 8 1 0 3 7 0 6 4 2) -> (5 6 1 0 3 7 0 8 4 2)

* + Parent Selection:
  + Survivor Selection:
* Local subproblem:
* Simulated annealing algorithm progress:

T = 1

ALPHA = 0.8 (for using in temperature updating)

TEMP\_MODE = EXP (temperature updating method)

INIT\_HEURISTIC = True (using initial heuristic)

NUM\_ITERATIONS = 500

* + Algorithm progress plot for sample instances:



p43.4.sop jpeg.4753.54.sop

The whole results (main, max, avg) came at the end.

* + **Without local search:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instance | best | worst | average | avg\_time |
| ESC78.sop | 20465 | 21370 | 21194.5 | 1.8862 |
| R.200.100.60.sop | 85057 | 87786 | 85892.3 | 24.435 |
| susan.260.158.sop | 1124 | 1143 | 1134.6 | 6.5894 |