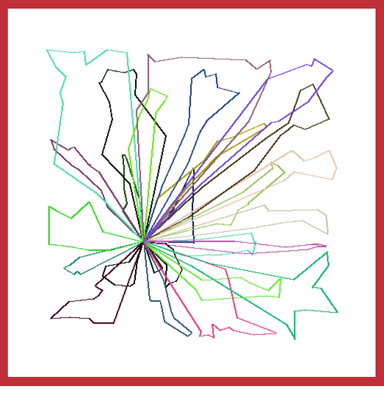
Genetic algorithm for the CVRP

Capacitated Vehicle Routing Problem, Algorithm type: Heuristic

Callum Mann

University of Bristol, Department of Computer Science 2016



Representation

The representation of solutions (*chromosomes*) are lists of tours that visit customers. The tour may exceed the trucks capacity making the solution infeasible, however there is a penalty term associated with each solution that affects their fitness. The benefit of this representation is that it allows genetic operators to arrange routes more freely and potentially generate much more optimal offsprings, as the optimal solutions are often very close to being infeasible.

Initial Population

The initial population is designed to contain optimised routes with some randomness. The purpose of some randomness in the initial solution is so that the population does not converge too quickly resulting in cost stagnation.

The population is created by choosing a random starting node, and performing nearest neighbour until the capacity of the truck is reached. With some probability, the nearest neighbour will favour routes closer to the depot, so that the population is not identical at the beginning. After these routes have been created, they are passed into the **2-Opt**, to optimise the order of the routes. These routes then have a starting cost of around 6500, depending on the randomness parameter.

Generation Step

The population is changed every step, by removing an undesirable solution and replacing it with a new child that ideally has a better fitness value. The population is stored in a priority queue and is advanced by the following steps:

```
def GenerationStep()
  for p1 in best n do
    for p2 in best n do
      child ⟵ BiggestOverlap(p1, p2)
      for some iterations do
        child ⟵ BiggestOverlap(p1, child)
      end
      2-Opt(child)
      Repair(child)
      Introduce(child)
    end
  end
end
```

Both P1 and P2 typically have high fitness. This is done to maximise the chance of creating new optimal solutions. The children are created from a crossover in P1 and P2 that may be repeated several times between P1 and themselves. This increases the chance of improved children per iteration.

The children also passed through **2-Opt** after the operators as they often leave some room for local optimisation after routes have been moved around. The children are partially repaired at the end of the generation so that routes cannot become unsightly, although typically only feasible solutions are optimal in later generations due to the penalty term outweighing reduced cost.

Local Search: 2-Opt [2]

The order in which a truck will visit customers may be very unoptimised, so the **2-Opt** is used to search locally *within* the solution for lower costs. The intuition behind the algorithm is simple; it searches for places in the tours where one path crosses over another, and removes this cross. An implementation of this found in other literature[0] called this a method of steepest improvement.

Crossover: Biggest Route Overlap [0]

In this crossover a random subroute of one solution is placed into another solution. First, the bounding boxes for all tours are calculated, then the bounding boxes of the tours in the destination solutions are compared with the random subroute. Out of the routes with maximal bounding box overlap, the route with minimal demand is chosen for the subroute to be placed into optimally.

It was found that this crossover is far superior to crossovers such as **pmx**, which does not tailor the crossover mechanic to the TSP problem. Methods based purely on randomness perform poorly as the solution becomes more optimal.

Mutation: Minimal cost insertion

This mutation selects a random customer from the solution and places it into another route. The selected customer is inserted between the two customers that minimise distance travelled. This typically helps later in the algorithm when moving whole subroutes causes too much cost increase.

Evaluation

The lowest cost ever achieved by the algorithm was 5889 for the *fruitybun250* dataset, which represents a solution within 5.5% of the best known value (5583). As with many GAs, the downfall is population convergence and stagnation, and other heuristics such as TABUROUTE[1] can achieve 1% within the optimal.

An improvement to the algorithm would be found in the crossover operator, more specifically one may be able to track the history of successful operations and use this in future generations. This decreases the reliance on randomness and would target tours that are not optimal based on previous evidence.

References

[0] - A. S. Bjarnadottir (2004), *Solving the Vehicle Routing Problem with Genetic Algorithms*,

[1] - M. Gendreau et. al (1994), *A Tabu Search Heuristic For The Vehicle Routing Problem*,

[2] - G. A. CROES (1958). *A method for solving traveling salesman problems*

University of BRISTOL