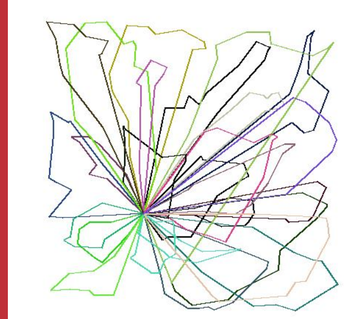
Genetic algorithm for the CVRP

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 **Steepest Improvement Algorithm (SIA)**, 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 generation is advanced through the following steps:

1. P1, P2 = ParentSelection()
2. Child = BiggestOverlapCrossover(P1, P2)
3. Child = Iterate crossover(P1, Child) n times
4. Mutate(Child)
5. SteeplyImprove(Child)
6. Repair(Child)
7. Introduce(Child)

P1 is typically a solution with a high fitness value, and P2 is randomly chosen. This is done to maximise the chance of creating new optimal solutions. The child is created from a crossover in P1 and P2 that may be repeated several times between P1 and itself. This increases the chance of an improved child per iteration.

The child is also passed through the **SIA** after the operators as they often leave some room for local optimisation after routes have been moved around. The child is 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: Steepest Improvement Algorithm [0]

The order in which a truck will visit customers may be very unoptimised, so the **SIA** is used to search locally *within* the solution for lower costs. The algorithm iterates over two indices, and compares the distance between themselves and the next customers. The customers are then swapped if the comparisons show a lesser cost between the indices. This process is repeated upon a route until there is no more cost to reduce.

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 routes are calculated, then the bounding boxes of the routes 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.

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 routes causes too much cost increase.

Evaluation

The lowest cost every achieved by the algorithm was 5937 for the *fruitybun250* dataset, which represents a solution within 6.4% 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.

References

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

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

University of BRISTOL