A Two-Phase Genetic Algorithm Approach for the Capacitated Vehicle Routing Problem

65091 Iman Malik im13557

University of BRISTOL

Introduction

The **Capacitated Vehicle Routing Problem** (CVRP) is a **NP-hard** combinatorial optimisation problem where vehicles are required to deliver goods to **n** customers while starting and finishing at a depot. CVRP seeks to find a path that visits each customer exactly once whilst trying to minimise the cost of delivery. This study proposes a two-phase genetic algorithm to solve this problem

Fig 1. Best Found Solutions for Phase 1

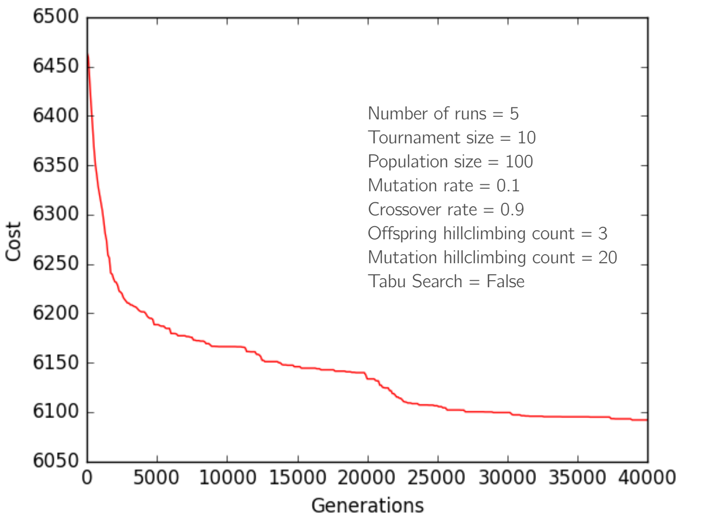
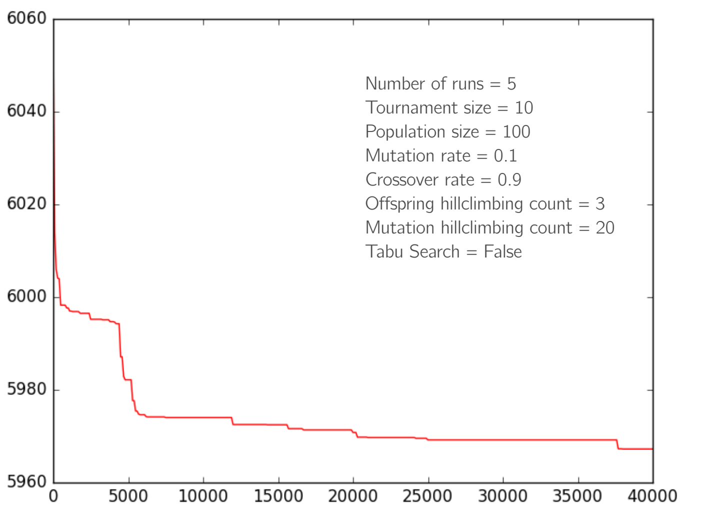


Fig 2. Best Found Solutions for Phase 2



Phase 1 Algorithm Outline

Phase 1 of the algorithm focuses on the crossovers of a good initial population.

1. Population Generation

The initial population is generated using a **Cluster-First Route-Second Method**[1]. Random customers are chosen as cluster seeds and then routes are initialised using a nearest neighbour approach. After route construction, a genetic **TSP** is run on every **vehicle** cluster.

2. Evaluation of fitness

The **fitness** of each individual is calculated as the total **euclidean distance** travelled by the vehicles. The non-binary **chromosome** representation is the collection of **all vehicles and their routes**.

3. Elitism

The **elite**, about 1%, are carried to the descendent population.

4. Tournament Selection

After initialisation, **tournament selection** takes place to select suitable parents for a crossover.

5. Crossover

Using the tournament winners, several children are generated using **one-point crossover** where the **crossover points are random**. The algorithm only produces one child per crossover so a ***hillclimber*** is used and the fittest child is selected. If the ***tabu setting*** is on, the fittest child is **remembered**, and marked as **Tabu**.[2] This is to help the algorithm **explore** other children if the fittest child is generated again. The algorithm stores generated children in a *bloom filter* which resets after a certain number of entries. This allows the algorithms to both **exploit and explore.** A larger reset threshold would mean the algorithm would spend more time exploring than exploiting. However this option causes the algorithm to *slow down considerably*.

6. Mutation

The descendant population undergoes mutations before its next evaluation and evolution. A simple **random mutation method** is used. Two **customer nodes** are selected in the chromosome and if constraint requirements of the vehicles hold, they are **swapped**. A hillclimber is used to select the best mutation over several possible mutations.

References

[1] M. L. Fisher and R. Jaikumar. "A Generalized Assignment Heuristic for Vehicle Routing". Networks, 11:109-124, 1981

[2] F. Glover. "Future Paths for Integer Programming and Links to Artificial Intelligence".Computers and Operations Research, 13:533-549. 1986

Phase 2 Algorithm Outline

Phase 2 of the algorithm focuses on the mutation of a good previous solution. This allows the algorithm to possibly converge to a better solution by jumping out of a local optimum.

1. Population Initialisation

After Phase 1 converges, the **previous best solution** is used as the input to the second phase of the algorithm.

2. Phase 1

The **Phase 1 GA** is run on the population while randomness/diversity is introduced to the population by **increasing** the mutation rate over the initial *100* iterations.

2. Increase Capacity Constraint

An allowance threshold of **10%** is periodically applied to the demand constraint: after every **500 iterations of seeing a better solution**. The algorithm also increases the mutation rate to **70%** and then switches back to **20%**, the default setting. The default settings are higher than Phase 1 by **10%** as it is *more likely* for the algorithm to converge at a local optimum as the solution approaches the optimal cost.

Evaluation

Fig.1 shows that Phase 1 produces best solutions for at least **40000** generations of the algorithm. If the algorithm was allowed to compute for longer, better solutions would be most likely seen.

Phase 2 initially rapidly reduces the cost when randomising the population as seen in **Fig.2**. **Local optima** can be observed such as one after 5000 generations which is broken out of by randomness introduced by mutations and invalidity. Again, similar to Phase 1, the algorithm would most likely continue to produce solutions if left to compute for longer.

Future Refinement

Tabu search could be turned on for offspring generation (and possibly mutation selection) for both Phases and the graphs would most likely plateau ***less quickly*** in comparison and eventually reach better solutions. However the downside would be that due to its exploratory nature, it would take **significantly longer**.

By observing **Fig,2**, we can see that Phase 2 performs really well initially. By further dynamically tweaking mutation rates during the learning run, the algorithm could possibly produce better solutions at a faster rate.