CS303 Project2: Genetics-based Heuristic Search Algorithm for Capacitated Arc Routing Problem

Shijie Chen
Department of Computer Science and Engineering
Southern University of Science and Technology
Shenzhen, Guangdong, China
Email: 11612028@mail.sustc.edu.cn

***Abstract*—The Capacitated Arc Routing Problem (CARP) is one of the many arc routing problems and has been proven to be NP-hard. In this project, the author implemented a genetic-based heuristic search algorithm. The algorithm is able to obtain good result on both small and larger datasets.**

I. Preliminaries

The Capacitated Arc Routing Problem is the problem of serving certain edges on a graph with a fleet of vehicles under a constraint on capacity [1]. The objective is to minimize the total cost in order to service all the tasks.

CARP has multiple applications in real life. For example, garbage collection and resource distribution. Problems that can be modeled as CARP often focus on serving edges rather than nodes.

II. Methodology

In this project, I used genetic algorithm as the main framework. Initial solutions is generated by a generalized path-scanning algorithm and Ulusoy split algorithm. Multiple mutation operators are designed to enhance search ability.

A. Notation

The notations used in this report is shown in the table below.

TABLE I
REPRESENTATION

Name	Variable
All tasks	tasks
Tasks remain	undone

B. Genetic Algorithm

1) Framework: A genetic based framework is used as the main search algorithm. Initially, a population is created by heuristic approaches. In each iteration, apply certain genetic operations to the population. In this problem, since crossover between different solutions hardly makes sense, only mutation is used. After that, a selection based on the fitness value of individuals in the population is used to keep the size of population unchanged.

Algorithm 1 Genetic Algorithm Framework

```
1: $population \leftarrow initPopulation()$
2: $size \leftarrow len(population)$
3: while end condition not met do
4:     $offSpring \leftarrow genOffspring(population)$
5:     $population \leftarrow population + offSpring$
6:     $population.sort(key = cost)$
7:     $population \leftarrow population[0 : size]$
8: end while
       return $population[0]$
```

2) Initial Population: The population is initiated by a Generalized Path Scanning algorithm, which generates a sequence of tasks to be done, and Ulusoy Split algorithm, which takes in a sequence of task and finds an optimal path in that sequence. Since Ulusoy Split algorithm only cares about the sequence of the task, load is not considered in the Generalized Path Scanning algorithm. Computational experiments also show that performance is better when path scanning ignores the capacity constraint.

The DAG in Ulusoy Split is a graph with 2N nodes. Each edge in the graph has its starting point index less than its end point index. Each path in the graph with odd starting point and even end point denotes a route that finished all the tasks between the two ends. A path from node 1 to node 2N is a solution to the CARP problem.

- Ulusoy split Ulusoy Split works by converting the tasks to be done to a directed acyclic graph (DAG). Then obtain the optimal solution by searching for a shortest path in the DAG. Details of the algorithm can be found elsewhere. XXXXX

Algorithm 2 Ulusoy Split

```
1: function ULUSOYSPLIT(tasks, depot, shortestPath, load)
2:     $DAG, incoming, outgoing \leftarrow toDAG(tasks)$
3:     for $node \in DAG$ do
4:         $minCost \leftarrow inf$
5:         $bestEdge \leftarrow minCost(incoming[node])$
6:         $node.bestPath \leftarrow bestEdge.bestPath.append(node)$
7:         $node.cost \leftarrow bestEdge.cost$
8:         for $edge \in outgoing[node]$ do
9:             $edge.cost \leftarrow edge.cost + node.cost$
10:            $edge.bestPath \leftarrow node.bestPath$
11:        end for
12:    end for
13:    $x \leftarrow 1$
       return $node[-1].cost, node[-1].bestPath$
14: end function
15: function TODAG(tasks, depot, shortestPath, load)
16:    $construct\ the\ DAG$
       return DAG
17: end function
```

- Generalized Path Scanning

3) Genetic Operators:

III. Validation

IV. Discussion

V. Conclusion

Acknowledgment

The authors would like to thank the TAs for their hint in the Lab and maintaining a online runtime platform.

References

[1] S. Wøhlk, *A Decade of Capacitated Arc Routing*, pp. 29–48. Boston, MA: Springer US, 2008.