2801ICT Assignment 3 –

K-shortest paths

S5084207

Yoon Jin Park

Bachelor of Computer Science

Griffith University

# Introduction

The purpose of this report is to address the development of the algorithm to solve the k-shortest loopless paths problem.

The problem focuses on finding the specified number of the shortest paths to move from a particular origin to a specified destination. The distance is determined by adding all weights between two vertices. This assignment problem articulates that the path must be loopless and the weights between vertices are undirected. In addition, the shortest path must be included as part of the solution while the rest of the solutions can be approximated as shortest paths.

# Algorithmic Design

## Overview of the algorithm and innovation

The algorithm to find the shortest path is selected to be Dijkstra algorithm because it is more efficient than Bellman-Ford algorithm when there are no negative weights. Initially, Bellman-Ford algorithm is implemented because it is assumed that there may be negative weights included within a sample of nodes. However, as the lecturer confirms that there will not be any negative weights, Dijkstra is implemented so as to improve time performance of the solution.

For k-1 alternative shortest paths, many algorithms are available for approximating solutions: Yen (1971), Eppstein (1997), K\* algorithm (Alijazzar and Leuem 2011), Large Networks (Akiba et al., 2015), Neural Network (Zhao and Sylla, 2013), or Genetic Algorithm (Wang et al., 2018). The algorithm for Yen, Eppsteain and K\* in addition to their variations have been around for a few decades while network approaches (e.g. Neural Network or Genetic Algorithm) for the k-shortest paths problem could be considered as relatively recent attempts. In this assignment, considering time and space efficiency and time constraints on the assignment deadline, approximation methods loosely based on Genetic Algorithm is selected as the overarching algorithm.

## Algorithm Description

This section addresses the classes implemented to solve the k-shortest paths problem and how the algorithm provides one shortest path in addition to other approximated shortest paths.

### Node class

The Node class is essential because it keeps an ID of a node as key (e.g. ‘C’ or ‘7865’) and weights to the next nodes. To do so, a dictionary data structure is used to store weights as value to the keys that are neighbour to a current node. Three getter methods and one setter method are implemented in this class, containing ‘get neighbours’, ‘get key’, ‘get weight’, and ‘add neighbour’.

The ‘get neighbours’ method provides the list of all neighbours connected from a current node. The ‘get key’ method obtains the key of the current node while the ‘get weight’ method returns the weights of the list of all neighbours linked to a current node. ‘add neighbour’ sets a list of neighbouring nodes with its weight information.

### Graph class

Built on the Node class explained previously, the Graph class includes four getters and two setters. In particular, two getters, ‘get vertex’ and ‘get vertices’, returns a current node object in the graph, a list of nodes in the graph, respectively.

The setters are important in this class because they help to construct a graph. The ‘add vertex’ method increases the size of graph and add a new node in a graph. The ‘add edge’ method sets weights of the neighbouring class by calling ‘add neighbour’ from the Node class.

### K Shortest class

The K Shortest class implements the original Dijkstra algorithm in addition to some variations.

First, traditional Dijkstra algorithm

A small number of variations to Dijkstra algorithm are explained in the next section, “Aspects on innovation”.

### Aspects on innovation

Two principles are adopted from Genetic Algorithm: mutation and population. Mutation aspect is simulated through Key concepts of normal GA include Population and Mutation. Population is a collection of Chromosome, meaning it consists of different valid paths. Mutation in Wang et al. (2018)’s concept corresponds to Crossover in that Mutation occurs in a Gene pool. However, for Mutation, a Gene pool is built from selecting one Chromosome/path from Population/collection of paths and a random valid path.

Placing all these concepts together generates a set of pseudo-codes in the next section.

## Algorithm Pseudo-Code

The following pseudo-code describes how K Shortest Algorithm is implemented to solve the K-shortest paths problem.

**Start**

**Initialise population by using BF algorithm and random initialisation method**

**Start loop for the number of evolutions**

**Create a gene pool and crossover/breed to make new paths**

**Add these new paths to the population**

**Create a gene pool and mutate the newly created population with the mutation percentage**

**Calculate fitness/total distance between the origin and the destination node**

**Selection by fitness criteria. Tournament selection if a graph is small (i.e. less than 150 nodes). Otherwise, Rank-based roulette wheel selection**

**Stop loop**

**Return k-shortest paths**

### Dijkstra algorithm initialisation

The Dijkstra algorithm is

# Result and Algorithm Analysis

In this section, the result of the algorithm and the analysis of this algorithm is presented.

## Result

The result to be returned from the algorithm must have two components: total score for each

## Algorithm Analysis

Two analysis is discussed in this section: Correctness and Performance. The correctness of the

### Correctness of the algorithm

The correct ness of algorithm is attempted to be proved by contradiction. Let us assume that BF algorithm and GA algorithm are implemented to provide correct outputs.

### Performance analysis

Sine the algorithm enables to build two priority queues from one list of balls, sharing the list of

# Conclusion

In this assignment,