Evolutionary Computation Final Project Proposal

Finding k-shortest Paths in Network Using Genetic Algorithm

0416235 劉昱劭

[Click here to see shorter description]

Abstract

In network topology, we often want to find some critical paths to make better performance or higher utilization of throughput. There are many real time multimedia services, like video streaming, need to guarantee stable bandwidth of data transmission. For this purpose, we need to find some paths with large enough bottleneck bandwidth, and these paths should be as short as possible for less transmission latency. If there are many paths acceptable, we could take these paths as redundancy to avoid congestion.

To find **k shortest paths** with **bandwidth constraint** is a significant issue.

Problem definition

Given a network topology (or a graph) $G(N,E)$, a source node S and a destination node D. There is a user-specified parameter **k** to decide at most how many paths we should find.

The whole search space could be represented by a search tree

e.g. Given **G(N,E)** = Fig.1, **Source** = 1, **Destination** = 4. The whole search space is showed as Fig.2.

Fig 1. A simple undirected graph

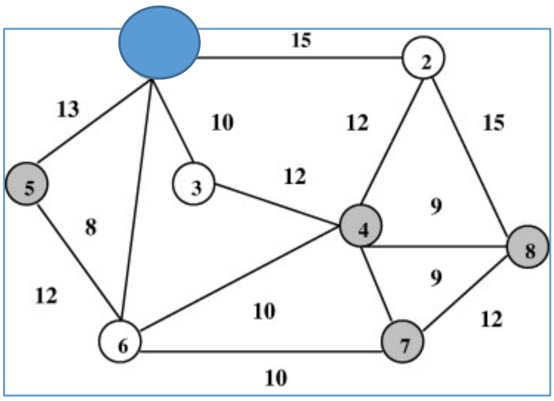
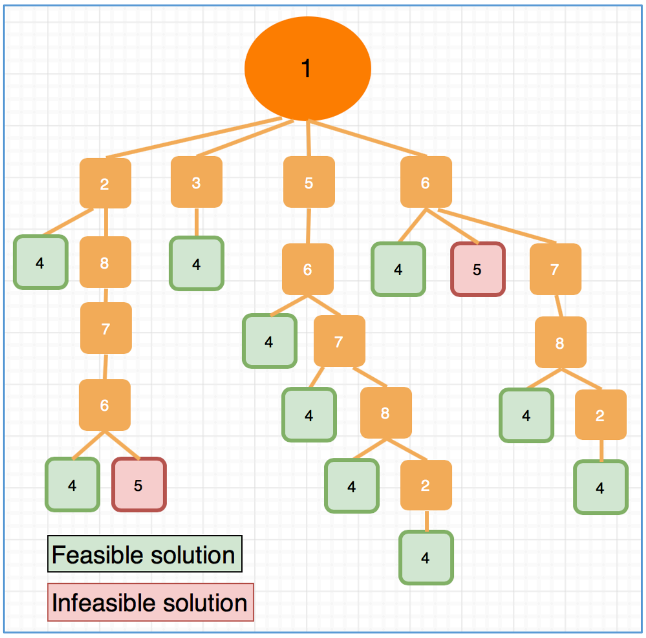


Fig 2. The search tree corresponding to 1→4 of Fig 1.



Method – Genetic Algorithm

- Individual representation
  - Integer array
  - Each individual shows a path from the source node to the destination node.
- Crossover
  - Use `one-cut point`
  - e.g. 2 parent individuals (2 paths)
    - we could cut these 2 individuals at both x_3 and get

x_1	x_2	x_3	x_4	x_5	x_6
Source	3	**5**	6	4	Destination

x_1	x_2	x_3	x_4	x_5	x_6	x_7
Source	**5**	9	8	12	11	Destination

  - we could cut these 2 individuals at the same node and get 2 children:

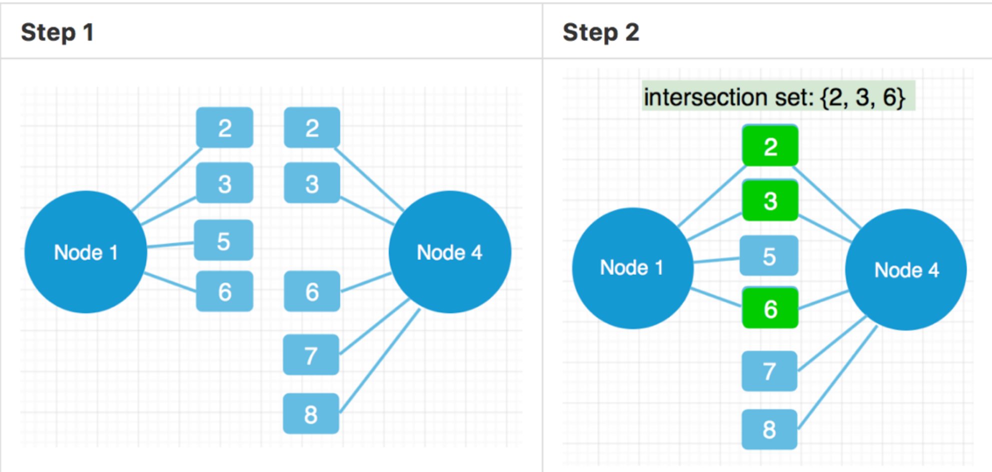
x_1	x_2	x_3	x_4	x_5
Source	**5**	6	4	Destination

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
Source	3	**5**	9	8	12	11	Destination

- Mutation
  - individual:
    - now mutate x_2=3

x_1	x_2	x_3
1	**3**	4

  - Bidirectional BFS to get common neighbor



  - mutate x_2 to one of {2, 3, 6}

x_1	x_2	x_3
1	**6**	4

Initialization – Seeding by Iterative-Deepening DFS

- Because the search space could be represented by a search tree, we could apply BFS or DFS on the given graph, eventually the search space could be traversed. BFS & DFS could be treated as exhaustive search or brute force.
- Let's meet them halfway. Iterative-Deepening DFS is a variant of DFS with a max depth d_{max}, this could traverse all branches within depth d_{max}, which takes advantages of the width of BFS and the speed of DFS.
- Initialization using Iterative-Deepening DFS (IDDFS)
    - To guarantee diversity and include known good solutions to population.

Fitness function

- Maximize $f_1(x) = min\,\{\,bandwidth(x) \mid x \in E_p\,\} \geq B$
    - $f_1(x)$ is the bottleneck bandwidth in the path E_p. B is the threshold bandwidth.
- Minimize $f_2(x) = |E_p|$
    - $f_2(x)$ represents the number of nodes in the path E_p, or the path length of E_p.

We could only optimize $f_1(x)$ so that this problem could be treated as single objective problem, or optimize both $f_1(x)$ & $f_2(x)$ so that this problem becomes multi-objective problem.

Related Work

For KSP problem, Yen's algorithm is very famous and efficient.

- Here is a GitHub repository that implements Yen's algorithm in python
  https://github.com/Pent00/YenKSP
  I will compare results of my project with this **YenKSP** (If I have enough time...)

Future Work

This work could be regarded as a template of path finding problem. If we change the objective function, this work could have different behavior.

- E.g.
  Minimize $f_3(x) = \sum_{x \in E_p} cost(x)$
  It will become a GA to find the shortest path.

Reference

Younes A. A genetic algorithm for finding the k shortest paths in a networks. Egypt Inform J 2010;11(2).

This project refers to this paper, but I think GA operators in this paper is not too novel or interesting, so I modify the objective function and add the seeding mechanism to make the initial population better.