Evolutionary Computation Term Project

Finding Widest k Shortest Paths in Network Using Genetic Algorithm

https://github.com/WarClans612/evolutionary_computation

0416106 彭敬樺

0416235 劉昱劭

0416223 許賀傑

Evolutionary Computation Term Project

Finding Widest k Shortest Paths in Network Using Genetic Algorithm

https://github.com/WarClans612/evolutionary_computation

0416106 彭敬樺
0416235 劉昱劭
0416223 許賀傑

Outline

- Introduction
- Approach
- Analysis
- Conclusion

Introduction

Path Finding in Networking

- Path diversity
- Guaranteed Minimum Bandwidth

Path Finding Networking

- Path diversity
  - For fault tolerance and avoiding congestion
  - To find more(k) shortest paths as redundancies
- Guaranteed Minimum Bandwidth
  - For quality of services (QoS)
  - 2 objective optimization
    - Maximize bottleneck bandwidth
    - Minimize path length

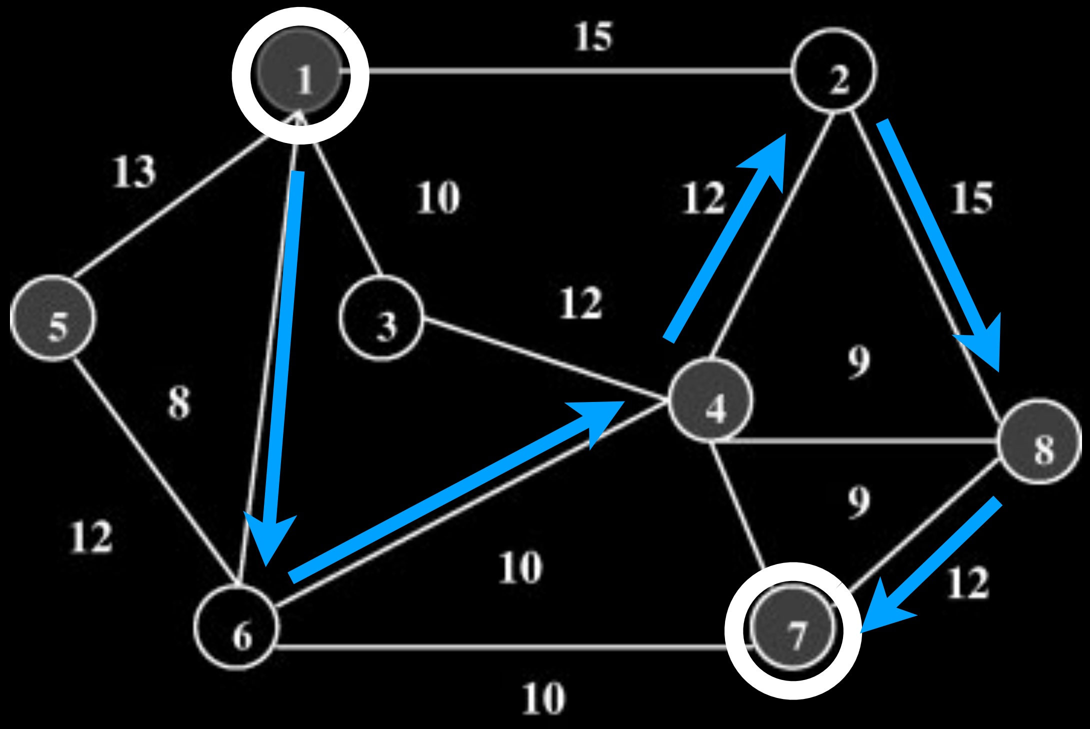
Widest Path

Vertices 1 6 4 2 8 7

Weight 8 10 12 15 12

Fitness = min(8 10 12 15 12)

= 8



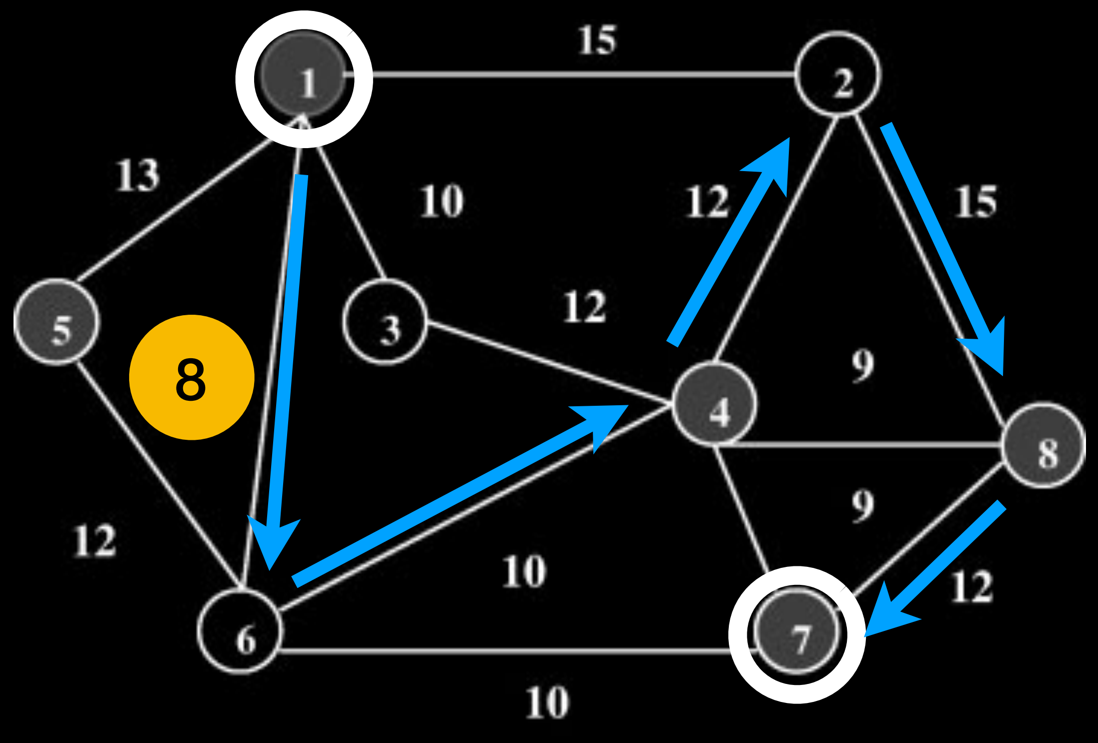
Widest Path

Vertices 1 6 4 2 8 7

Weight 8 10 12 15 12

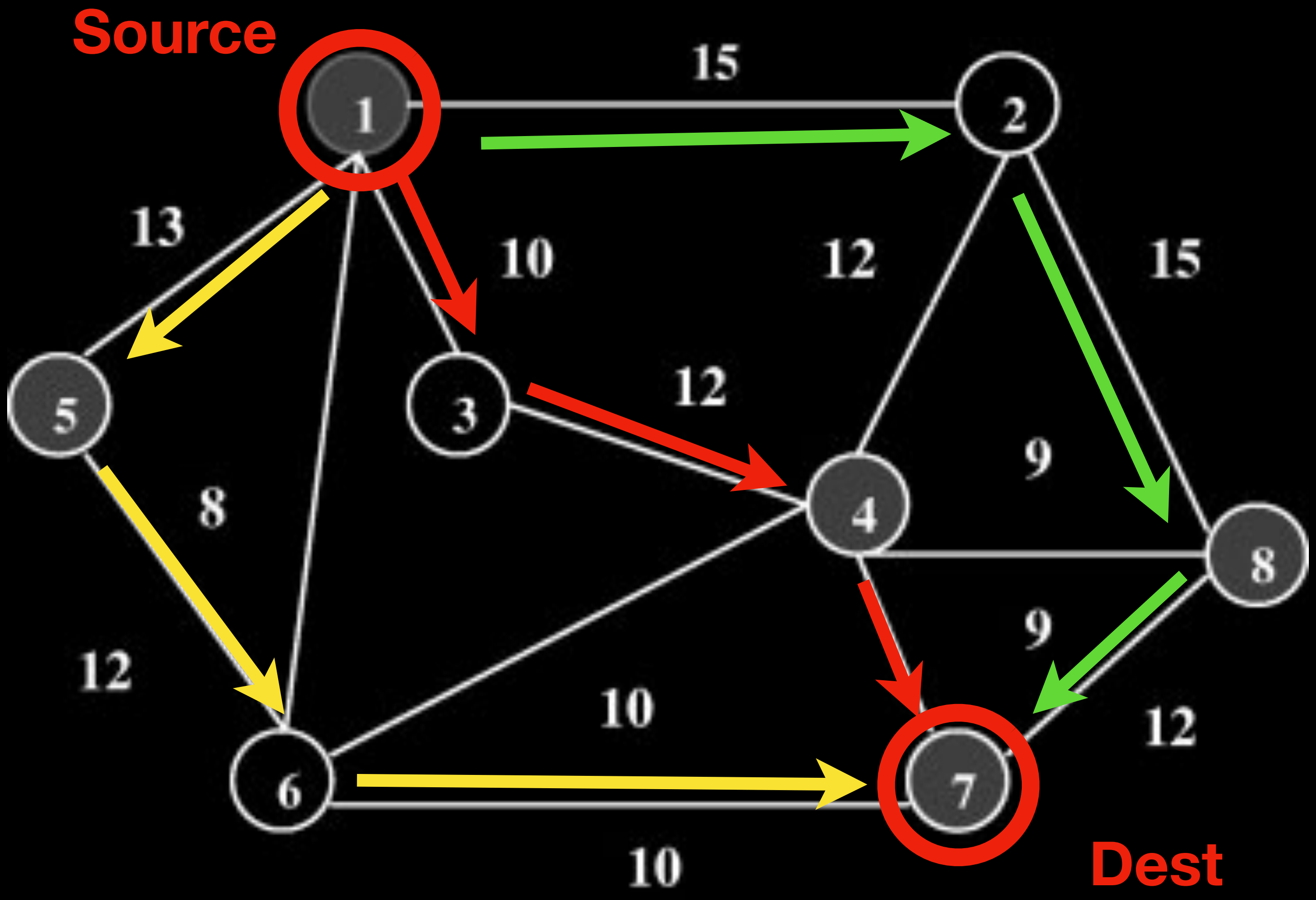
Fitness = min(8 10 12 15 12)

= 8



K Shortest Path

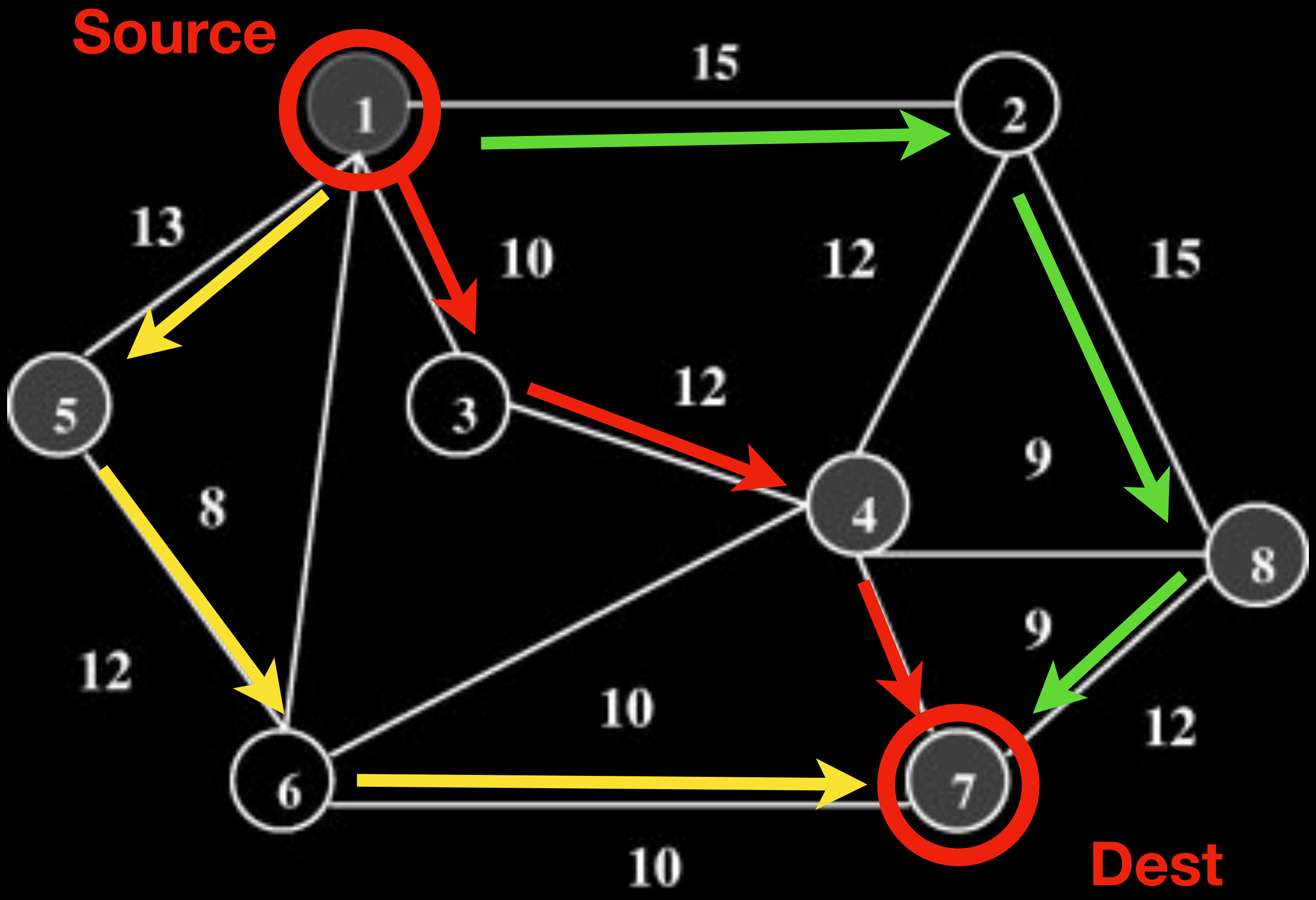
- Weighted graph
- For all the path, find top k shortest path given source and destination
- NP-Complete
  - Yen's algorithm



ref: https://en.wikipedia.org/wiki/K_shortest_path_routing

Widest K-Shortest Path In Network

- Use weight to represent bandwidth between hop
- Bottleneck would be the smallest bandwidth from source to destination
- Not minimizing the sum of weight



ref: https://en.wikipedia.org/wiki/K_shortest_path_routing

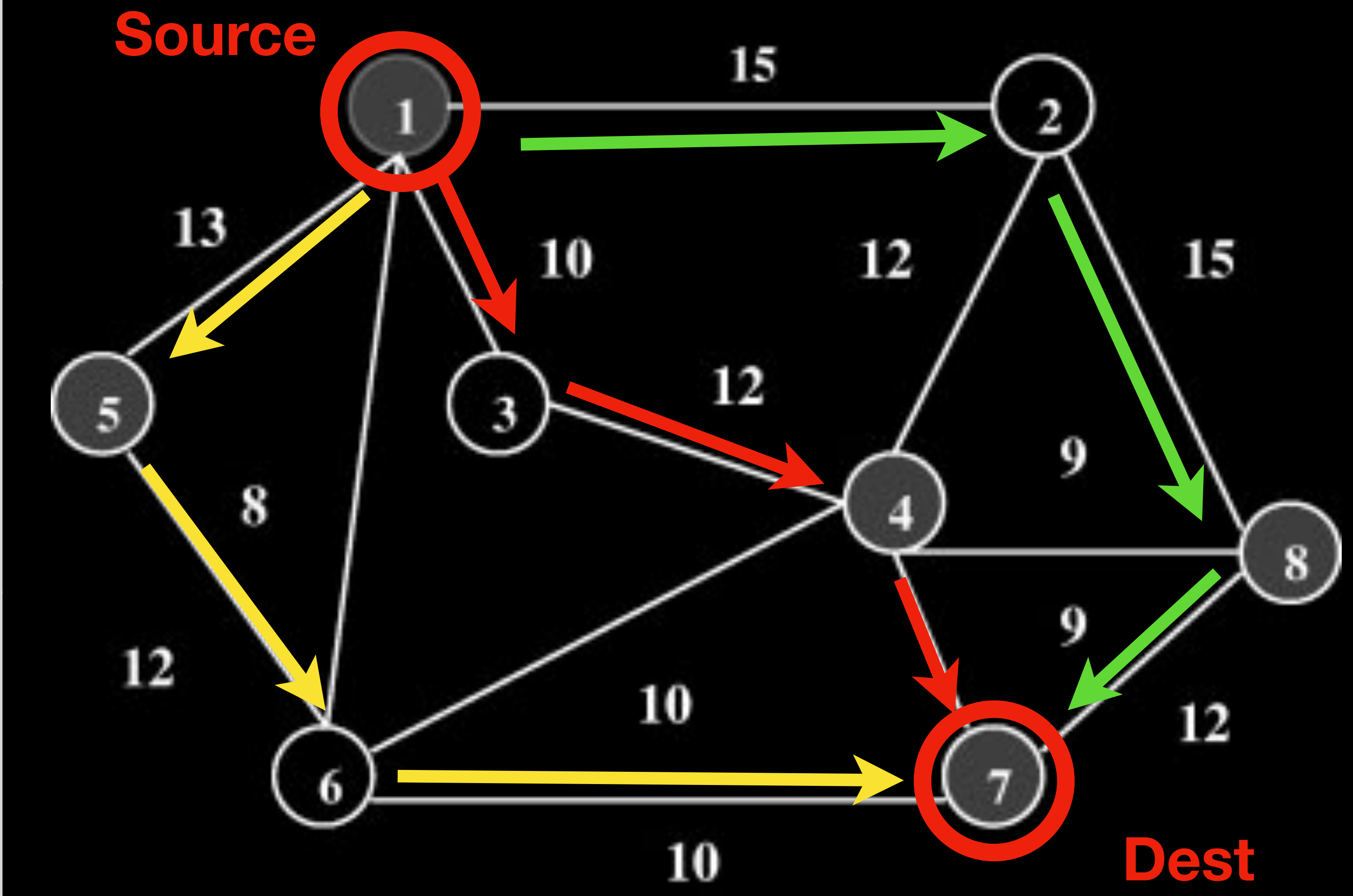
Suitable Problem For GA

- There might be multiple path to reach the same goal
- Goal is to find a good enough path without spending too much time
- Choosing solution other then optimal one is acceptable

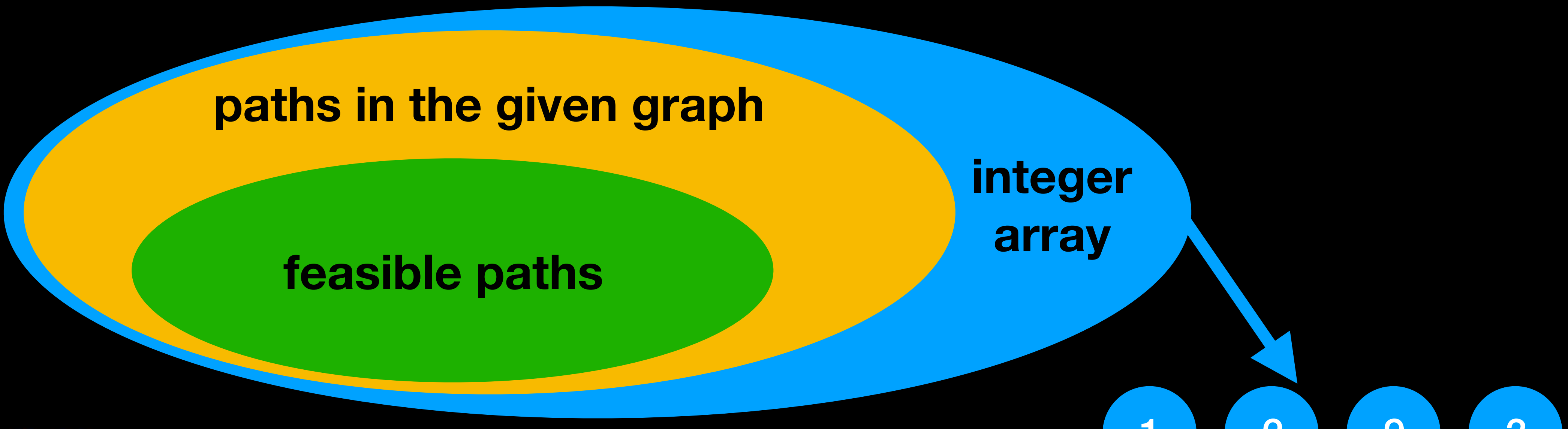
Approach

Encode

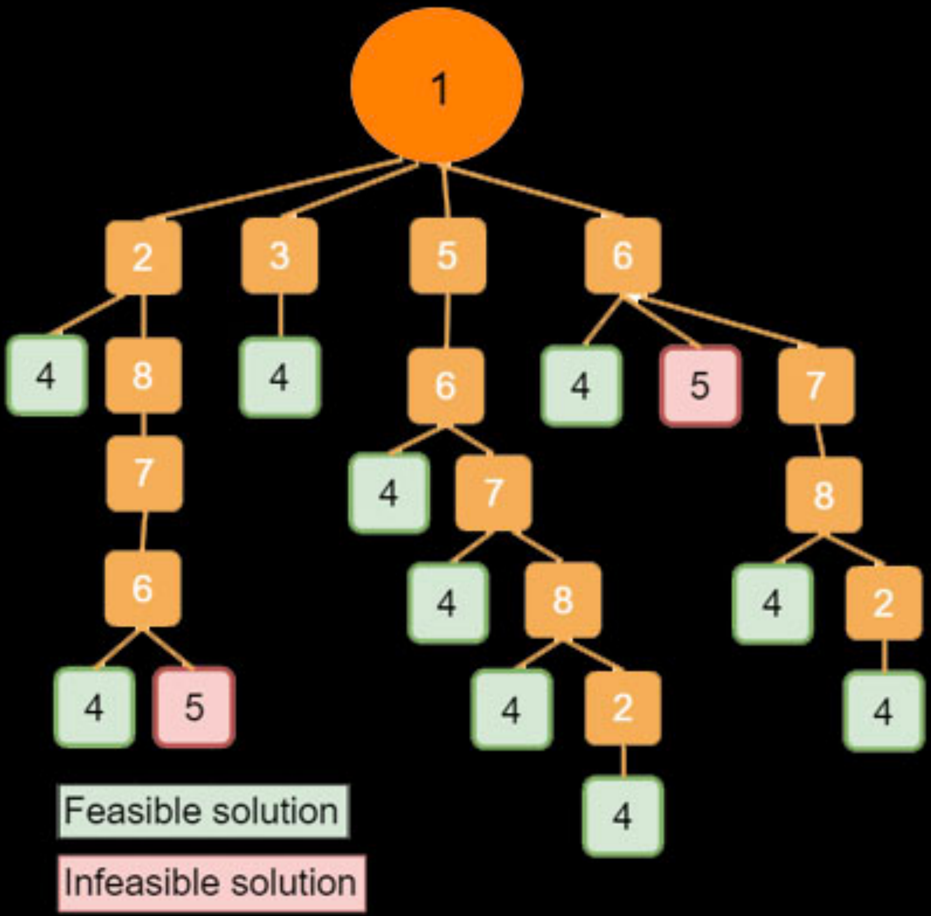
	Phenotype	Genotype
Red	1, 3 ,4, 7	(1, 3 ,4, 7)
Yellow	1, 5, 6, 7	(1, 5, 6, 7)
Green	1, 2, 8, 7	(1, 2, 8, 7)

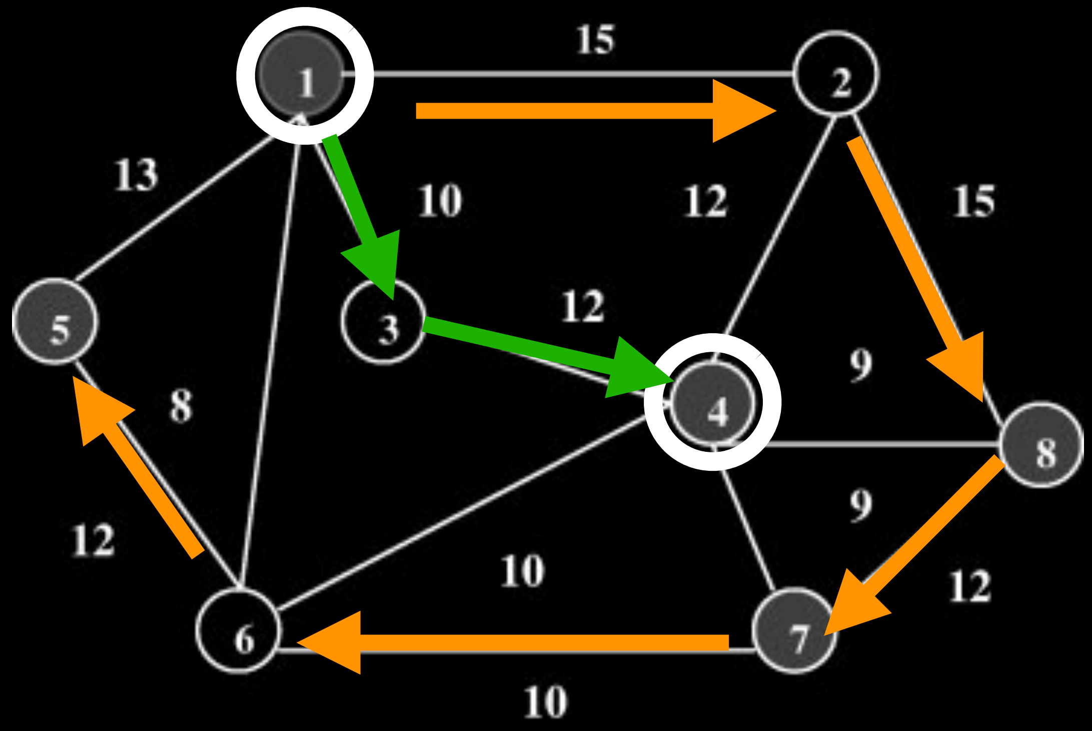


Search Space



1 2 9 3





Initialization

- Random walk
  - Walk from source to destination with at least N steps
  - Delete all cycles
- Mutation with high mutation rate
  - Create N individuals to match population size

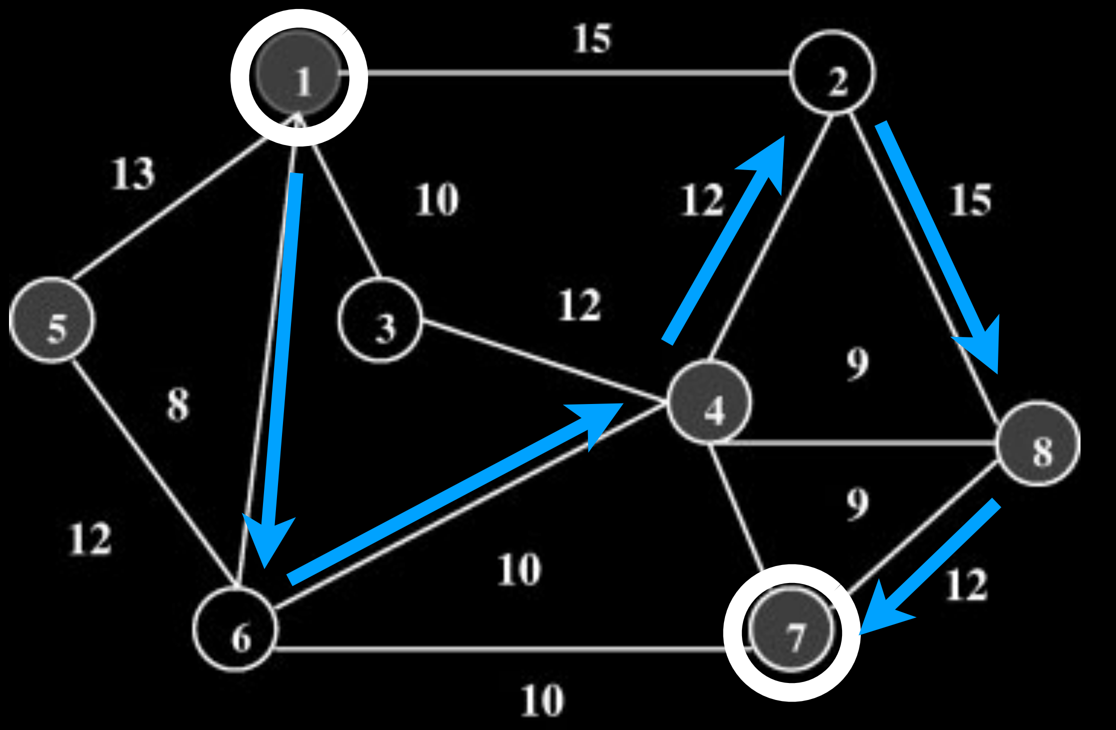
Fitness

Vertices 1 6 4 2 8 7

Weight 8 10 12 15 12

Fitness = min(8 10 12 15 12)

= 8



2-objective using ε - constraint

$$max\ f_1(x) = min\,\{\,bandwidth(e)\ |\ e\ \in\ individual\,\}$$

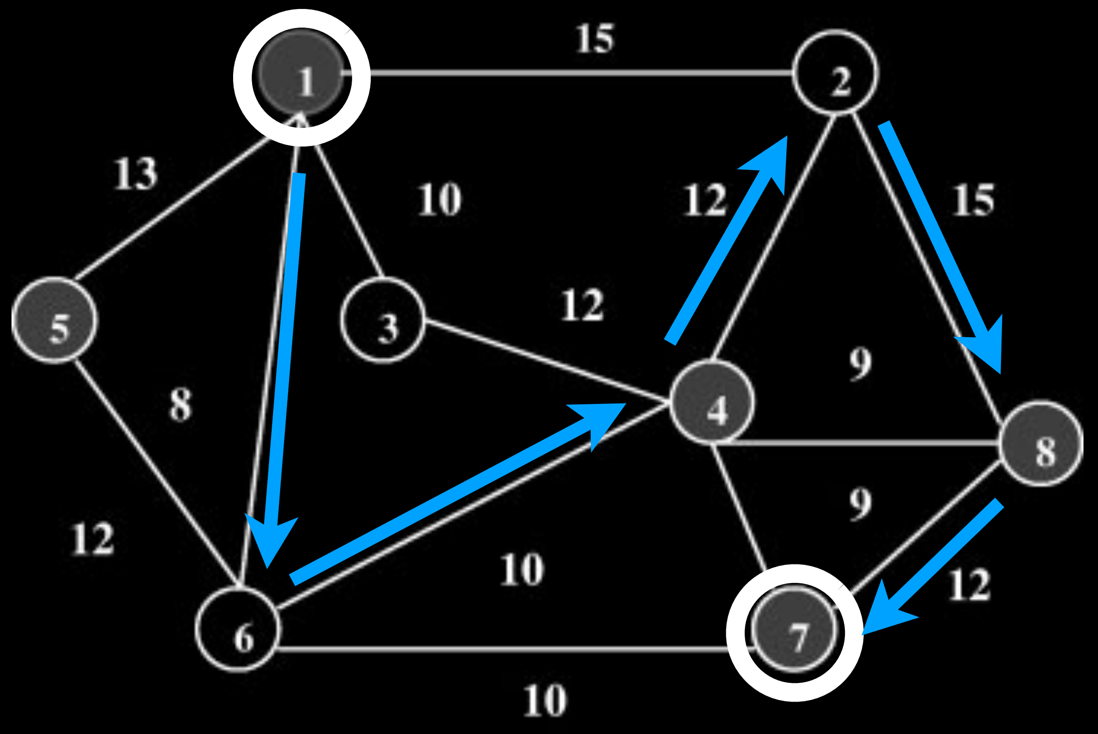
$$min\ \ f_2(x) = |individual|$$

Vertices 1 6 4 2 8 7

Weight 8 10 12 15 12

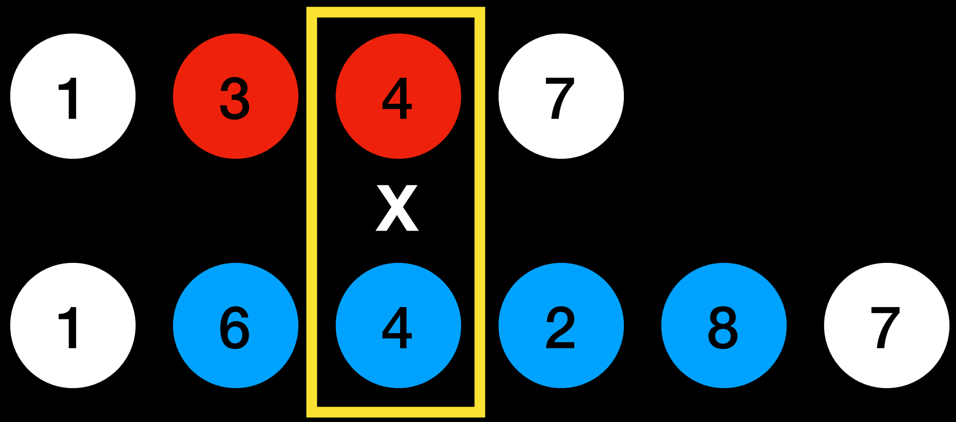
Fitness = min(8 10 12 15 12)

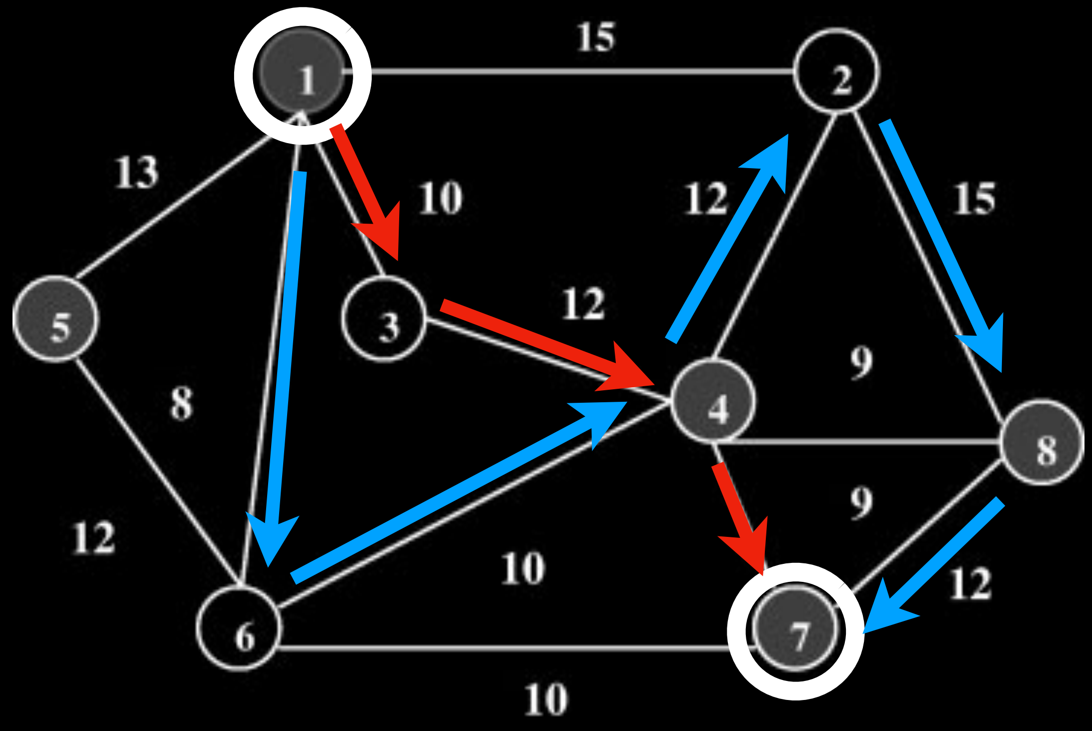
= 8



Crossover

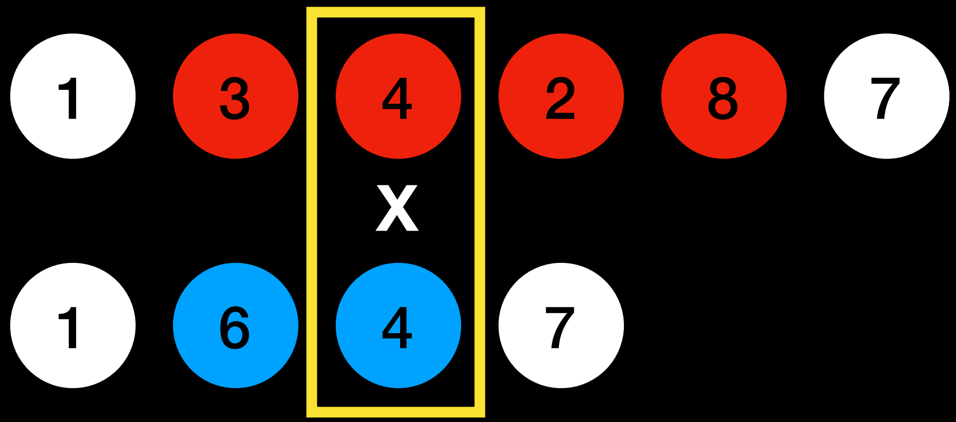
- Randomly choose a point from genotype
- One-point crossover
- Check if the result is a legal path
- Choose another point if the path is illegal

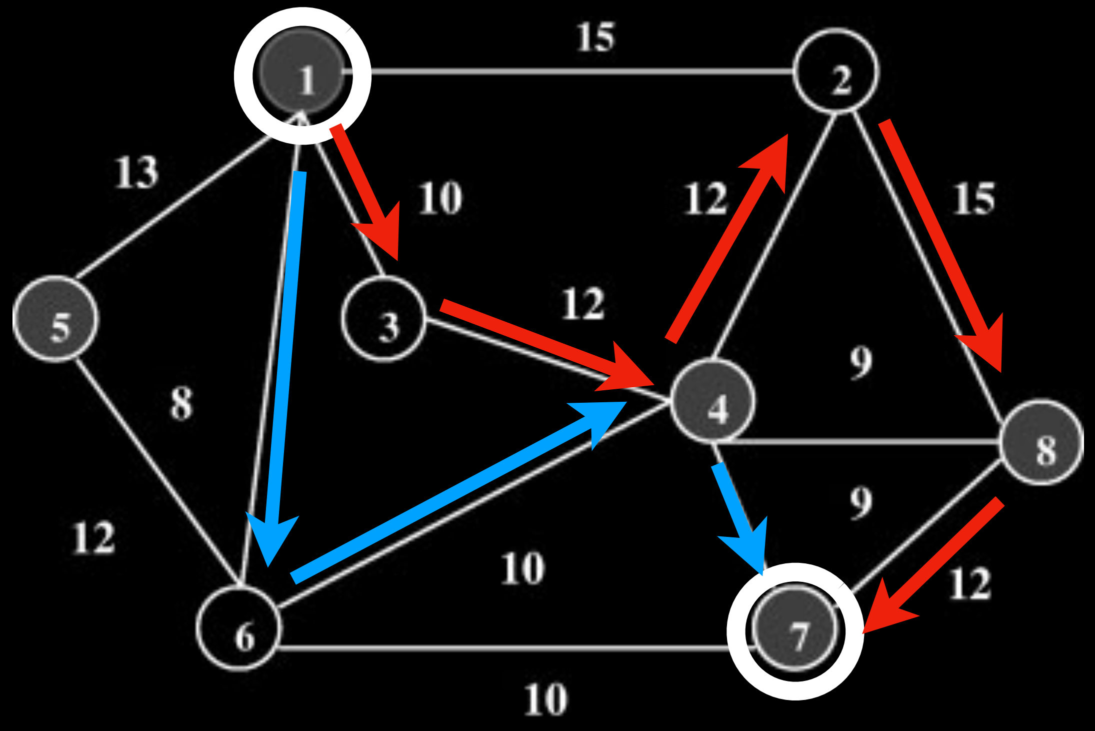




Crossover

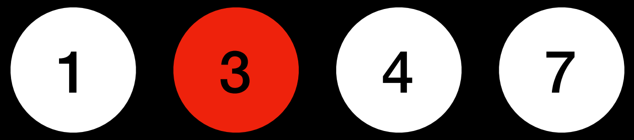
- Randomly choose a point from genotype
- One-point crossover
- Check if the result is a legal path
- Choose another point if the path is illegal



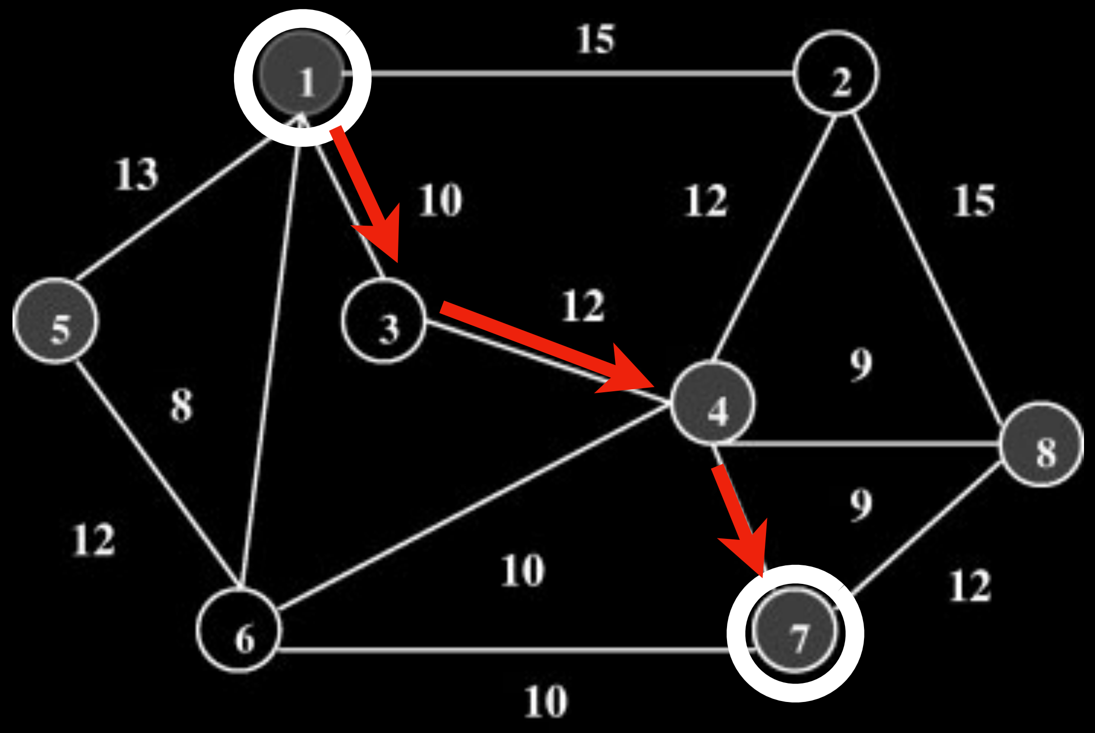


Mutation

- Randomly choose a point to mutate

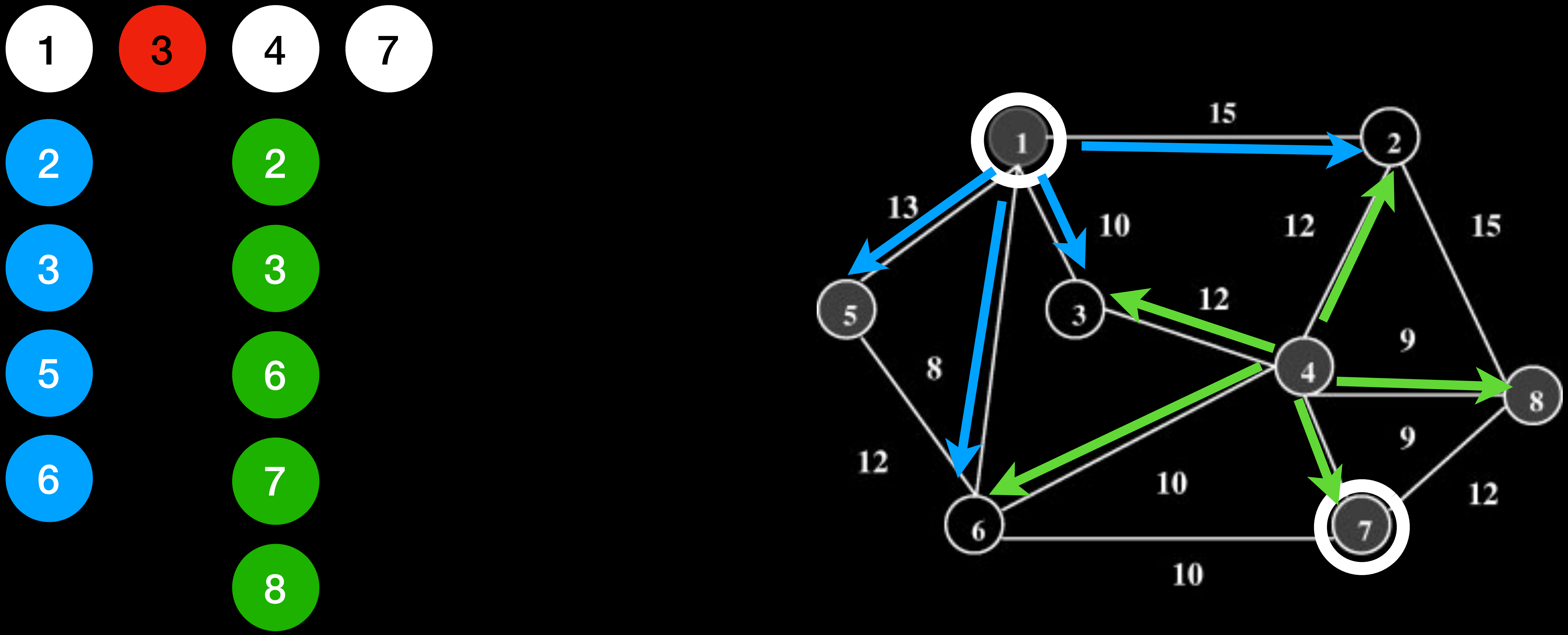


1 3 4 7



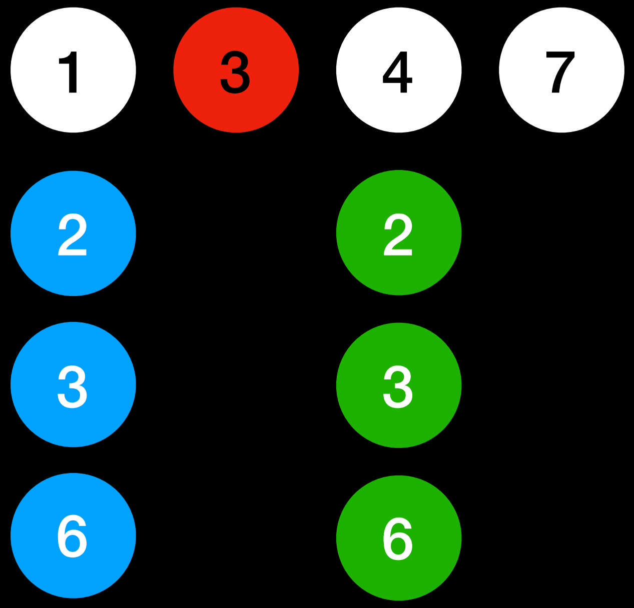
Mutation

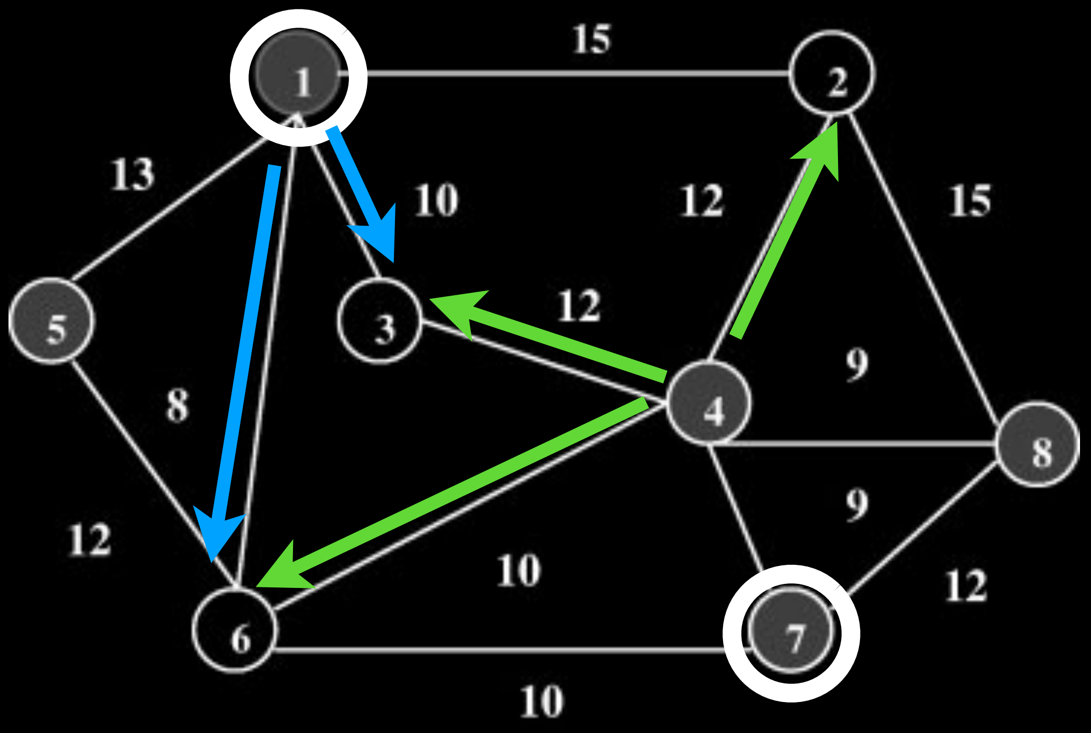
- Randomly choose a point to mutate
- Apply BFS on its previous and next node to get neighbors



Mutation

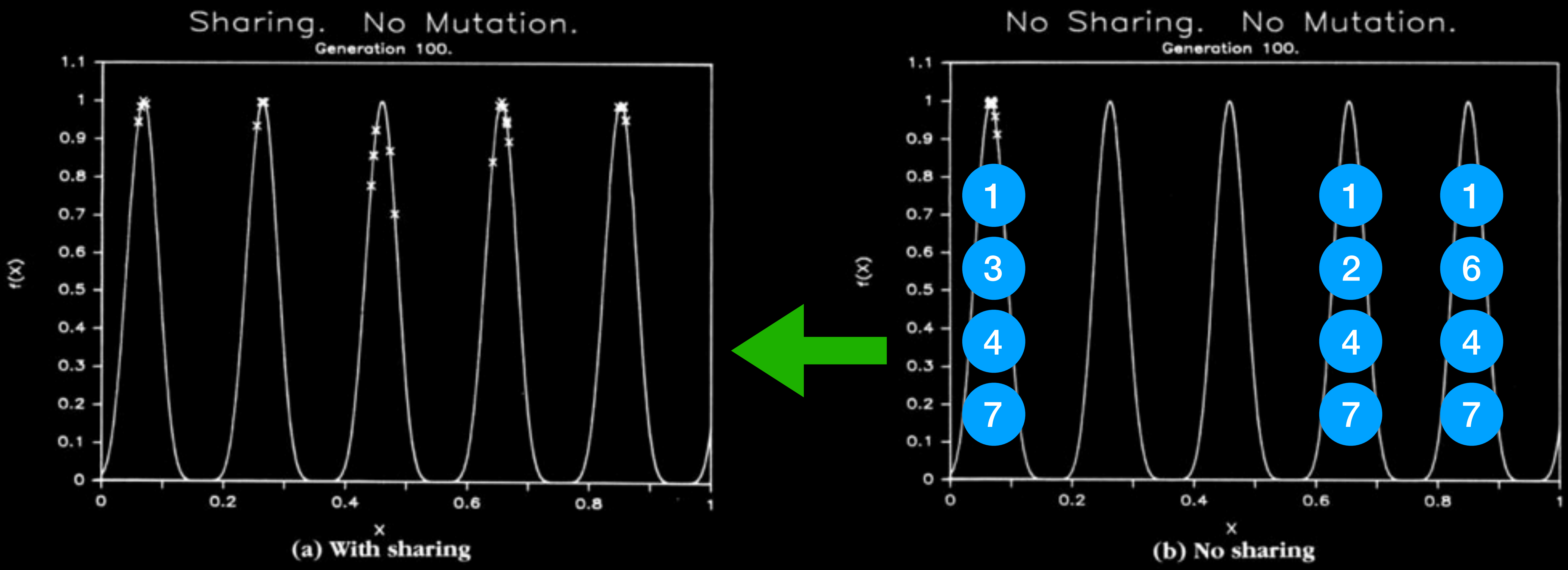
- Randomly choose a point to mutate
- Apply BFS on its previous and next node to get neighbors
- Choose one of the intersection of neighbors





Mutation

- Kind of diversity maintenance & local search



(a) With sharing

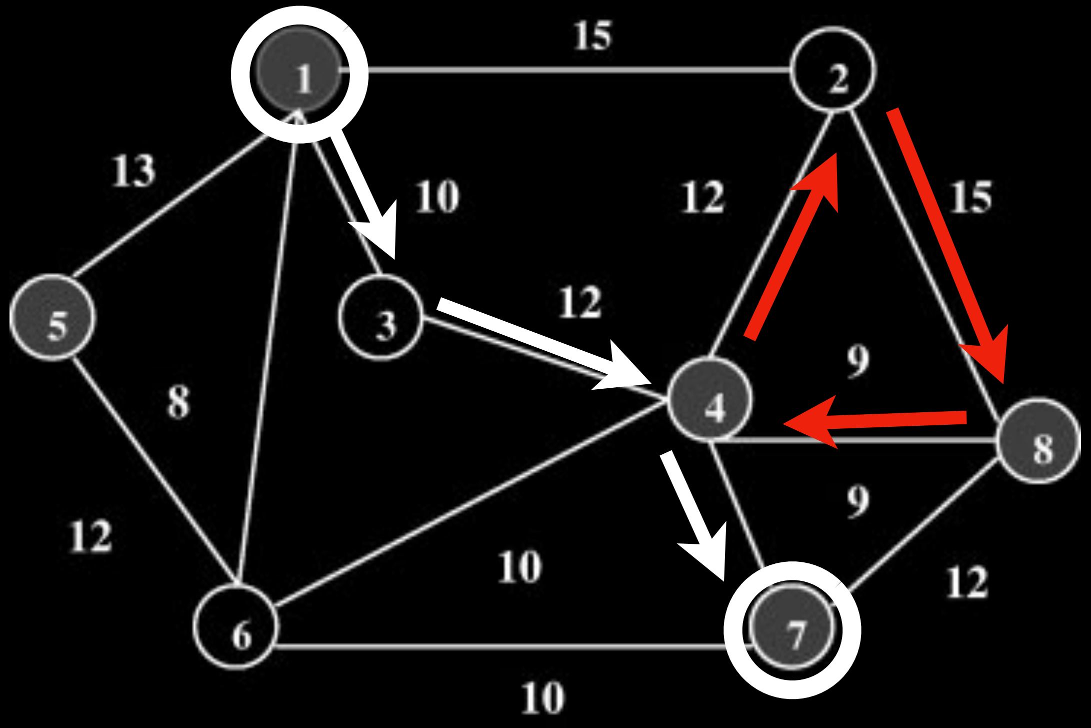
(b) No sharing

Cycle Check

- Cycle may appear after crossover and mutation
- Discard cycled path

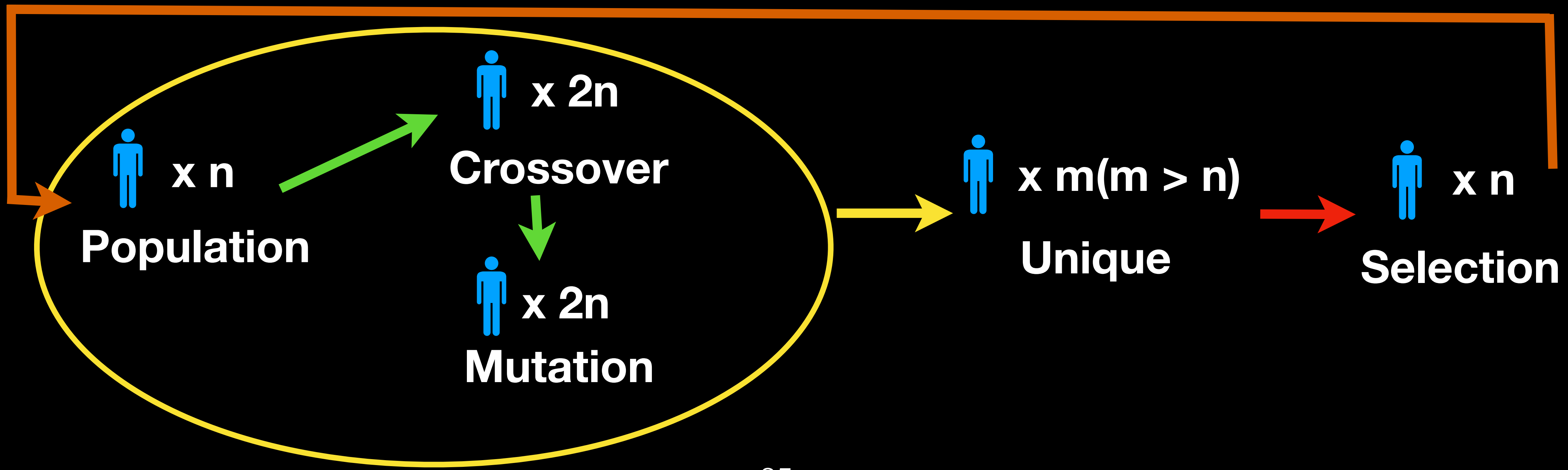
1 3 4 2 8 4 7

1 3 4 7



Survival Selection (μ+λ)

1. Population with n individuals
2. 100% crossover generates 2n individuals
3. 50% mutation generates another n individuals
4. Find all the unique individuals from 1 ~ 3
5. Elitism truncation and choose n individuals

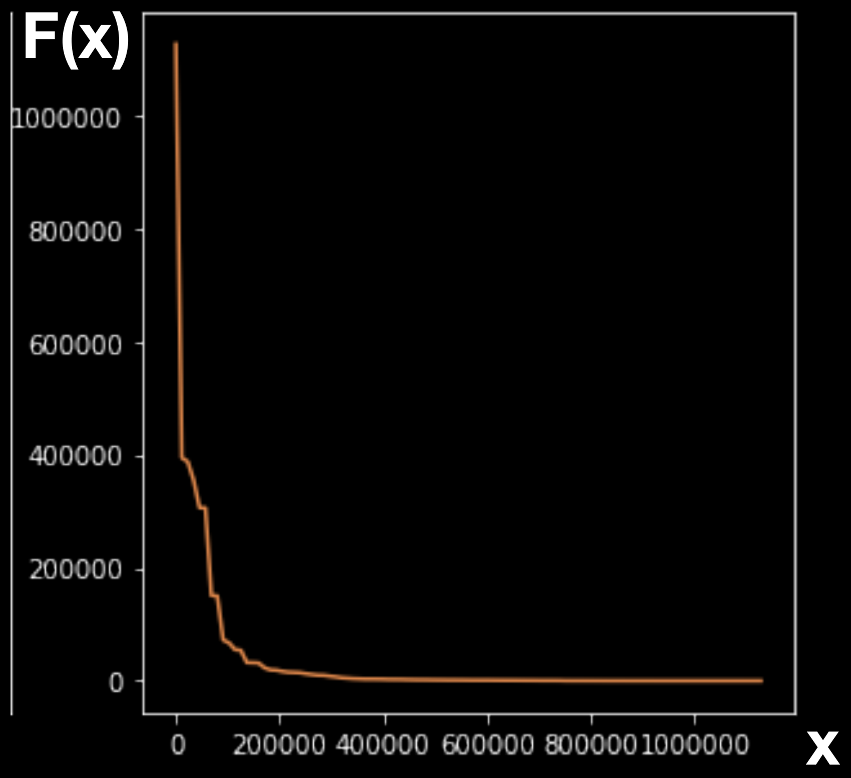


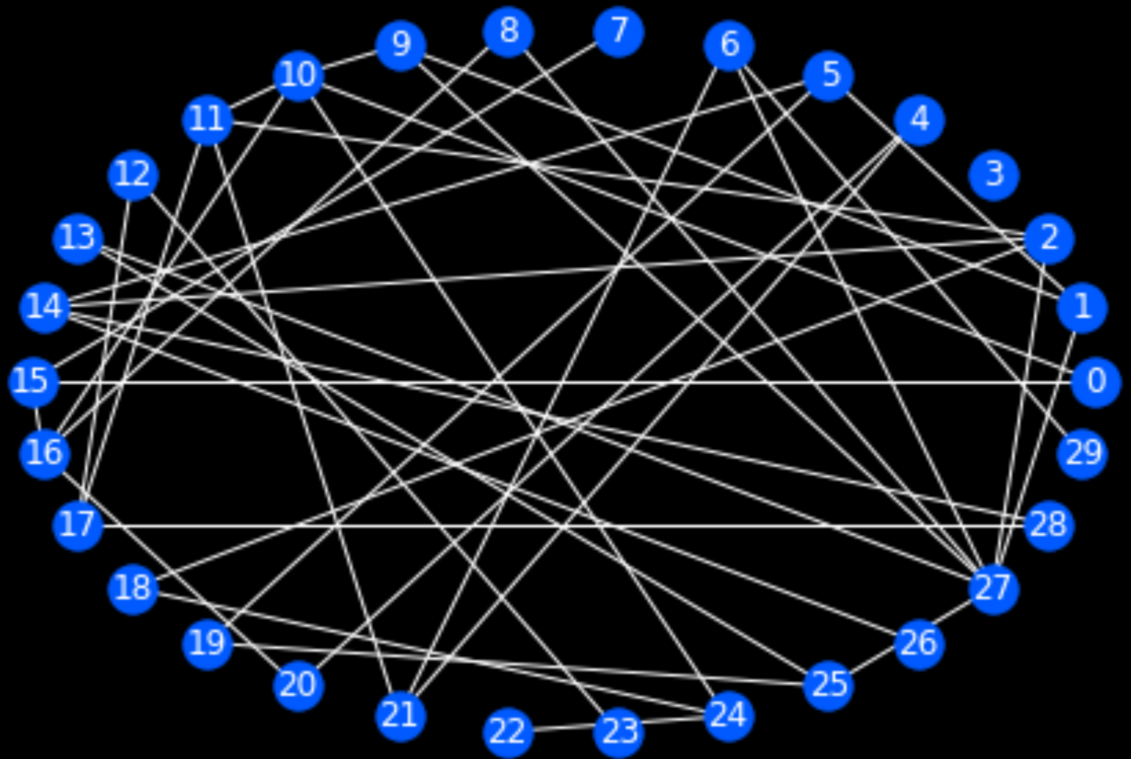
Analysis

Benchmark

- Generate multiple sparse networks
  - Edge creation probability is 0.1
  - Weight assigned is based on F-distribution with d1=d2=1

F distribution

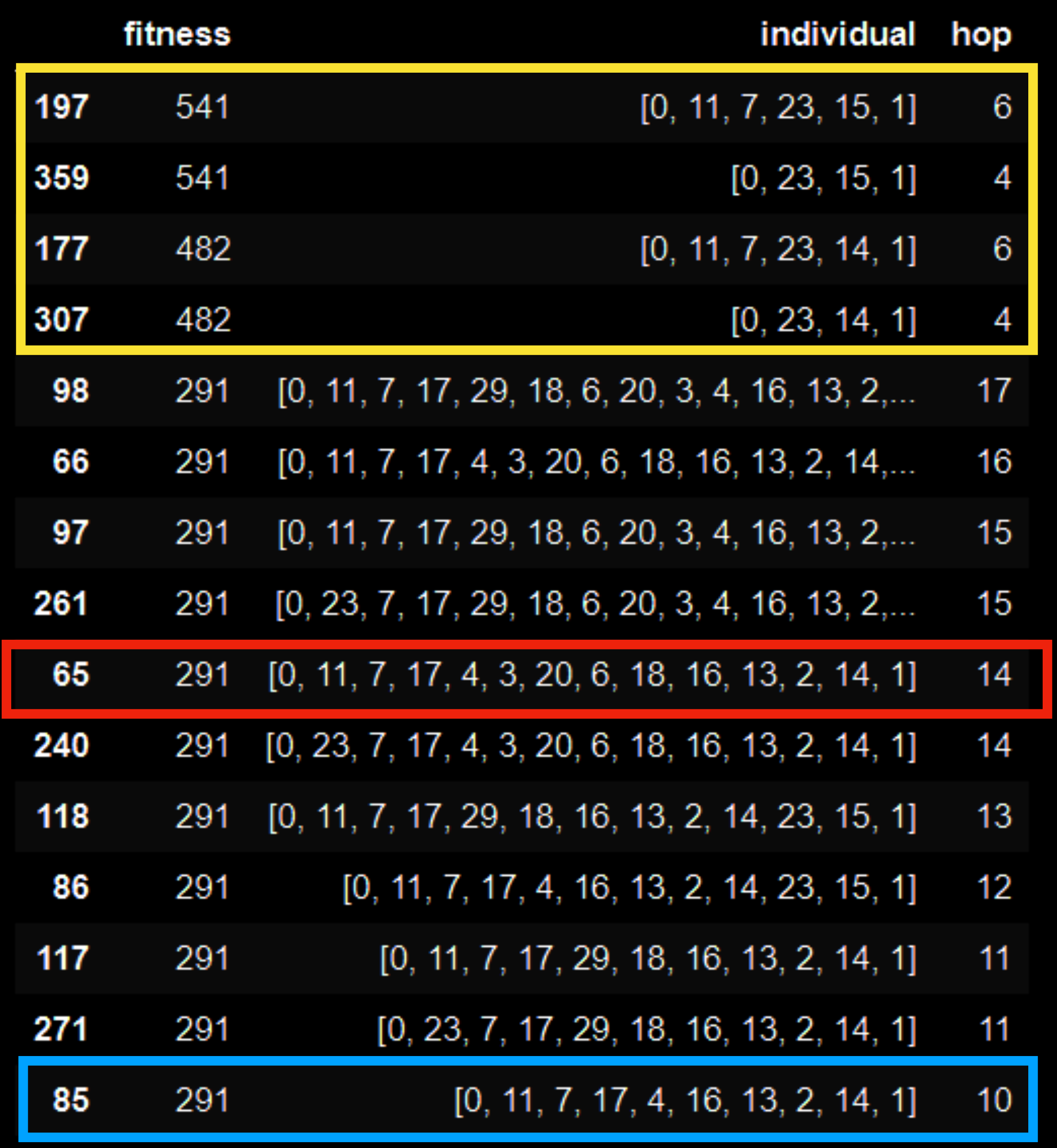




Generated network
G(n, p), n=30, p=0.1

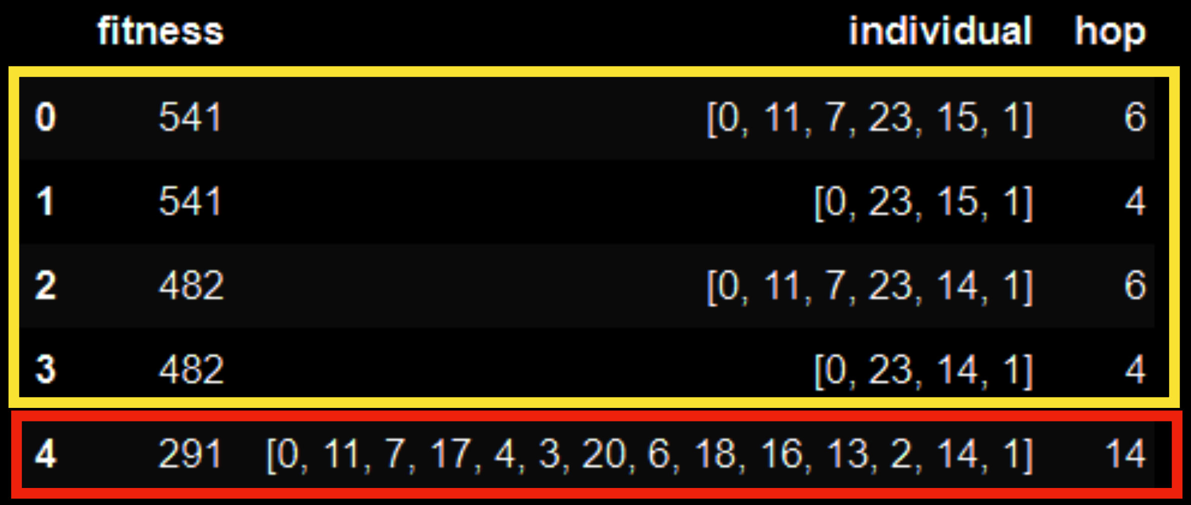
[7] Vladimir Batagelj and Ulrik Brandes, “Efficient generation of large random networks” , Phys. Rev. E, 71, 036113, 2005.

Result



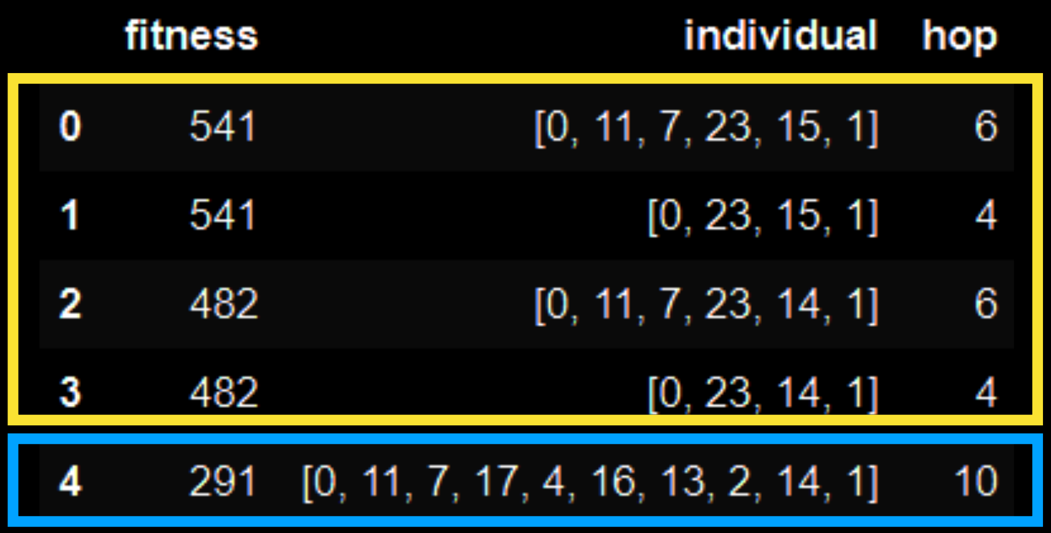
	fitness	individual	hop
197	541	[0, 11, 7, 23, 15, 1]	6
359	541	[0, 23, 15, 1]	4
177	482	[0, 11, 7, 23, 14, 1]	6
307	482	[0, 23, 14, 1]	4
98	291	[0, 11, 7, 17, 29, 18, 6, 20, 3, 4, 16, 13, 2,...	17
66	291	[0, 11, 7, 17, 4, 3, 20, 6, 18, 16, 13, 2, 14,...	16
97	291	[0, 11, 7, 17, 29, 18, 6, 20, 3, 4, 16, 13, 2,...	15
261	291	[0, 23, 7, 17, 29, 18, 6, 20, 3, 4, 16, 13, 2,...	15
65	291	[0, 11, 7, 17, 4, 3, 20, 6, 18, 16, 13, 2, 14, 1]	14
240	291	[0, 23, 7, 17, 4, 3, 20, 6, 18, 16, 13, 2, 14, 1]	14
118	291	[0, 11, 7, 17, 29, 18, 16, 13, 2, 14, 23, 15, 1]	13
86	291	[0, 11, 7, 17, 4, 16, 13, 2, 14, 23, 15, 1]	12
117	291	[0, 11, 7, 17, 29, 18, 16, 13, 2, 14, 1]	11
271	291	[0, 23, 7, 17, 29, 18, 16, 13, 2, 14, 1]	11
85	291	[0, 11, 7, 17, 4, 16, 13, 2, 14, 1]	10

Result using DFS



	fitness	individual	hop
0	541	[0, 11, 7, 23, 15, 1]	6
1	541	[0, 23, 15, 1]	4
2	482	[0, 11, 7, 23, 14, 1]	6
3	482	[0, 23, 14, 1]	4
4	291	[0, 11, 7, 17, 4, 3, 20, 6, 18, 16, 13, 2, 14, 1]	14

Result using GA w/ population of 200



	fitness	individual	hop
0	541	[0, 11, 7, 23, 15, 1]	6
1	541	[0, 23, 15, 1]	4
2	482	[0, 11, 7, 23, 14, 1]	6
3	482	[0, 23, 14, 1]	4
4	291	[0, 11, 7, 17, 4, 16, 13, 2, 14, 1]	10

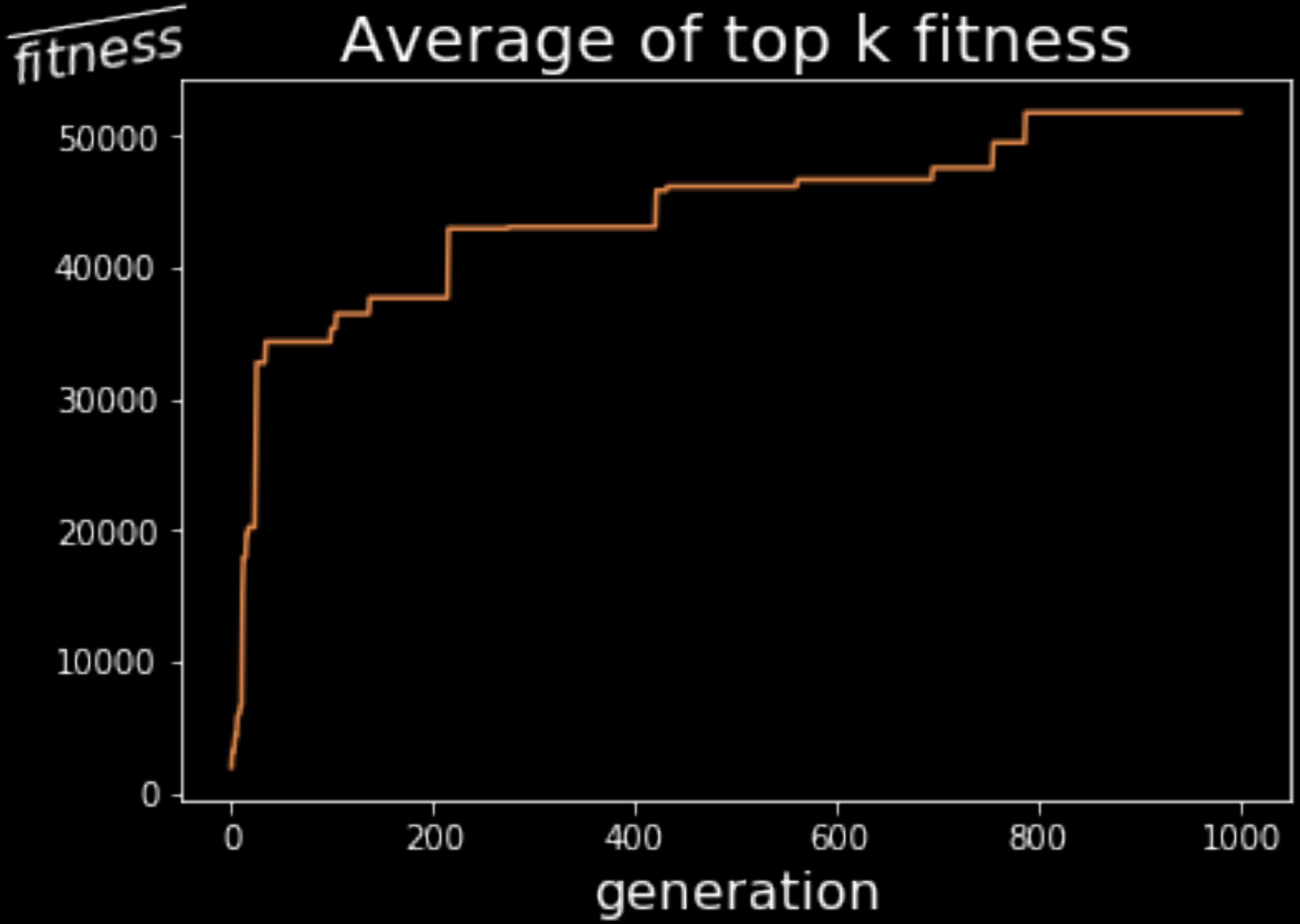
Result using GA w/ population of 50

Done on a 30 nodes sparse network. Larger network is not feasible for DFS

Benchmark

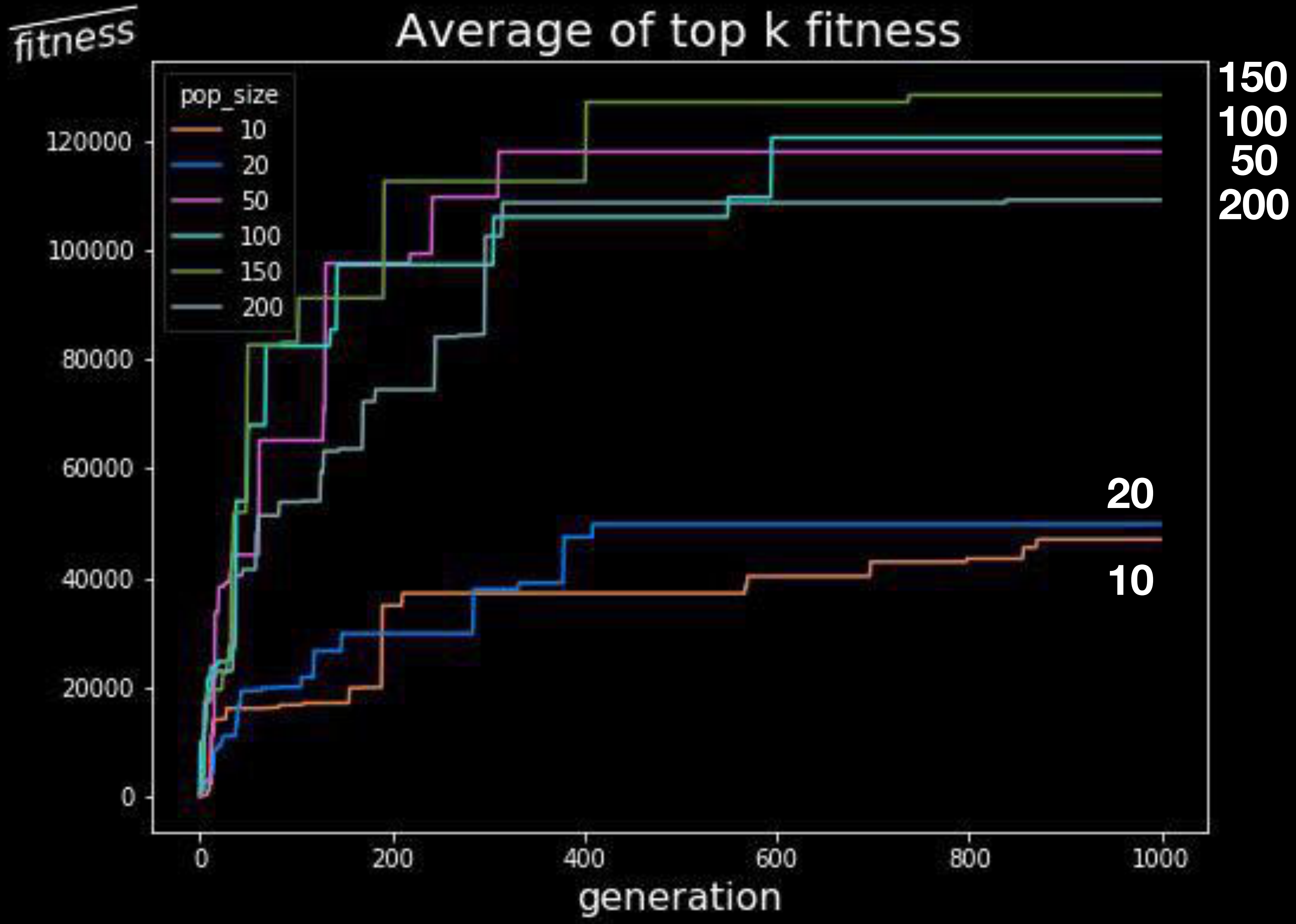
- Create network in different scales
  - Total nodes: 10, 30, 100, 1000, 10000
- Use DFS for exhaustive search
  - Find the top k shortest path and compare with our work

Result


fitness
Average of top k fitness
50000
40000
30000
20000
10000
0
0
200
400
600
800
1000
generation

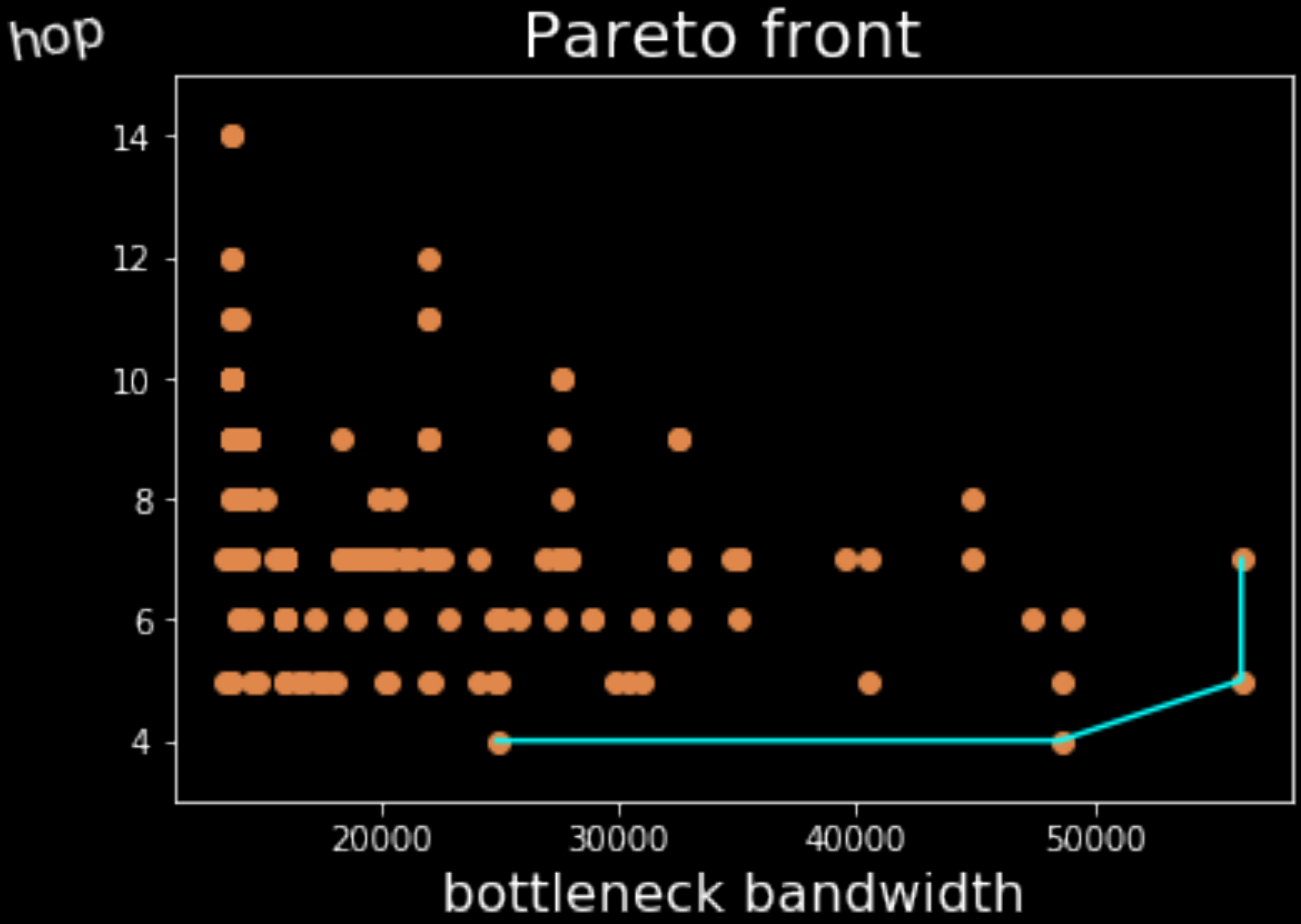
Population Size

- Large population size has better result



Pareto Front

- Pareto front represents the optimal solutions

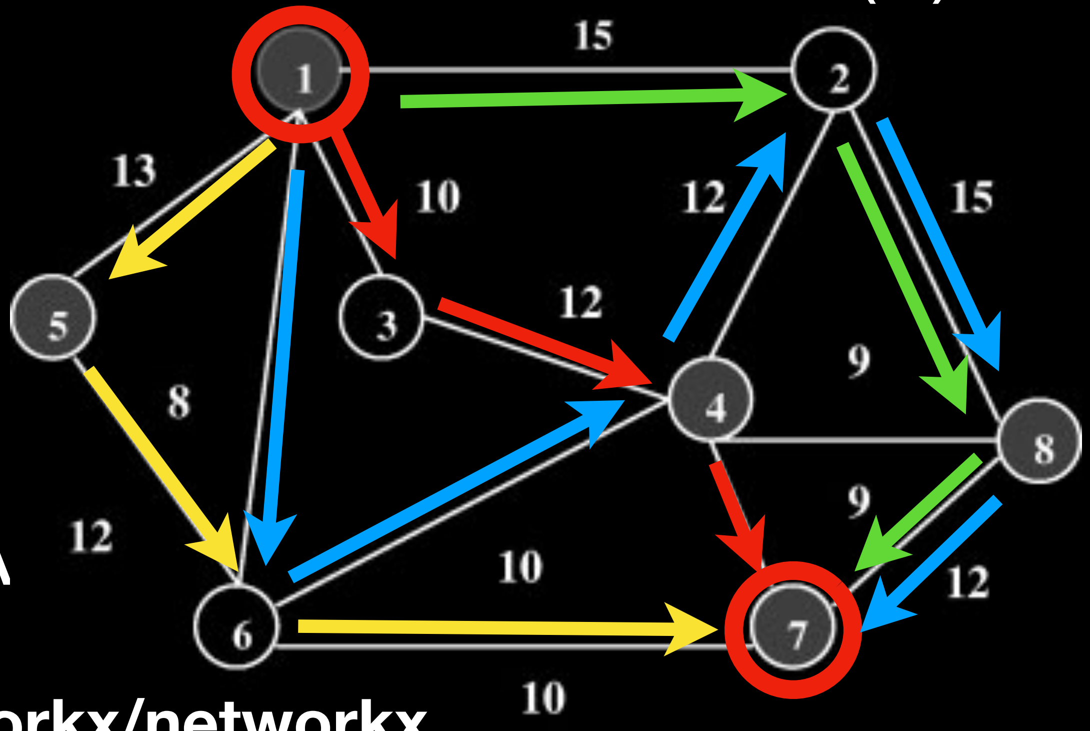


Conclusion

Q & A

Old Initialization

- Find the shortest path from source to destination based on the hop needed and temporary ignore weight(RGY)
  - Shortest path can be achieved using less hops in most of cases
- Randomly choose legal path from source to destination(B)
- Mixed them as population
- Make sure it does not contain loop
- Population is too good to use GA



Python3 networkx: https://github.com/networkx/networkx