GENETIC ALGORITHM

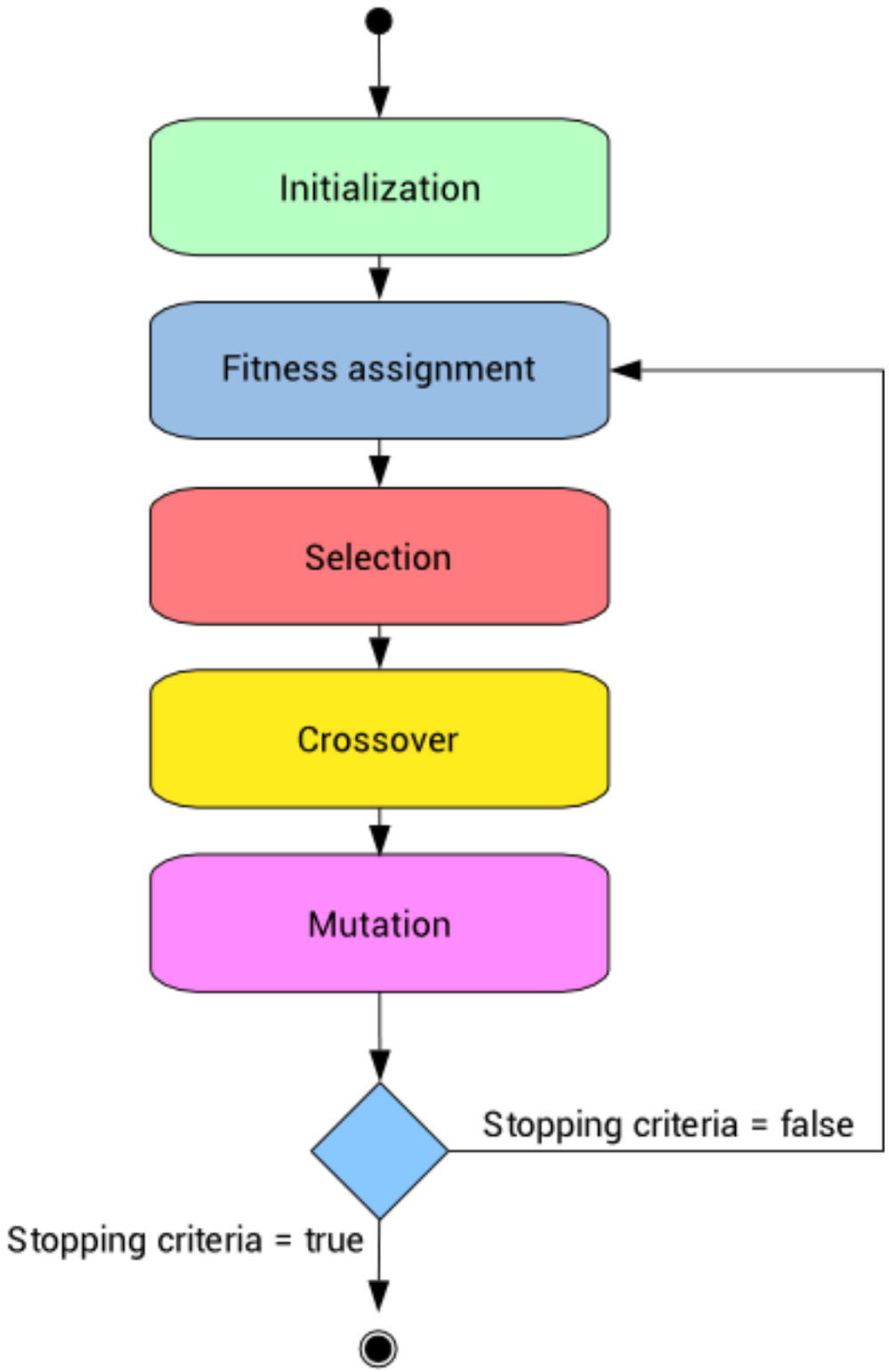
SHORTEST PATH
TRAVELING SELF-MAN

Người thực hiện: Trần Ngọc Bảo Duy - 51702091
Phạm Anh Duy - 51702088

1. Genetic Algorithm

Khái niệm

- Giải thuật di truyền là một kỹ thuật của khoa học máy tính nhằm tìm kiếm giải pháp thích hợp cho các bài toán tối ưu tổ hợp (combinatorial optimization).
- Giải thuật di truyền là một phân ngành của giải thuật tiến hóa vận dụng các nguyên lý của tiến hóa như di truyền, đột biến, chọn lọc tự nhiên, và trao đổi chéo.
- Ngày nay, giải thuật di truyền được dùng phổ biến trong một số ngành như tin sinh học, khoa học máy tính, trí tuệ nhân tạo, …



2. Phương pháp

- Trong giải thuật di truyền cho một bài toán tối ưu hóa, ý tưởng là một tập hợp (gọi là nhiễm sắc thể) của những giải pháp có thể (gọi là cá thể) sẽ được cho "tiến triển" theo hướng chọn lọc những giải pháp tốt dần hơn.

- While(true):
  - Chọn ra cá thể có fitness tốt từ tập cá thể hiện tại để làm cá thể lai tạo
  - Lai tạo (crossover) hoặc đột biến (mutate) các cá thể với nhau để sinh ra tập thể thế hệ kế tiếp
  - Kiểm tra thế hệ vừa sinh ra đã đạt yêu cầu hoặc tốt hơn hay chưa
    - Nếu đạt, hoặc trong qúa trình tổ hợp vẫn sinh ra kết quả cũ mà không có tiến triển mới thì ngừng vòng lặp
    - Nếu không tập hợp này sẽ tiếp tục được chọn lọc lặp đi lặp lại trong các thế hệ kế tiếp của giải thuật

3. Mô tả

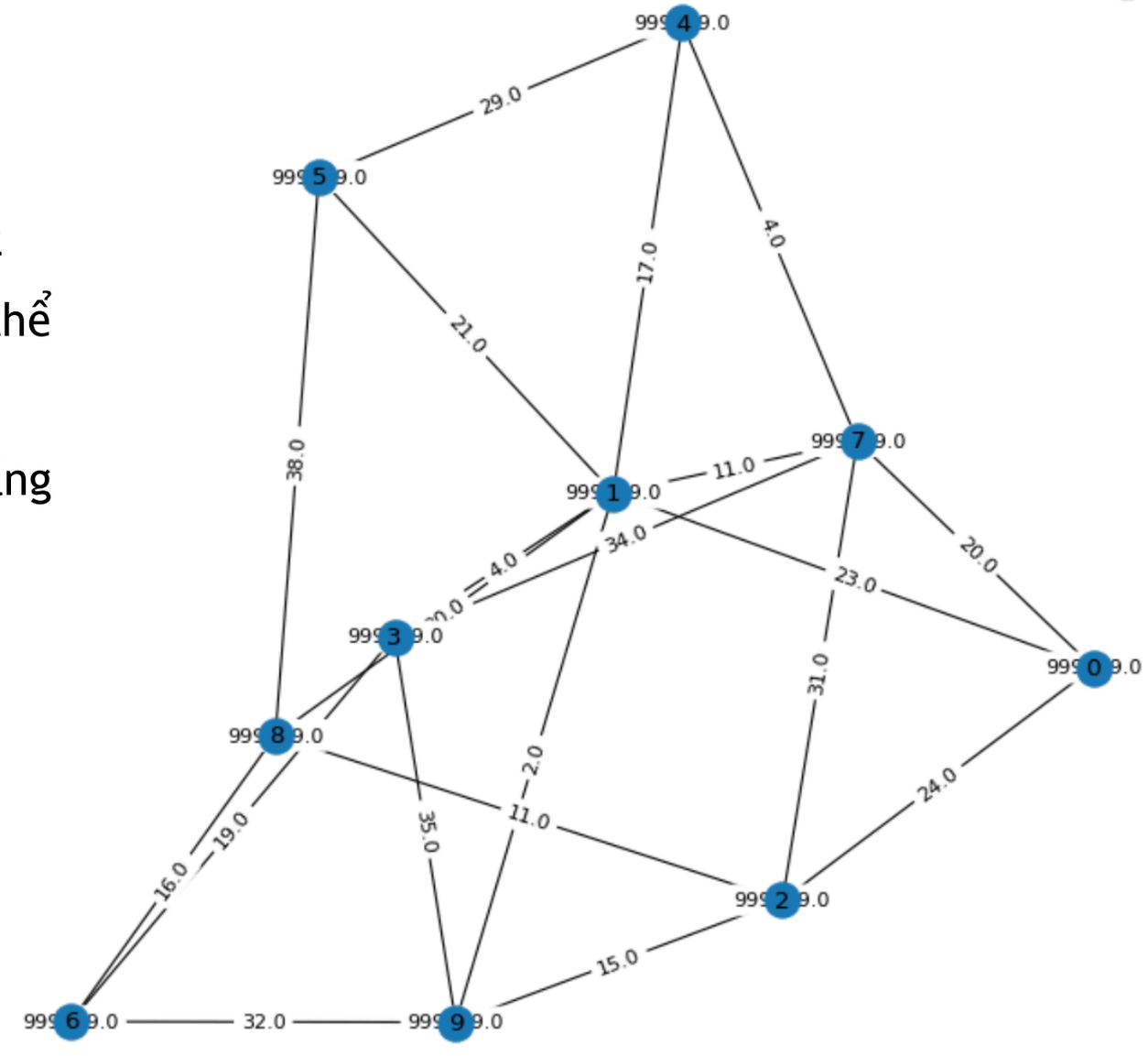
- Cá thể: một đường đi từ điểm bắt đầu cho đến điểm kết thúc
- Quần thể: tập hợp nhiều cá thể lại với nhau
- Phương pháp tạo mới (cross-over):

- B1: Khởi tạo quần thể với các cá thể là một đường đi, có thể khởi tạo 100 hay 1000 cá thể tuỳ vào người sử dụng.
- Các cá thể này phải được tạo ra **<u>hợp lí</u>**, chẳng hạn: đi từ 6 -> 0 thì ta có

◉ 6 -> 9 -> 1 -> 0

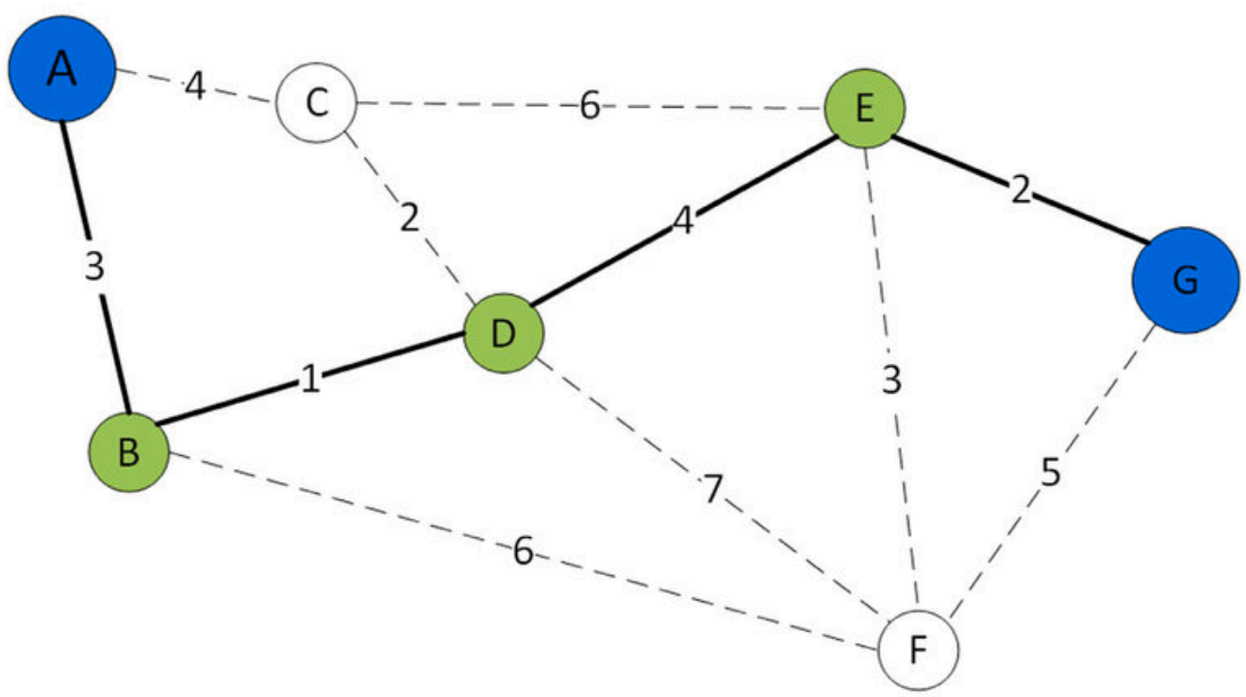
◉ 6 -> 9 -> 3 -> 8 -> 5 -> 1 -> 0

◉ 6 -> 9 -> 3 -> 1 -> 4 -> 7 -> 2 -> 0



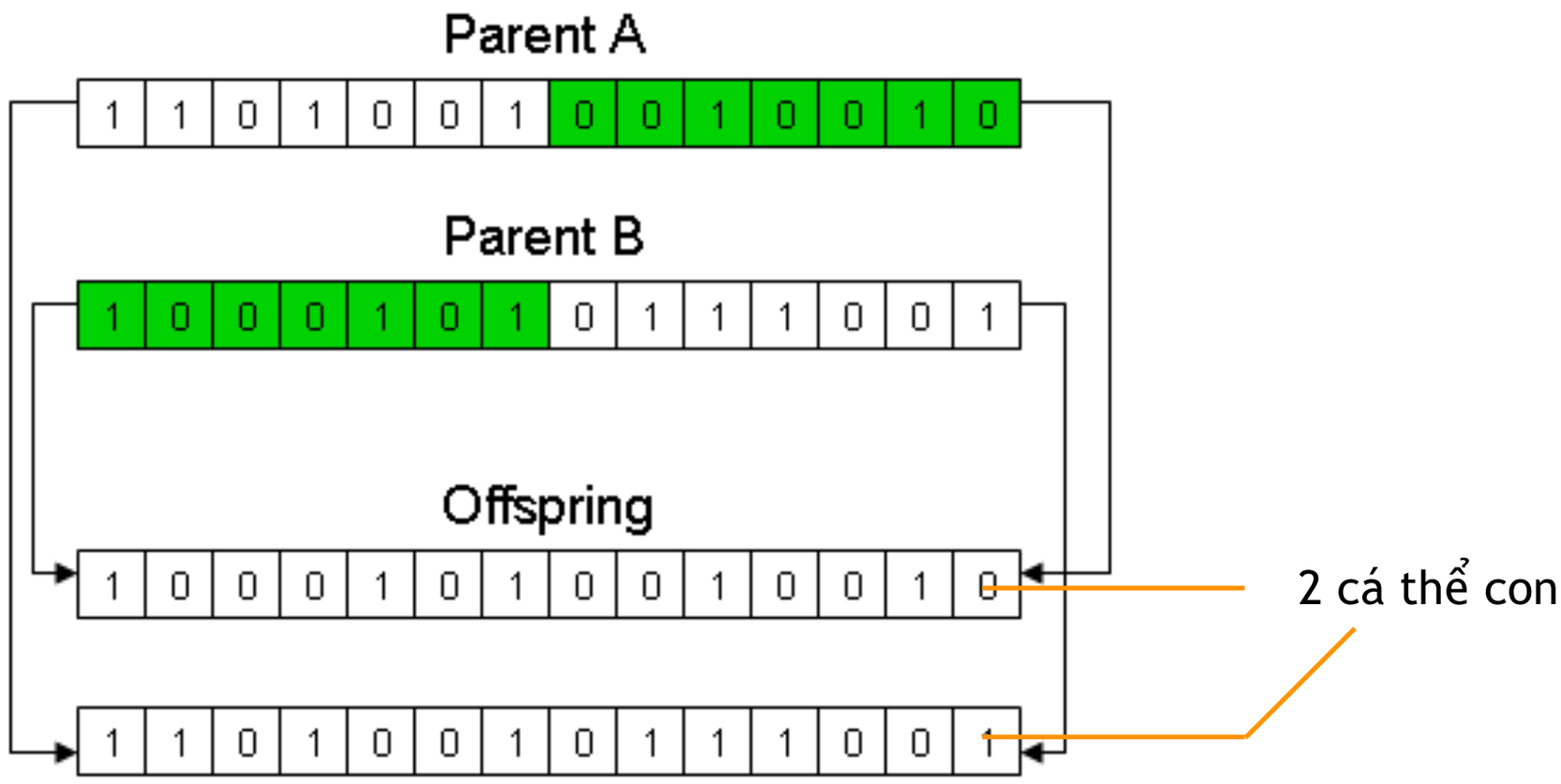
2. Mô tả

- B2: Chọn ngẫu nhiên 2 cá thể tốt nhất trong quần thể gọi là bố mẹ. Đối với từng bài toán thì ta sẽ có cách tính toán để đánh giá được cá thể nào có fitness tốt nhất.
- Trong bài toán cụ thể tìm đường đi ngắn nhất thì thuật toán đánh giá phải đáp ứng được cá thể được chọn có đường đi là ngắn nhất ***tức là giá trị tiêu hao từ điểm A -> B càng ít thì fitness càng cao***
  - Có nhiều thuật toán tìm đường đi ngắn nhất như: dijkstra, ford-bellman, Depth-First Search (DFS) , Breadth-First Search (BFS).



2. Mô tả

- Phương pháp tạo mới (cross-over) single point:
  - B3: tìm điểm giống nhau giữa 2 cá thể bố mẹ, chọn ngẫu nhiên 1 điểm và gắn nữa đầu là cá thể bố và nữa sau là cá thể mẹ, với cách như vậy có thể sinh ra 2 cá thể con



Single Point Crossover

2. Mô tả

- Phương pháp tạo mới (cross-over) 2 point:
  - B3: Two point crossover: chúng ta sẽ tìm ra 2 điểm và cắt ra sau đó trộn lẫn nó với nhau để tạo ra hai cá thể mới như hình bên dưới.

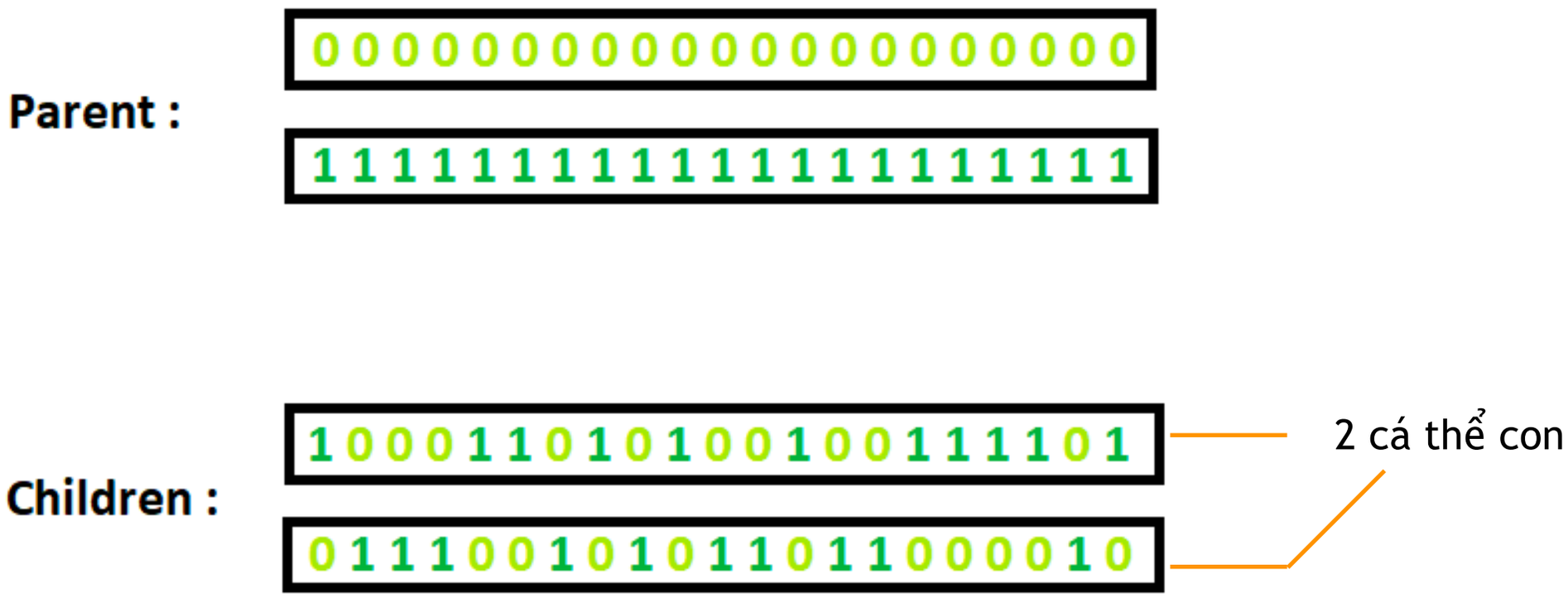
Chromosome1	11011|00100|110110
Chromosome2	10101|11000|011110
Offspring1	11011|11000|110110
Offspring2	10101|00100|011110

2 cá thể con

Two Point Crossover

2. Mô tả

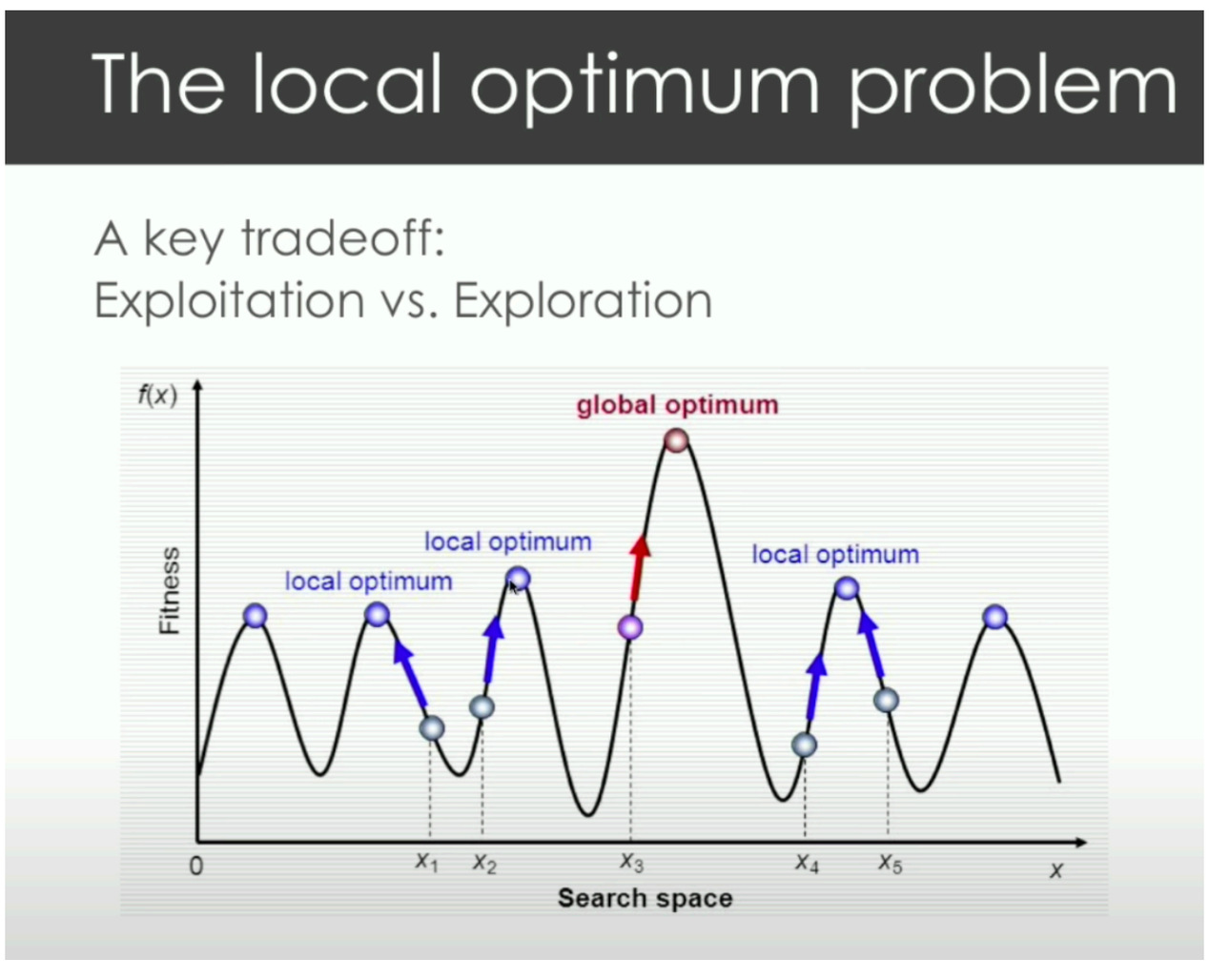
- Phương pháp tạo mới (cross-over) uniform:
  - B3: Uniform crossover: chúng ta sẽ tìm ra các cặp tại vị trí của 2 bố mẹ sau đó tráo cho nhau, cứ thế cho hết chuỗi ADN



Uniform Crossover

2. Mô tả

- Phương pháp tạo mới (cross-over) Mutation:
  - B3: Mutation: là một phương pháp biến dị, phương pháp này giống như một cách tiến hoá của quần thể lên một thế hệ tốt hơn, hoặc có thể xuống 1 thế hệ với fitness xấu hơn. => mục đích làm cho quần thể không bị **bão hoà**



2. Mô tả

- B4: Cập nhật lại quẩn thể từ các cá thể được lai tạo mới. Kiểm tra xem quần thể đã chứa các cá thể mục tiêu hay không ? Hoặc kiểm tra quẩn thể có bị bão hoà hay không
  - ➡ Nếu quần thể chứa cá thể mục tiêu hoặc bị bão hoà thì tới bước 5.
  - ➡ Nếu không thì quay lại bước 2
- B5: Tìm được mục tiêu, kết thúc qúa trình evolution

Remember: This is a heuristic approach! It will NOT always find the best solution (the 'global optimum')

*Source: Skiena, The Algorithm Design Manual

Time Complexities (if 1 op = 1 ns)

input size n	log n	n	n·log n	n^2	2^n	n!
10	0.003 µs	0.01 µs	0.03 µs	0.1 µs	1 µs	3.63 ms
20	0.004 µs	0.02 µs	0.09 µs	0.4 µs	1 ms	77.1 years
30	0.005 µs	0.03 µs	0.15 µs	0.9 µs	1 sec	8.4×10^{15} yrs
40	0.005 µs	0.04 µs	0.21 µs	1.6 µs	18.3 min	

3. Ứng dụng - Shortest path

```
def dijsktra(self):
    visited = {self.start_point: 0}
    path = {}
    nodes = set(self.edges)
    while nodes:
        min_node = None
        for node in nodes:
            if int(node) in visited:
                if min_node is None:
                    min_node = int(node)
                elif visited[int(node)] < visited[min_node]:
                    min_node = int(node)
        if min_node is None:
            break

        nodes.remove(int(min_node))
        current_weight = visited[min_node]

        for edge in self.edges[min_node]:
            weight = current_weight + self.distances[(int(min_node), int(edge))]
            if edge not in visited or weight < visited[edge]:
                visited[edge] = weight
                path[edge] = min_node

    self.path = path
    self.visited = visited
    return visited, path
```

- Hàm dijkstra dùng tìm đường đi ngắn nhất

3. Ứng dụng - Shortest path

```
def trace(self, is_print=False):
    self.create_list_edge()
    self.dijsktra()
    if is_print: self.print_matrix_edge()

    cost = self.visited[self.end_point]
    p = list()
    p.append(self.end_point)
    while self.path[p[-1]] != self.start_point:
        p.append(self.path[p[-1]])

    p.append(self.start_point)
    with open(self.shortest_path,'w') as fout:
        st = ""
        for item in reversed(p):
            st += str(item)+"->"
        fout.write(st[:-2]+"\n")
        fout.write(str(cost))
    return list(reversed(p))
```

- Hàm trace để biết được đường đi ngắn nhất là những node nào.
- Hàm trace_cross_all_point tìm hết tất cả đường có thể đi từ điểm A tới điểm B **nhưng là đường đi ngắn nhất**

```
def trace_cross_all_point(self, is_print=False):
  print("Target là ",self.start_point,self.end_point)
  print("Tìm tất cả đường đi từ đầu đến đích có số node bằng với tổng số node (",self.NUM_MATRIX ,") cần đi qua")
  self.get_all_path = list(self.dfs_paths(self.start_point, self.end_point))
  filter_NUM_MATRIX = list(filter(lambda x: len(x) == self.NUM_MATRIX ,self.get_all_path))
  if len(filter_NUM_MATRIX) == 0:
    filter_NUM_MATRIX = list(filter(lambda x: len(x) >= self.NUM_MATRIX -1 ,self.get_all_path))
  sorted_NUM_MATRIX = list(sorted(filter_NUM_MATRIX, key=lambda y: self.get_value_of_path(y)))
  return sorted_NUM_MATRIX[0]
```

3. Ứng dụng - Shortest path

```
def print_all_path_M(self, u, d, visited, path):
  visited[u]= True
  path.append(u)
  if u == d:
      self.gene_path.append(path.copy())
  else:
      for i in self.edges[u]:
          if visited[i]==False:
              self.print_all_path_M(i, d, visited, path)
  path.pop()
  visited [u]= False
```

```
def find_all_path(self, from_edge, to_edge):
  visited =[False]*self.NUM_MATRIX
  path = []
  self.print_all_path_M(from_edge, to_edge, visited, path)
  return self.gene_path
```

```
def get_value_of_path(self, path):
  value = 0
  for index, item in enumerate(path[:-1]):
    v = self.matrix_edge[item,path[index+1]]
    if v > 0:
      value+= v
    else:
      return -999
  return value
```

- Dùng phương pháp DFS, duyệt theo chiều sau để có thể tìm hết tất cả đường đi từ điểm A -> B
- Hàm get_value_of_path() trả về giá trị của một đường đi từ A-> B

3. Ứng dụng - Shortest path

```python
class Schedule():

  def __init__(self, path):
    self.path = path
    self.index_like = []

  def tim_diem_giong(self, lover_path):
    index_like = []
    if len(self.path) <= len(lover_path):
      for index, item in enumerate(self.path):
        if item == lover_path[index]:
          index_like.append(index)
    else:
      for index, item in enumerate(lover_path):
        if item == self.path[index]:
          index_like.append(index)
    self.index_like = index_like[1:-2].copy()

  def lai_tao(self, lover, len_target, method):
    new_path = []
    self.tim_diem_giong(lover.path)
    if len(self.index_like) > 0:
      split_point = random.choice(self.index_like)
      if method == "single":
        new_path.extend(self.path[:split_point])
        new_path.extend(lover.path[split_point:])
      # if method == "two":
      # if method == "unifrom":
      return Schedule(new_path)
    else:
      return Schedule(self.path)
```

✦ **<u>Cấu trúc một cá thể ADN:</u>**

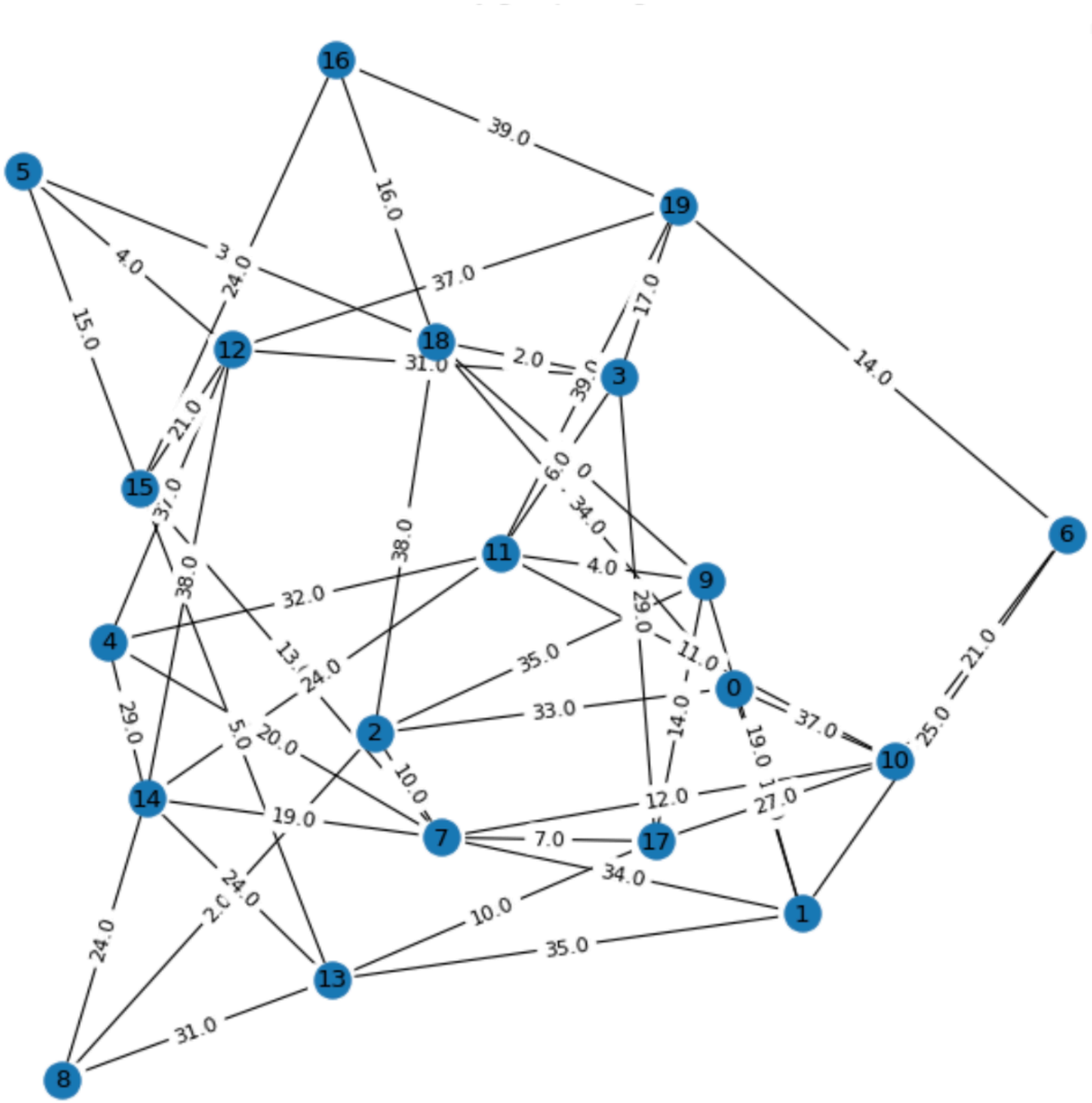
- Path
- Hàm tìm ra điểm giống nhau
- Hàm lai tạo (crossover)
  - Single point
  - Two point
  - Uniform

3. Ứng dụng - Shortest path

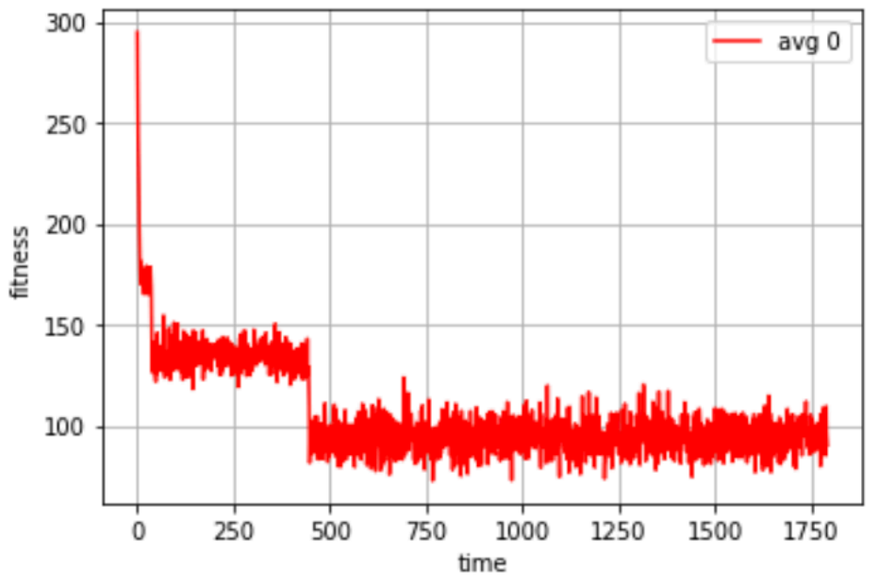
✦ **<u>Cấu trúc một quần thể:</u>**

- Có n population (n cá thể)
- Hàm khởi tạo quần thể ban đầu
- Hàm đánh giá từng cá thể
- Hàm fitness - đánh giá toàn bộ cá thể trong quần thể (Heuristic function)
- Hàm tiến hoá 1 lần (evolute one time)
  - Có các mode chọn parent khác nhau:
    - Chọn ngẫu nhiên trong quần thể
    - Chọn ngẫu nhiên trong nữa quần thể có fitness cao
    - Chọn hai cá thể có fitness tốt nhất
  - Các mode lai tạo:
    - Single point
    - Two point
    - Uniform
    - Mutation (biến dị)
- Hàm tiến hoá (quá trình tiến hoá)

3. Ứng dụng - Shortest path

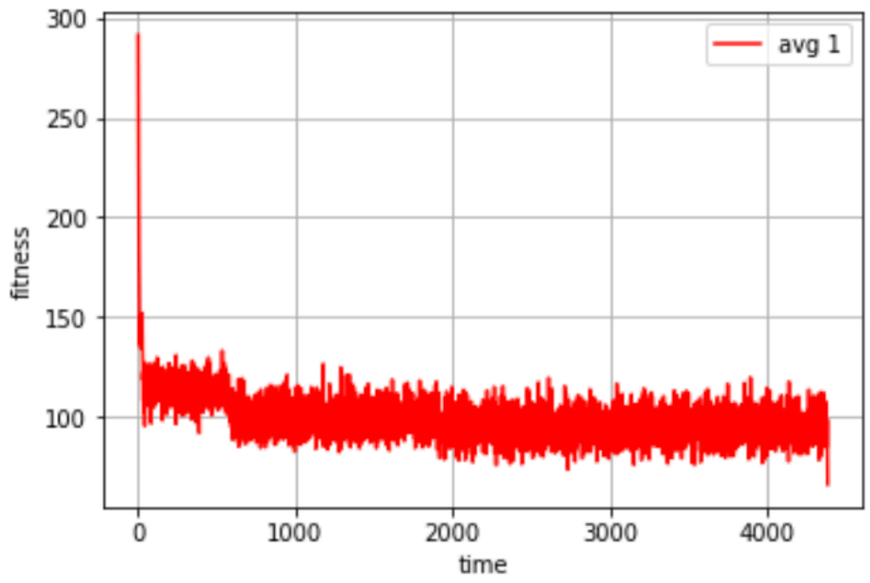


3. Ứng dụng - Shortest path




```
Graph gồm 20 địa điểm

Phướng pháp chon lọc là: choose_half_best
Mục tiêu cá thể tốt nhất là: [4, 7, 17, 13] 37.0
Khởi tạo vật liệu di truyền thành công
Tìm tất cả đường đi từ đầu đến đích,tổng cộng 549381 đường (cá thể) có thể lai từ đây
Khởi tạo đường đi hoàn tất tổng số cá thể trong quần thể là 1000
Khởi tạo quần thể hoàn tất
        Evolution 1: New PATH = [4, 7, 2, 9, 1, 13], value 119.0
                Mutation number is: 114
        Evolution 8: New PATH = [4, 11, 10, 17, 13], value 80.0
                Mutation number is: 110
        Evolution 441: New PATH = [4, 7, 15, 13], value 38.0
                Mutation number is: 99
        Evolution 1791: New PATH = [4, 7, 17, 13], value 37.0
                Mutation number is: 71
```

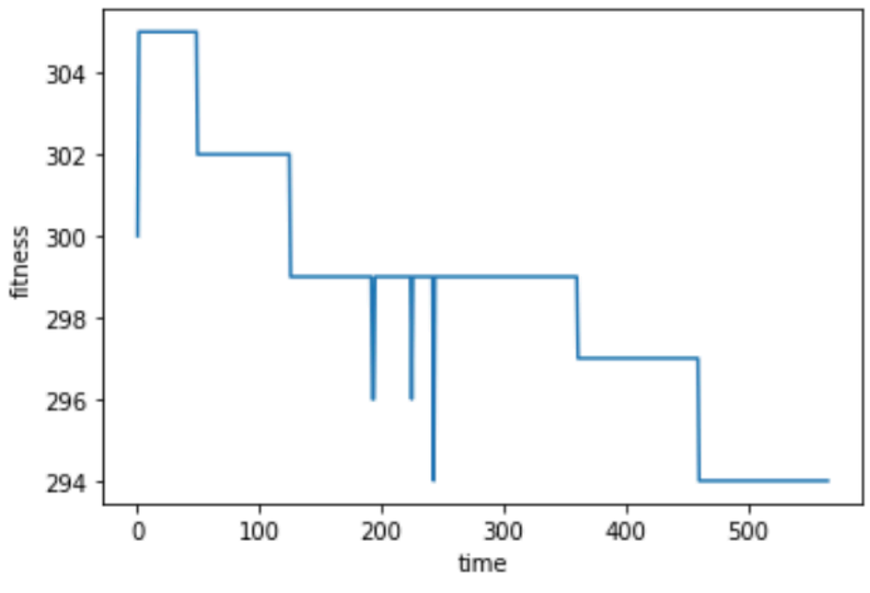



```
Graph gồm 20 địa điểm

Phướng pháp chon lọc là: choose_half_best
Mục tiêu cá thể tốt nhất là: [18, 5, 15] 18.0
Khởi tạo vật liệu di truyền thành công
Tìm tất cả đường đi từ đầu đến đích,tổng cộng 336881 đường (cá thể) có thể lai từ đây
Khởi tạo đường đi hoàn tất tổng số cá thể trong quần thể là 1000
Khởi tạo quần thể hoàn tất
        Evolution 1: New PATH = [18, 9, 11, 14, 7, 15], value 99.0
                Mutation number is: 125
        Evolution 7: New PATH = [18, 3, 11, 9, 1, 13, 17, 7, 15], value 96.0
                Mutation number is: 115
        Evolution 25: New PATH = [18, 3, 11, 14, 7, 15], value 64.0
                Mutation number is: 108
        Evolution 559: New PATH = [18, 3, 12, 15], value 54.0
                Mutation number is: 92
        Evolution 578: New PATH = [18, 3, 17, 13, 15], value 46.0
                Mutation number is: 98
        Evolution 1291: New PATH = [18, 3, 11, 10, 7, 15], value 44.0
                Mutation number is: 93
        Evolution 1909: New PATH = [18, 16, 15], value 40.0
                Mutation number is: 117
        Evolution 2983: New PATH = [18, 5, 12, 15], value 28.0
                Mutation number is: 107
        Evolution 4387: New PATH = [18, 5, 15], value 18.0
                Mutation number is: 26
```

---

3. Ứng dụng - Traveling self-man

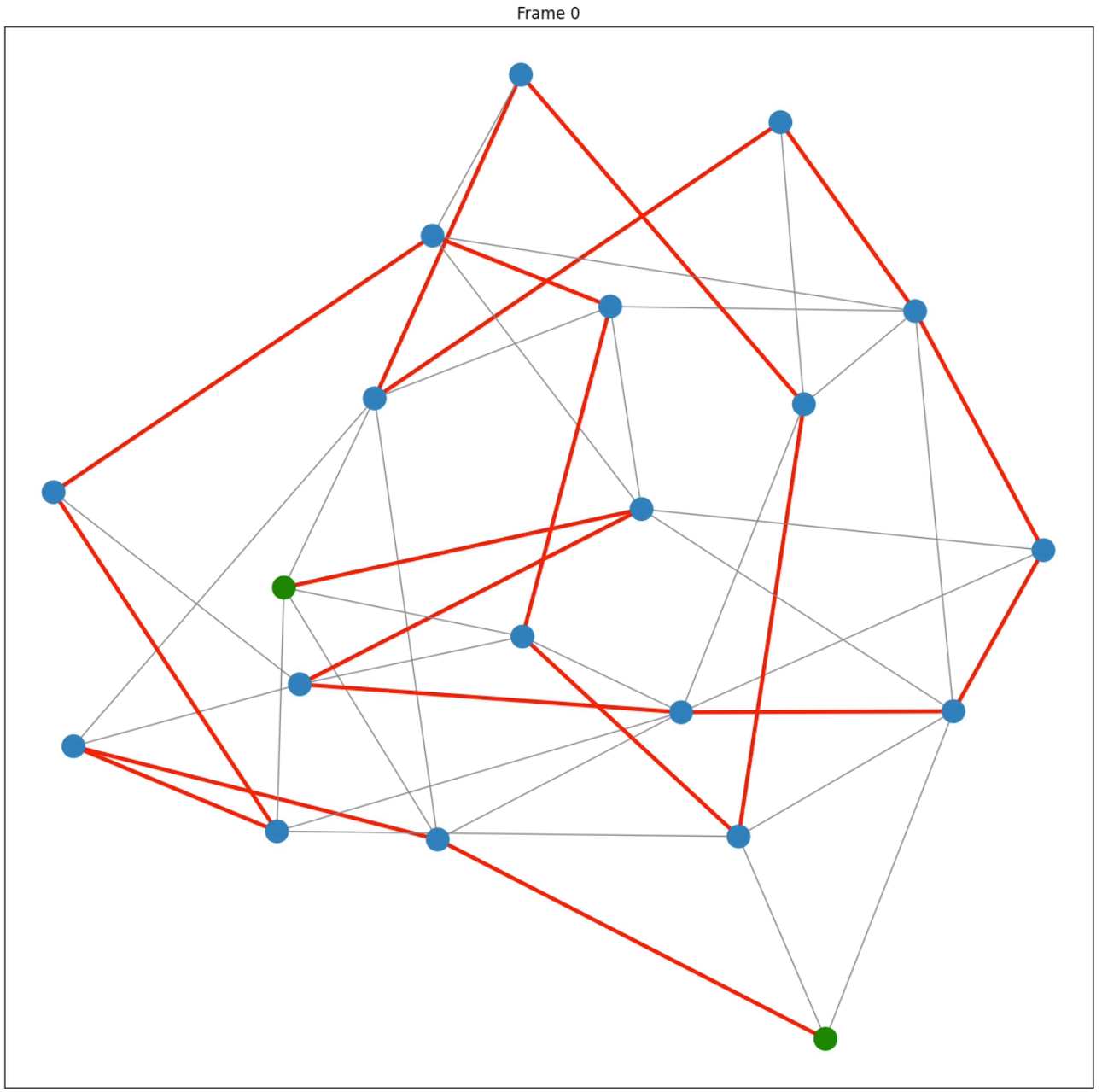



```
Target là  8 9
Tìm tất cả đường đi từ đầu đến đích có số node bằng với tổng số node ( 20 ) cần đi qua
Graph gồm 20 địa điểm

Phướng pháp chon lọc là: choose_half_best
Mục tiêu cá thể tốt nhất là: [8, 2, 0, 1, 6, 19, 3, 11, 10, 7, 14, 4, 12, 5, 18, 16, 15, 13, 17, 9] 292.0
Khởi tạo vật liệu di truyền thành công
Tìm tất cả đường đi từ đầu đến đích,tổng cộng 10363 đường (cá thể) có thể lai từ đây
Khởi tạo đường đi hoàn tất tổng số cá thể trong quần thể là 2000
Khởi tạo quần thể hoàn tất
        Evolution 1: New PATH = [8, 2, 0, 1, 6, 19, 3, 17, 13, 15, 16, 18, 5, 12, 4, 14, 7, 10, 11, 9], value 305.0
                Mutation number is: 10
        Evolution 49: New PATH = [8, 2, 0, 1, 6, 19, 16, 15, 13, 14, 4, 12, 5, 18, 3, 11, 10, 7, 17, 9], value 302.0
                Mutation number is: 17
        Evolution 125: New PATH = [8, 2, 7, 17, 13, 14, 4, 12, 5, 15, 16, 18, 3, 19, 6, 1, 0, 10, 11, 9], value 299.0
                Mutation number is: 17
        Evolution 192: New PATH = [8, 2, 7, 17, 13, 15, 16, 18, 5, 12, 4, 14, 11, 3, 19, 6, 10, 0, 1, 9], value 296.0
                Mutation number is: 16
        Evolution 242: New PATH = [8, 2, 0, 1, 6, 19, 3, 11, 10, 7, 4, 14, 12, 5, 18, 16, 15, 13, 17, 9], value 294.0
                Mutation number is: 18
        Evolution 565: New PATH = [8, 2, 0, 1, 6, 19, 3, 11, 10, 7, 14, 4, 12, 5, 18, 16, 15, 13, 17, 9], value 292.0
                Mutation number is: 6
```

---

3. Ứng dụng - Traveling self-man



3. Ứng dụng - Shortest path

✦ **<u>Cấu trúc một quần thể:</u>**

- Có n population (n cá thể)
- Hàm khởi tạo quần thể ban đầu
- Hàm đánh giá từng cá thể
- Hàm fitness - đánh giá toàn bộ cá thể trong quần thể (Heuristic function)
- Hàm tiến hoá 1 lần (evolute one time)
  - Có các mode chọn parent khác nhau:
    - Chọn ngẫu nhiên trong quần thể
    - Chọn ngẫu nhiên trong nữa quần thể có fitness cao
    - Chọn hai cá thể có fitness tốt nhất
  - Các mode lai tạo:
    - Single point
    - Two point
    - Uniform
    - Mutation (biến dị)
- Hàm tiến hoá (quá trình tiến hoá)

4. Reference

Genetic Algorithm – Thuật toán di truyền

https://quangnle.com/genetic-algorithm-thuat-toan-di-truyen/

Genetic Algorithms - Jeremy Fisher

https://www.youtube.com/watch?v=7J-DfS52bnI

Genetic Algorithm Tutorial - How to Code a Genetic Algorithm

https://www.youtube.com/watch?v=XP8R0yzAbdo

Genetic Algorithms - Explanation & Implementation

https://www.youtube.com/watch?v=9bht7Vq0DqY