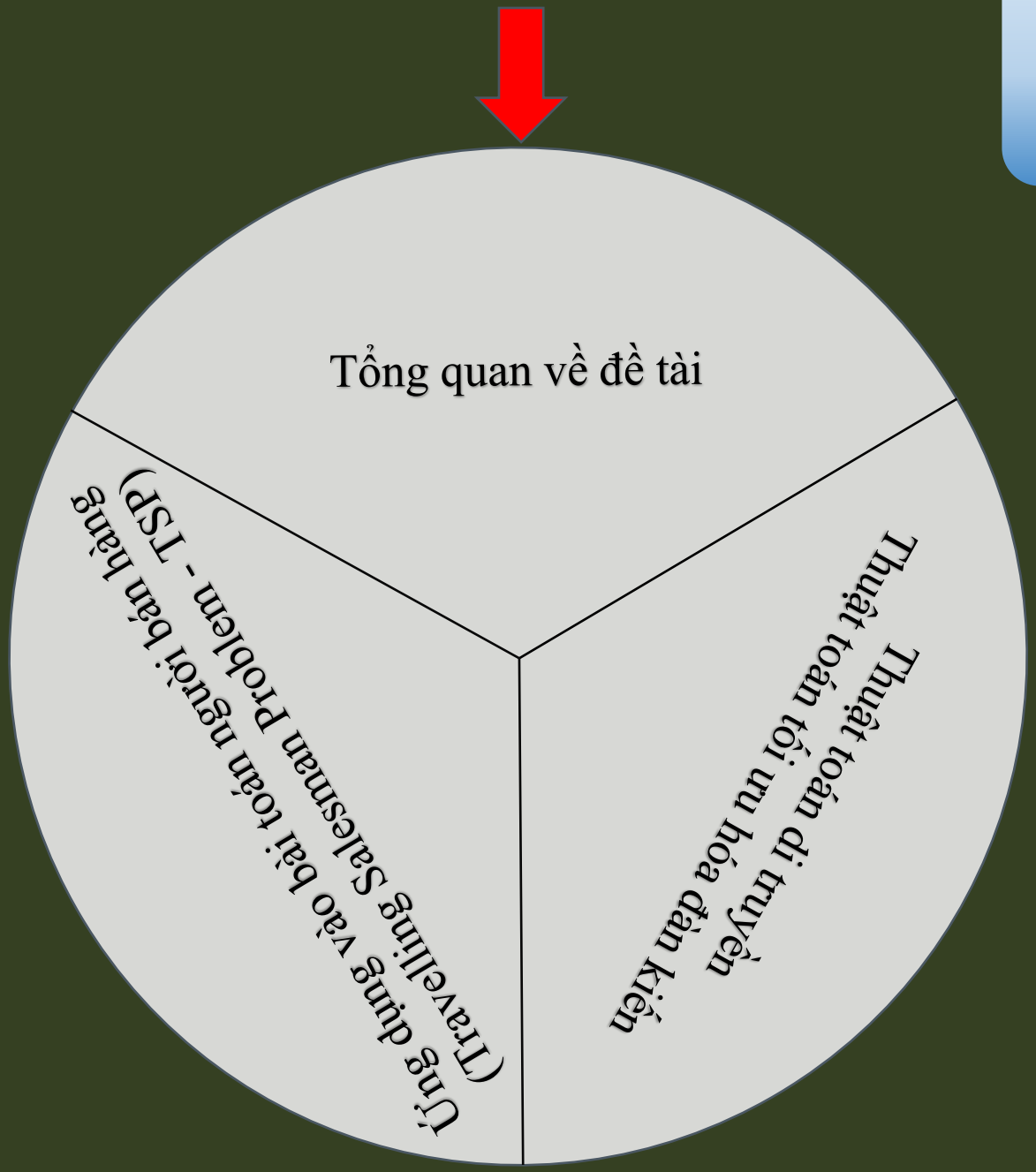
CHÀO MỪNG THẦY CÔ ĐẾN VỚI BÀI THUYẾT TRÌNH CỦA EM

THUẬT TOÁN MÔ PHỎNG TỰ NHIÊN GIẢI BÀI TOÁN TỐI ƯU TỔ HỢP

Các ý chính buổi bảo vệ



Các ý chính buổi bảo vệ

- Các thành phần chính
- Mã giả
- Độ phức tạp
- Bài toán điển hình

Các ý chính buổi bảo vệ

Ý tưởng thực hiện

Thực nghiệm

Đánh giá

TỔNG QUAN VỀ ĐỀ TÀI

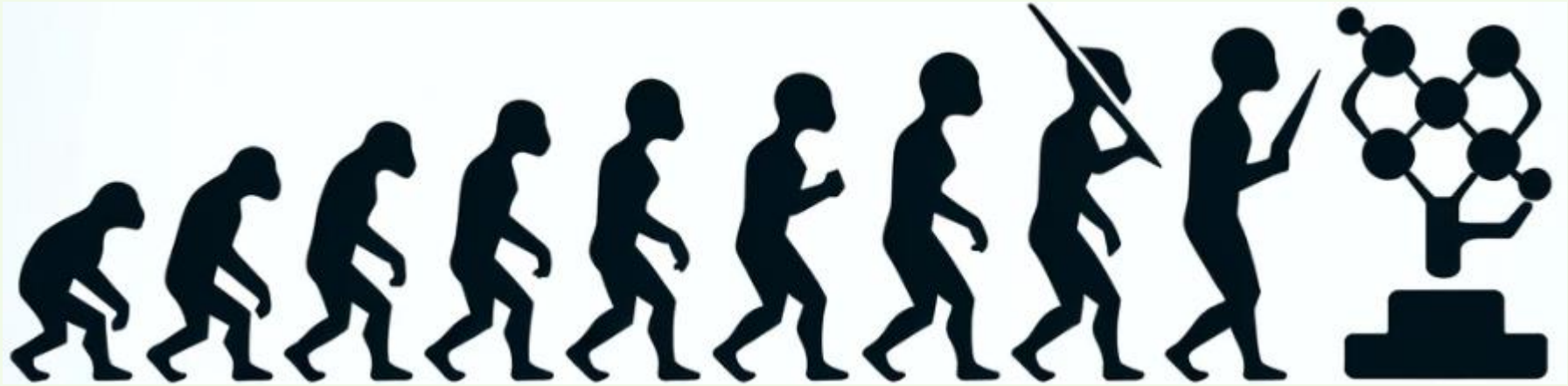
Thuật toán mô phỏng tự nhiên là nhóm các phương pháp tính toán bắt chước các quy trình và hệ thống tự nhiên để giải quyết các vấn đề trong toán học và khoa học máy tính.

Ứng dụng rộng rãi trong nhiều lĩnh vực như tối ưu hóa, nhận dạng mẫu, xử lý hình ảnh, dự báo thời tiết, phân tích dữ liệu sinh học, và hơn thế nữa. Chúng đặc biệt hữu ích trong các bài toán tối ưu hóa tổ hợp và liên tục mà không gian tìm kiếm quá lớn hoặc phức tạp.

Các thuật toán mô phỏng tự nhiên

- Thuật toán di truyền (Genetic Algorithms - GA)
- Thuật toán ưu hóa đàn kiến (Ant Colony Optimization – ACO)
- Tối ưu hóa bầy đàn (Particle Swarm Optimization – PSO
- Thuật toán hệ miễn dịch nhân tạo (Artificial Immune Systems – AIS)
- Thuật toán tiến hóa vi phân (Differential Evolution – DE)
- Thuật toán tối ưu hóa bọ cánh cứng (Beetle Antennae Search - BAS).
- Thuật toán bầy chim (Flock Algorithms)

Thuật toán di truyền (Genetic Algorithms - GA)

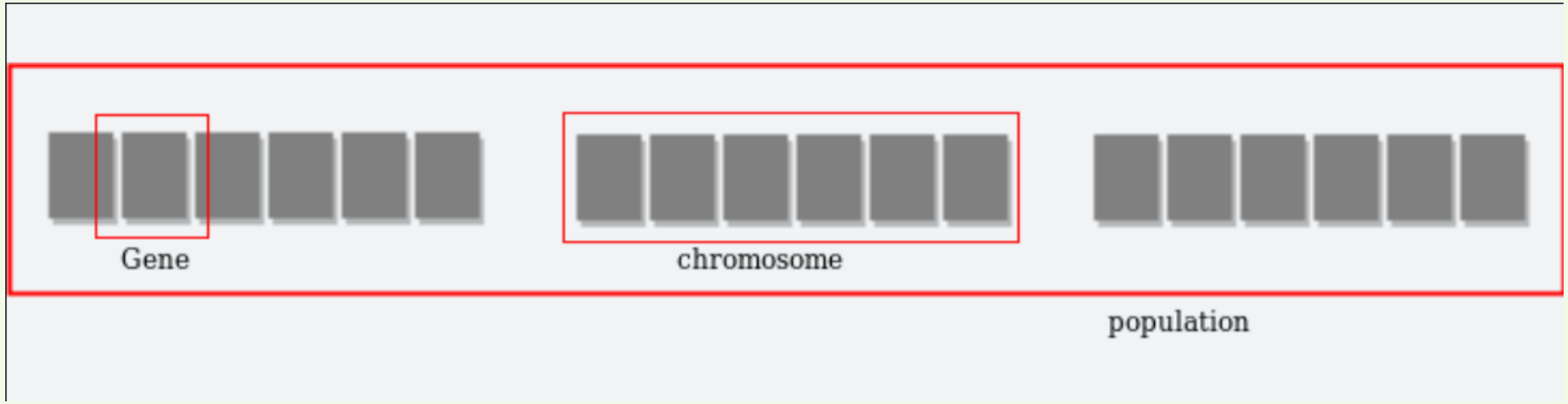


1

Các thành phần chính

1. Mã hóa (Encoding)
2. Khởi tạo quần thể (Population Initialization)
3. Hàm thích nghi (Fitness Function)
4. Chọn lọc (Selection)
5. Lai ghép (Crossover)
6. Đột biến (Mutation)

Mã hóa là quá trình biểu diễn các biến quyết định của bài toán thành một chuỗi, thường được gọi là chromosome.



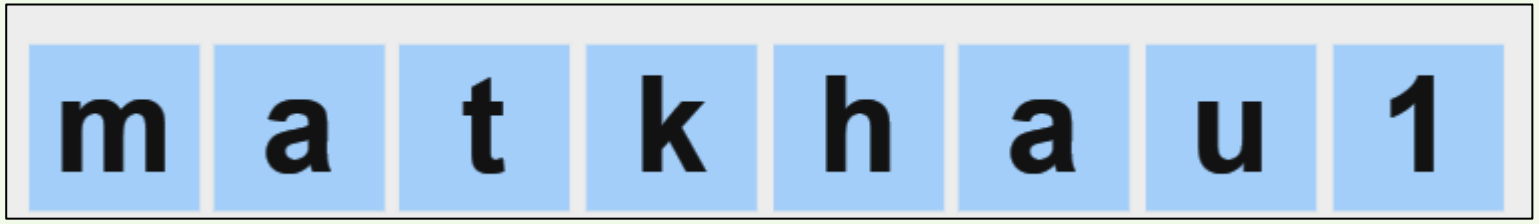
Source: https://www.geeksforgeeks.org/genetic-algorithms/?ref=header_search

1

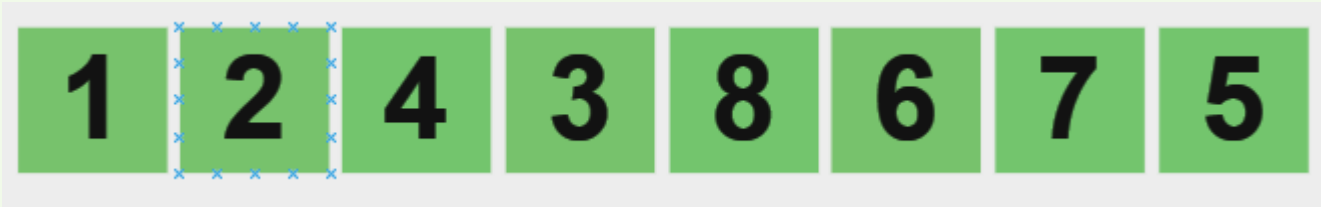
Các thành phần chính

1. Mã hóa (Encoding)
2. Khởi tạo quần thể (Population Initialization)
3. Hàm thích nghi (Fitness Function)
4. Chọn lọc (Selection)
5. Lai ghép (Crossover)
6. Đột biến (Mutation)

Phổ biến nhất là mã hóa nhị phân, nhưng còn có mã hóa số nguyên, số thực, và mã hóa dựa trên đối tượng cho các bài toán phức tạp hơn



Mã hóa gen thành ký tự cho cá thể



Mã hóa gen thành số cho cá thể

Cụ thể như bài này là tìm đường đi ngắn nhất với gen là các đỉnh

Source: https://drive.google.com/file/d/1AzUInPAe_jdiX3Xp4ggHPvXV-d2pKQRJ/view?usp=sharing

1

Các thành phần chính

1. Mã hóa (Encoding)
2. Khởi tạo quần thể (Population Initialization)
3. Hàm thích nghi (Fitness Function)
4. Chọn lọc (Selection)
5. Lai ghép (Crossover)
6. Đột biến (Mutation)

Quần thể ban đầu của GA thường được sinh ra một cách ngẫu nhiên để đảm bảo sự đa dạng. Quần thể gồm nhiều cá thể, mỗi cá thể đại diện cho một giải pháp tiềm năng.

1

Các thành phần chính

1. Mã hóa (Encoding)
2. Khởi tạo quần thể (Population Initialization)
3. Hàm thích nghi (Fitness Function)
4. Chọn lọc (Selection)
5. Lai ghép (Crossover)
6. Đột biến (Mutation)

Hàm thích nghi là hàm đánh giá mức độ "thích hợp" của mỗi cá thể trong quần thể, thường dựa trên chất lượng của giải pháp mà cá thể đó biểu diễn.

1

Các thành phần chính

1. Mã hóa (Encoding)
2. Khởi tạo quần thể (Population Initialization)
3. Hàm thích nghi (Fitness Function)
4. Chọn lọc (Selection)
5. Lai ghép (Crossover)
6. Đột biến (Mutation)

Chọn lọc là quá trình chọn ra cá thể từ quần thể hiện tại để tạo ra cá thể cho quần thể thế hệ tiếp theo.

1

Các thành phần chính

1. Mã hóa (Encoding)
2. Khởi tạo quần thể (Population Initialization)
3. Hàm thích nghi (Fitness Function)
4. Chọn lọc (Selection)
5. Lai ghép (Crossover)
6. Đột biến (Mutation)

Lai ghép là quá trình kết hợp đặc điểm của hai cá thể cha mẹ để tạo ra cá thể con.

1

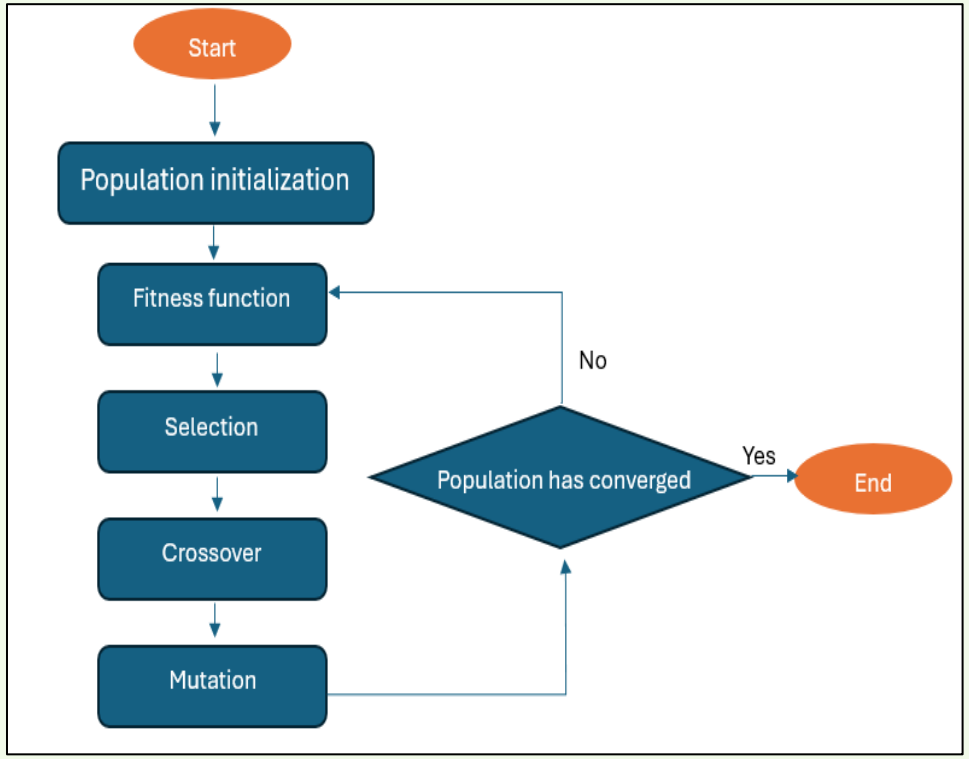
Các thành phần chính

1. Mã hóa (Encoding)
2. Khởi tạo quần thể (Population Initialization)
3. Hàm thích nghi (Fitness Function)
4. Chọn lọc (Selection)
5. Lai ghép (Crossover)
6. Đột biến (Mutation)

Đột biến là quá trình làm thay đổi gen nhằm tạo cho quần thể đa dạng về mã gen

2

Mã giả



Start

Population Initialization

REPEAT:

    Fitness Function

    Selection

    Crossover

    Mutation

Until: Population has converged

Stop

3

Độ phức tạp

1. Độ phức tạp thời gian
2. Độ phức tạp không gian

Độ phức tạp về thời gian của GA thường được xem xét qua ba yếu tố chính: Kích thước quần thể (N), số lượng thế hệ (G), và độ phức tạp của hàm đánh giá (f):

$$O(N \times G \times f)$$

Độ phức tạp về không gian phụ thuộc vào kích thước quần thể và biểu diễn của nhiễm sắc thể, thường biểu thị qua $O(N \times s)$, với s là không gian lưu trữ cần thiết cho mỗi cá thể.

4

Bài toán điển hình:

Bài toán tìm mật khẩu

Xét bài toán Tìm mật khẩu với các yêu cầu sau:

- Mật khẩu gồm 8 kí tự, bao gồm chữ cái, chữ số và khoảng trắng. Ví dụ mật khẩu: matkhau1
- Mỗi lần thử, hệ thống sẽ báo về số lượng kí tự đúng với mật khẩu.
- Yêu cầu tìm ra chuỗi mật khẩu đã cho trước

Khởi tạo quần thể bất kỳ với chiều dài mỗi mật khẩu là 8 ký tự:

Với đầu vào là các chữ cái, chữ số và khoảng trắng.

Khởi tạo các giá trị

n_Population : Số lượng cá thể trong quần thể. Giả sử n_Population = 7

n_Generation : Số lần thế hệ được tạo ra. Giả sử n_Generation = 1

4

Bài toán điển hình:

Bài toán tìm mật khẩu

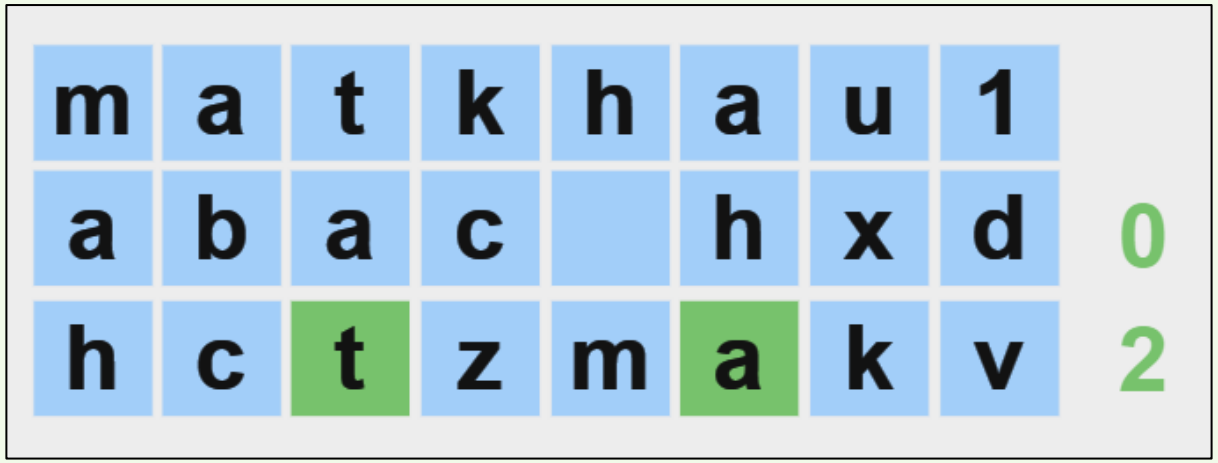
n_Population = 7, n_Generation = 1

4

Bài toán điển hình:

Bài toán tìm mật khẩu

Đánh giá mật khẩu dựa theo tiêu chí: Đúng thứ tự và đúng ký tự

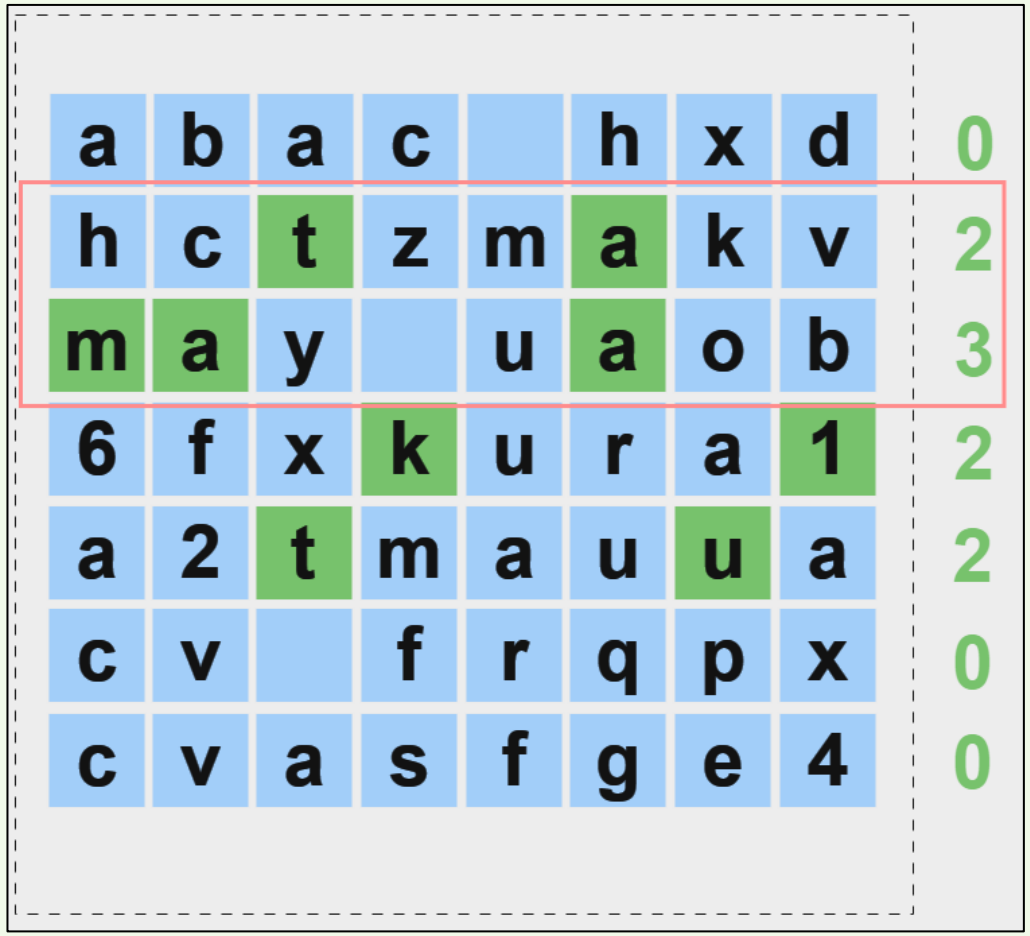


4

Bài toán điển hình:

Bài toán tìm mật khẩu

⇨ Từ đó đánh giá được cả Quần thể dựa trên 2 tiêu chí trên



4

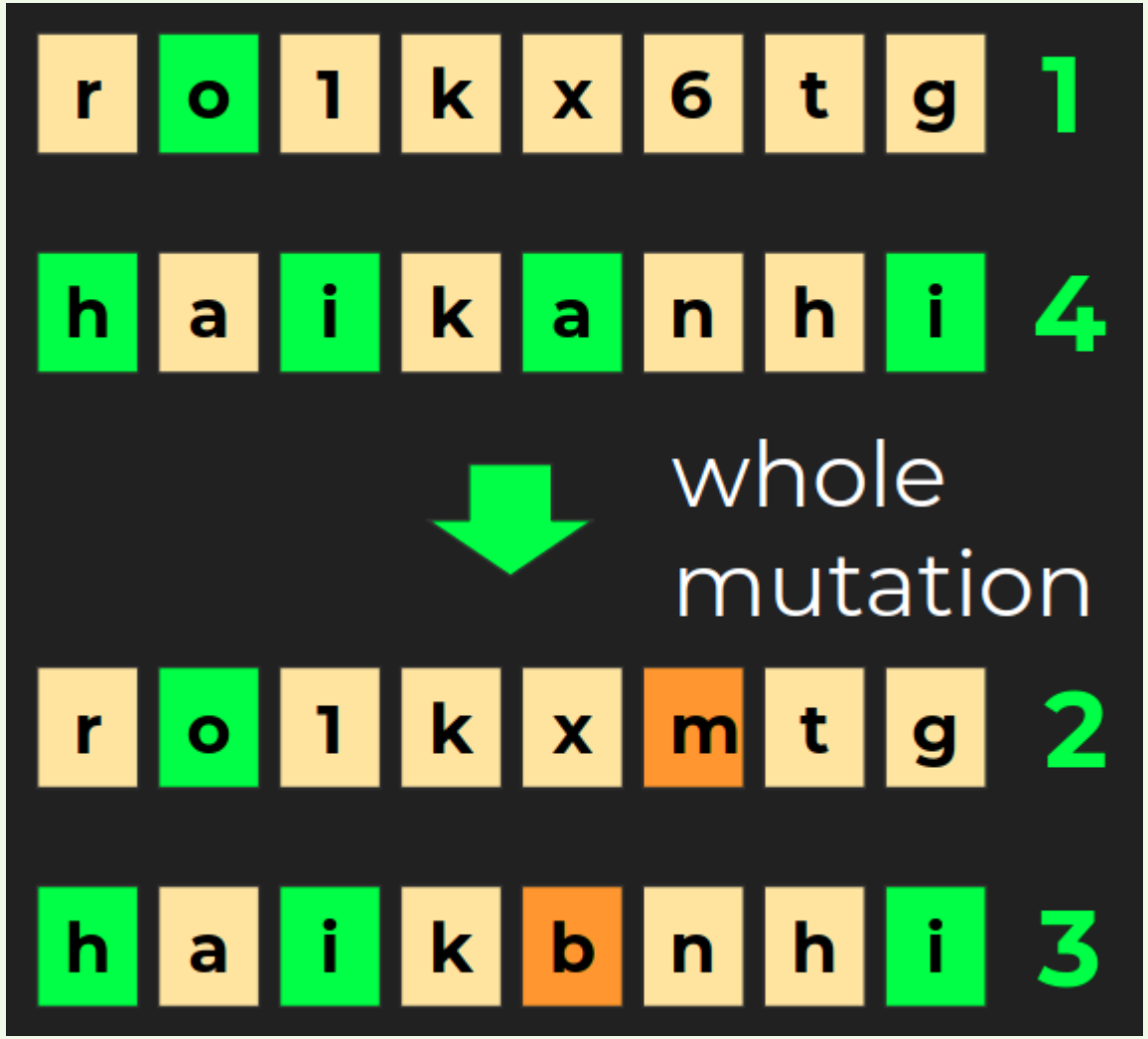
Bài toán điển hình:

Bài toán tìm mật khẩu

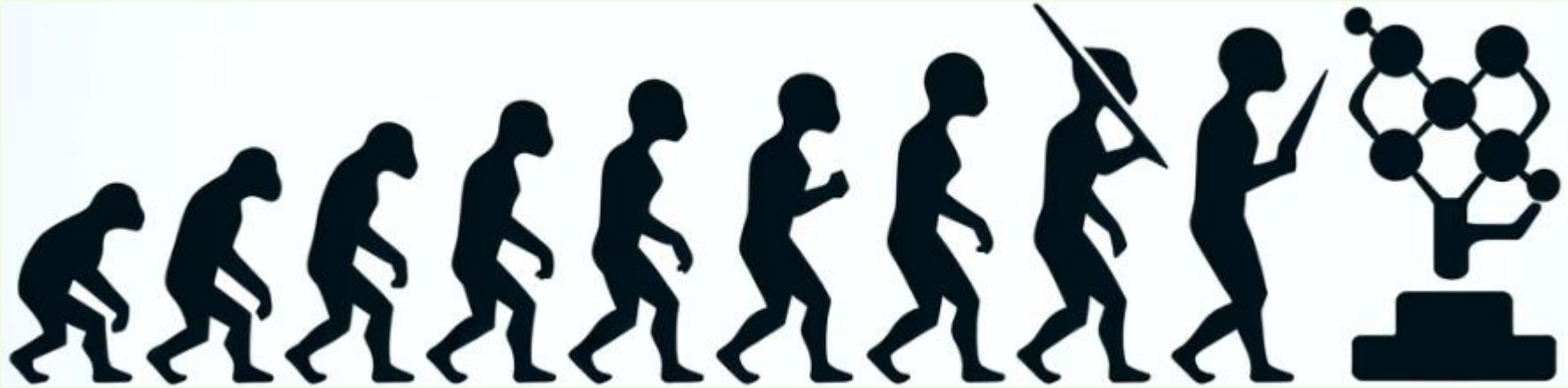
4

Bài toán điển hình:

Bài toán tìm mật khẩu



Thuật toán di truyền (Genetic Algorithms - GA)



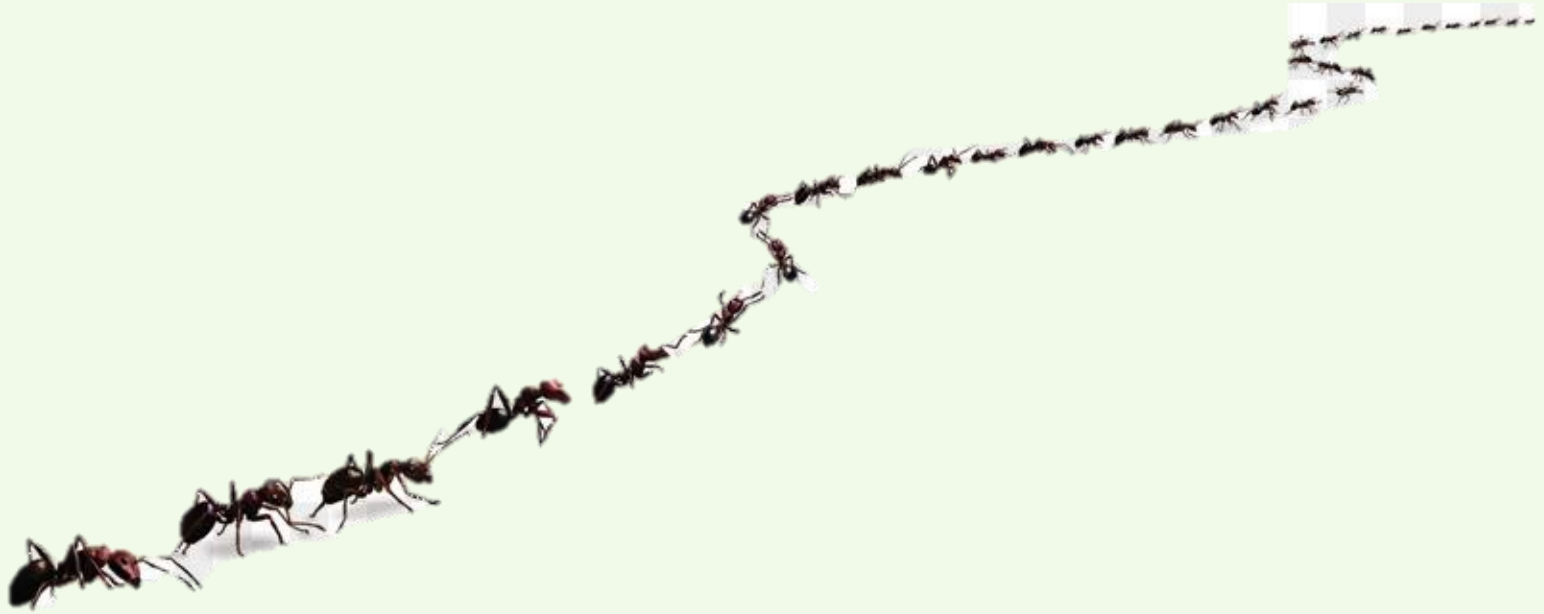
Thuật toán tối ưu hóa đàn kiến
(Ant Colony Otimization - ACO)

1

Các thành phần chính

1. Kiến (Ants)
2. Pheromone
3. Heuristic Information
4. Hàm Mục tiêu (Objective Function):

Chúng bắt đầu từ một điểm, thường là điểm khởi đầu của bài toán, và dần dần xây dựng lộ trình bằng cách lựa chọn từng bước tiếp theo dựa trên pheromone và thông tin heuristic

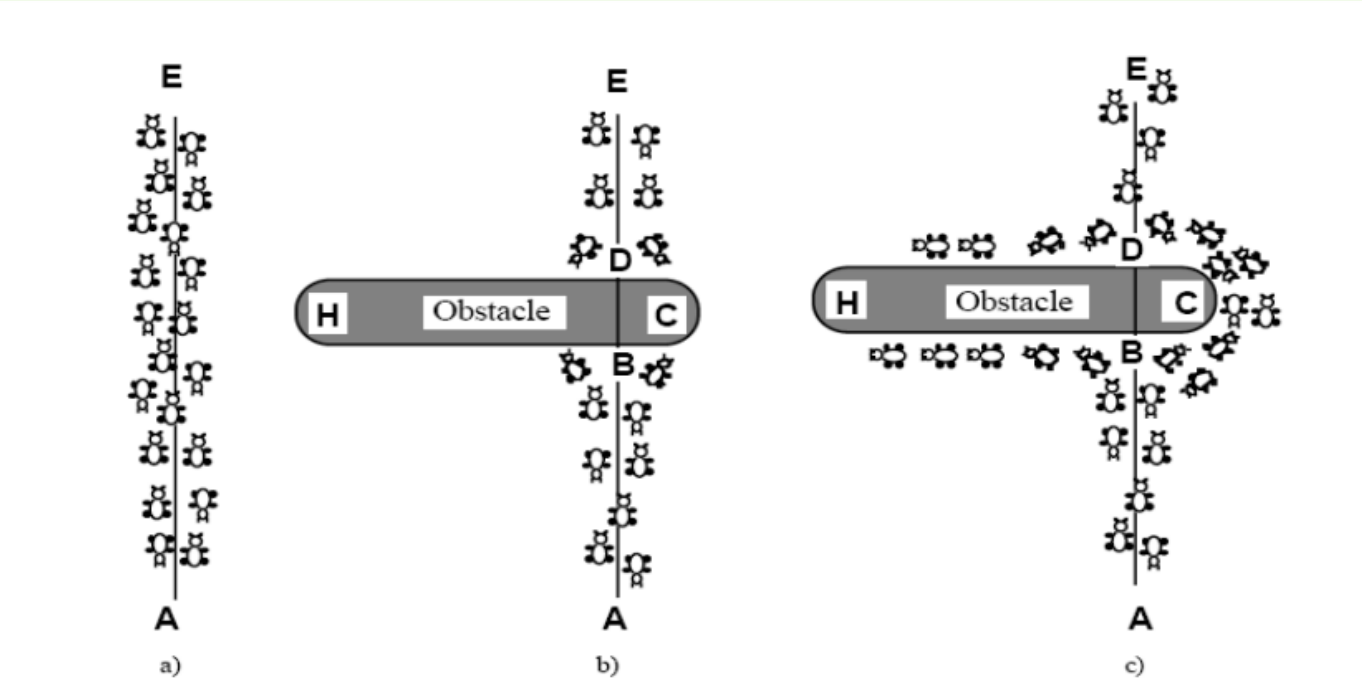


1

Các thành phần chính

1. Kiến (Ants)
2. Pheromone
3. Heuristic Information
4. Hàm Mục tiêu (Objective Function):

Là một biến số ảo mà kiến "để lại" trên lộ trình mà chúng đã đi qua. Lượng pheromone trên một lộ trình phản ánh mức độ thành công của lộ trình đó trong quá khứ, càng nhiều kiến đi qua và càng thành công thì lượng pheromone càng cao

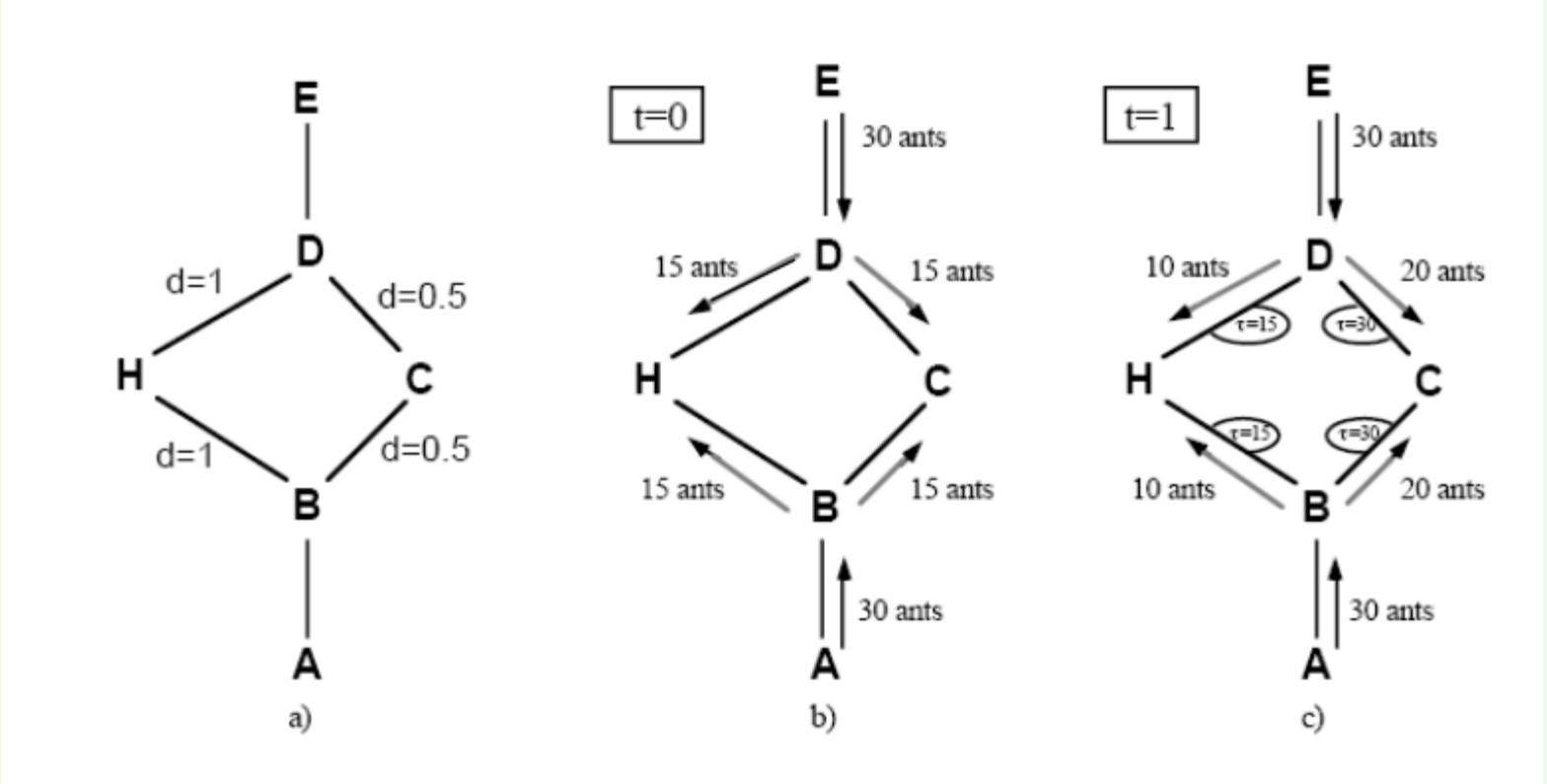


1

Các thành phần chính

1. Kiến (Ants)
2. Pheromone
3. Heuristic Information
4. Hàm Mục tiêu (Objective Function):

Thông tin heuristic là dữ liệu dựa trên bản chất của bài toán, chẳng hạn như khoảng cách giữa các điểm trong bài toán người bán hàng, hoặc chi phí liên quan đến một bước chuyển động trong bài toán tối ưu hóa.

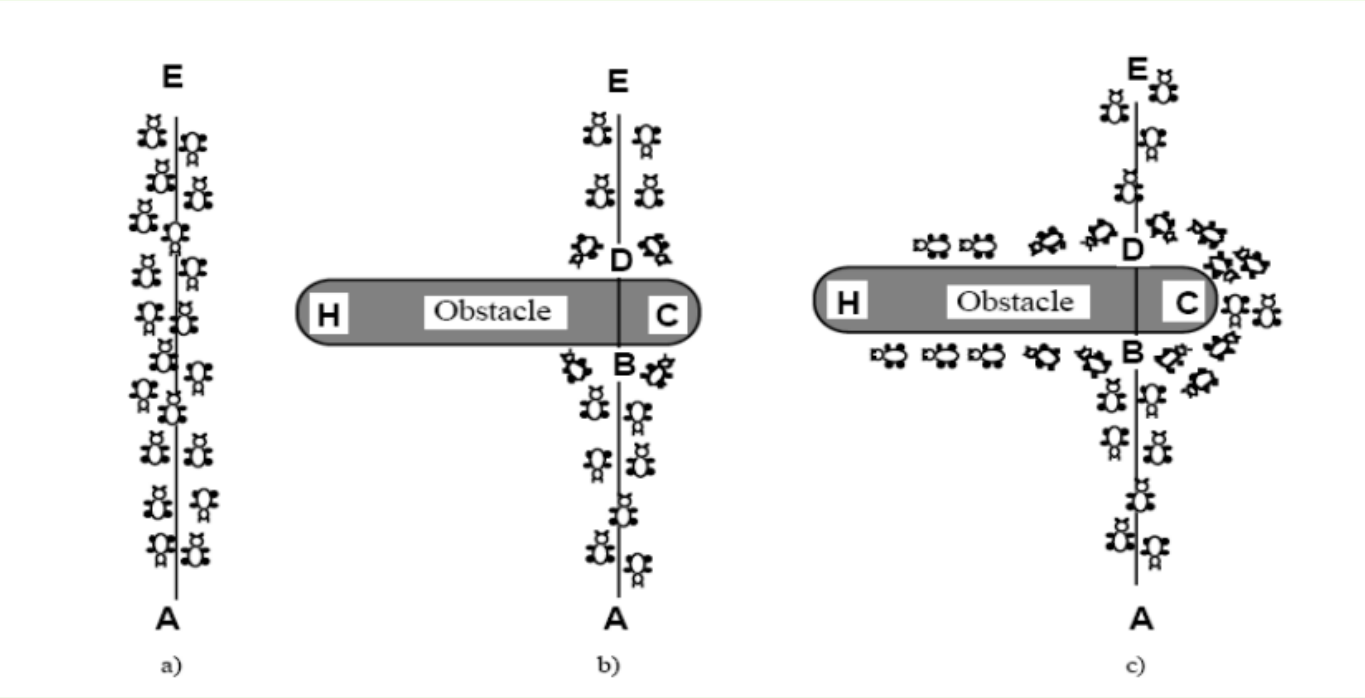


1

Các thành phần chính

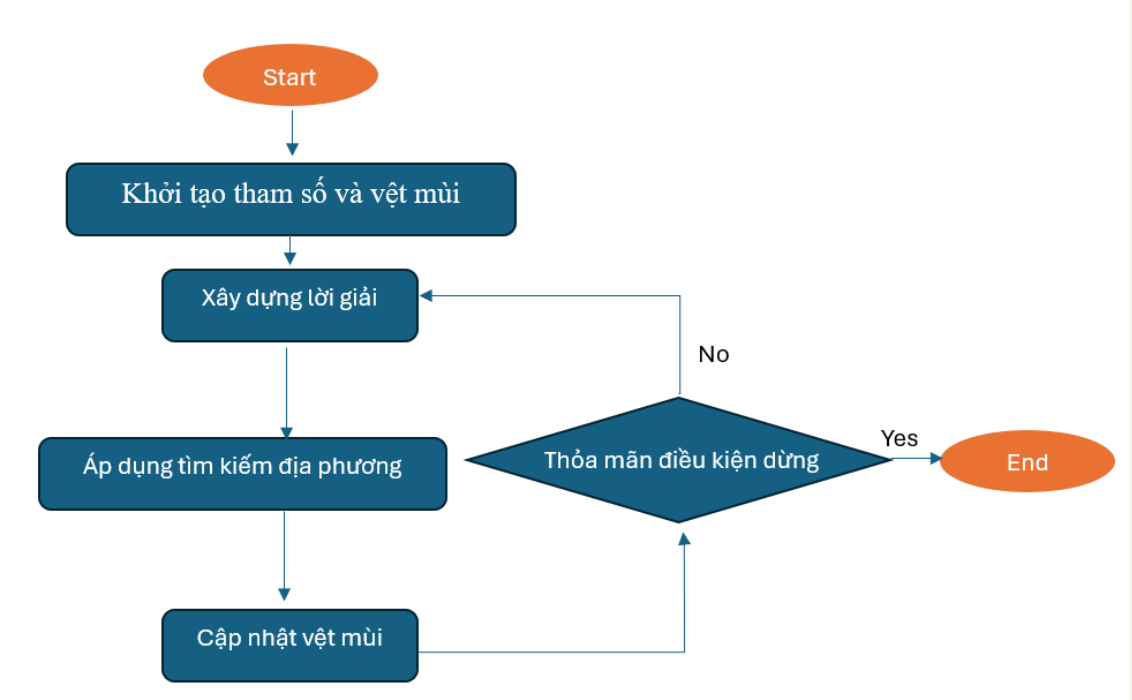
1. Kiến (Ants)
2. Pheromone
3. Heuristic Information
4. Hàm Mục tiêu (Objective Function):

Là tiêu chuẩn để đánh giá chất lượng của một lộ trình hoặc giải pháp. Trong ACO, hàm mục tiêu thường được dùng để tính toán "mức độ thích hợp" của một lộ trình dựa trên tổng chi phí, khoảng cách, thời gian, hoặc các yếu tố tương tự.



2

Mã giả



Mã giả:

Start

Khởi tạo tham số và vệt mùi khởi tạo:

REPEAT:

Xây dựng lời giải

Áp dụng tìm kiếm địa phương(có thể có hoặc không)

Cập nhật các vệt mùi

Stop

3

Độ phức tạp

1. Độ phức tạp thời gian
2. Độ phức tạp không gian

- Số lượng kiến (m): Mỗi con kiến trong quần thể tạo một lộ trình hoàn chỉnh trong bài toán.
- Số thành phần (n): Đây có thể là số lượng thành phố trong TSP. Mỗi kiến cần xem xét mỗi thành phần để xây dựng lộ trình.
- Số lần lặp (t): Đại diện cho số lần thuật toán thực hiện quá trình tìm kiếm giải pháp, bao gồm cả việc cập nhật pheromone.
- Chi phí tính toán cho mỗi lượt di chuyển của kiến (c): Đây là chi phí để mỗi con kiến tạo ra một lộ trình hoàn chỉnh.

Độ phức tạp thời gian tổng thể của ACO có thể là:

$$O(t \times m \times c)$$

ACO cũng yêu cầu một lượng lớn bộ nhớ để lưu trữ thông tin pheromone trên mọi cạnh của đồ thị, điều này có độ phức tạp không gian là: O(n^2)
Đối với bài toán như TSP, nơi pheromone cần được lưu trữ cho mỗi cặp thành phố.

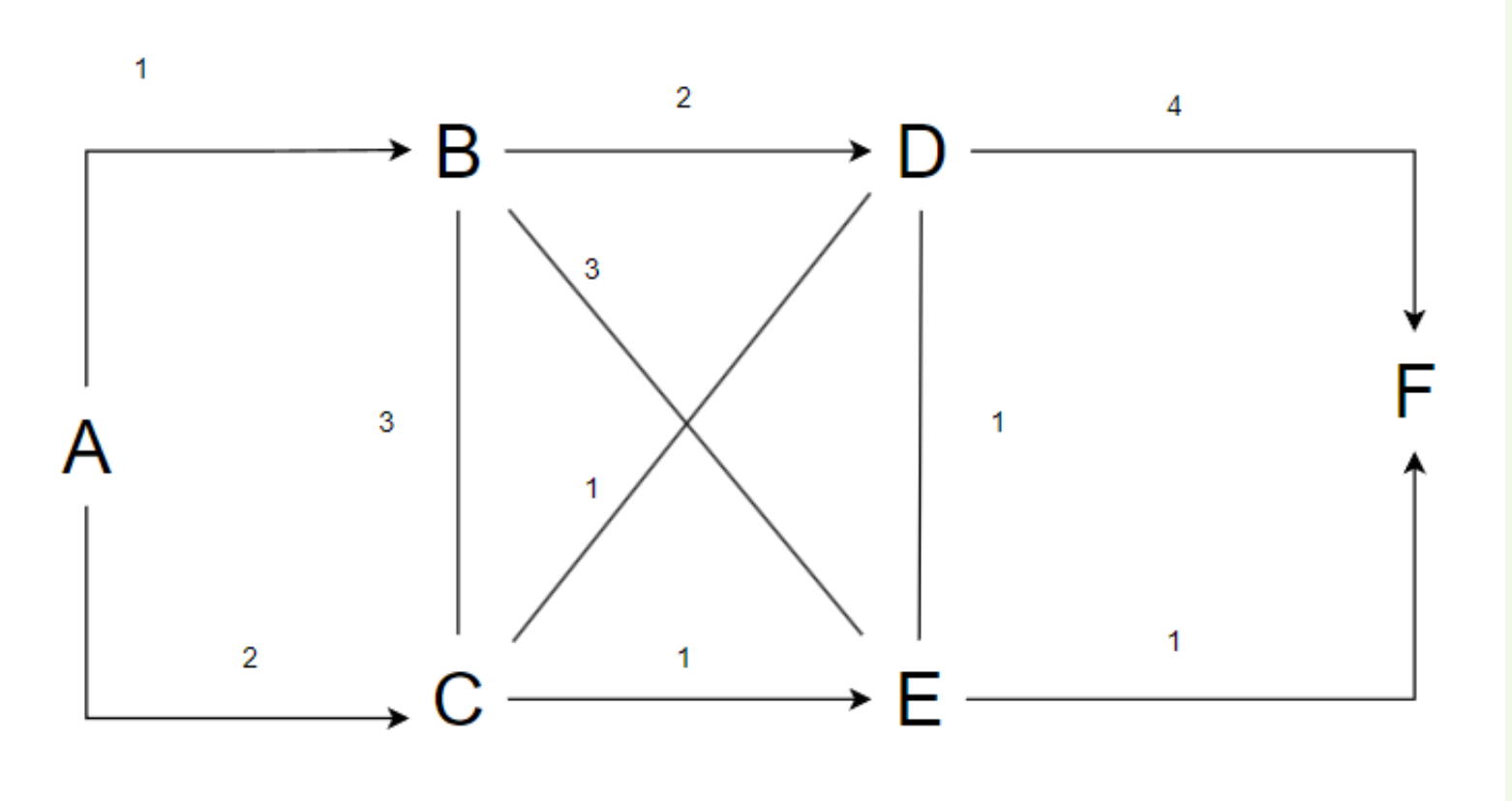
4

Bài toán điển hình:

Bài toán tìm đường đi ngắn nhất

Bài toán tìm đường đi ngắn nhất giữa điểm bắt đầu tới điểm đích là 1 bài toán điển hình của ACO nhằm tối ưu đường đi tới đích một cách ngắn nhất nhanh nhất.

Ví dụ: Cho 6 đỉnh A, B, C, D, E, F với Khoảng cách các đình như hình, khoảng cách tổ kiến bắt đầu từ A và thức ăn tại F. Tìm đường đi ngắn nhất



4

Bài toán điển hình:

Bài toán tìm đường đi ngắn nhất

	A	B	C	D	E	F
A	0	1	2	0	0	0
B	1	0	3	2	3	0
C	2	3	0	1	1	0
D	0	2	1	0	1	4
E	0	3	1	1	0	1
F	0	0	0	4	1	0

4

Bài toán điển hình:

Bài toán tìm đường đi ngắn nhất

$$P_{i,j}^{k}(t) = \frac{\left[\tau_{i,j}(t)\right]^{\alpha} . \left[\eta_{i,j}\right]^{\beta}}{\sum_{l \in N_i^k} \left[\tau_{i,l}(t)\right]^{\alpha} . \left[\eta_{i,l}\right]^{\beta}}$$

- $\tau_{A,B}, \tau_{A,C}$: Mức pheromone ban đầu trên các cạnh A-B và A-C. Giả sử ban đầu là 0.1 cho cả hai cạnh
- $\eta_{A,B}, \eta_{A,C}$: Độ thuận lợi của cạnh, tính bằng nghịch đảo của khoảng cách từ A đến B và từ A đến C. với khảng cách AB = 1 và AC = 2, ta có $\eta_{A,B}$ = 1/1 = 1 và $\eta_{A,C}$ = 1/2 = 0.5
- α, β: Các tham số điều chỉnh ảnh hưởng của pheromone (α) và thông tin heuristic (β). Giả sử $\alpha = 1,\ \beta = 2$.

- $\tau_{A,B}, \tau_{A,C} = 0.1$
- $\eta_{A,B} = 1, \eta_{A,C} = 0.5$
- $\alpha = 1, \beta = 2$

4

Bài toán điển hình:

Bài toán tìm đường đi ngắn nhất

$$P_{i,j}^{k}(t) = \frac{\left[\tau_{i,j}(t)\right]^{\alpha}.\left[\eta_{i,j}\right]^{\beta}}{\sum_{l \in N_i^k}\left[\tau_{i,l}(t)\right]^{\alpha}.\left[\eta_{i,l}\right]^{\beta}}$$

	A	B	C	D	E	F
A	0	1	2	0	0	0
B	1	0	3	2	3	0
C	2	3	0	1	1	0
D	0	2	1	0	1	4
E	0	3	1	1	0	1
F	0	0	0	4	1	0

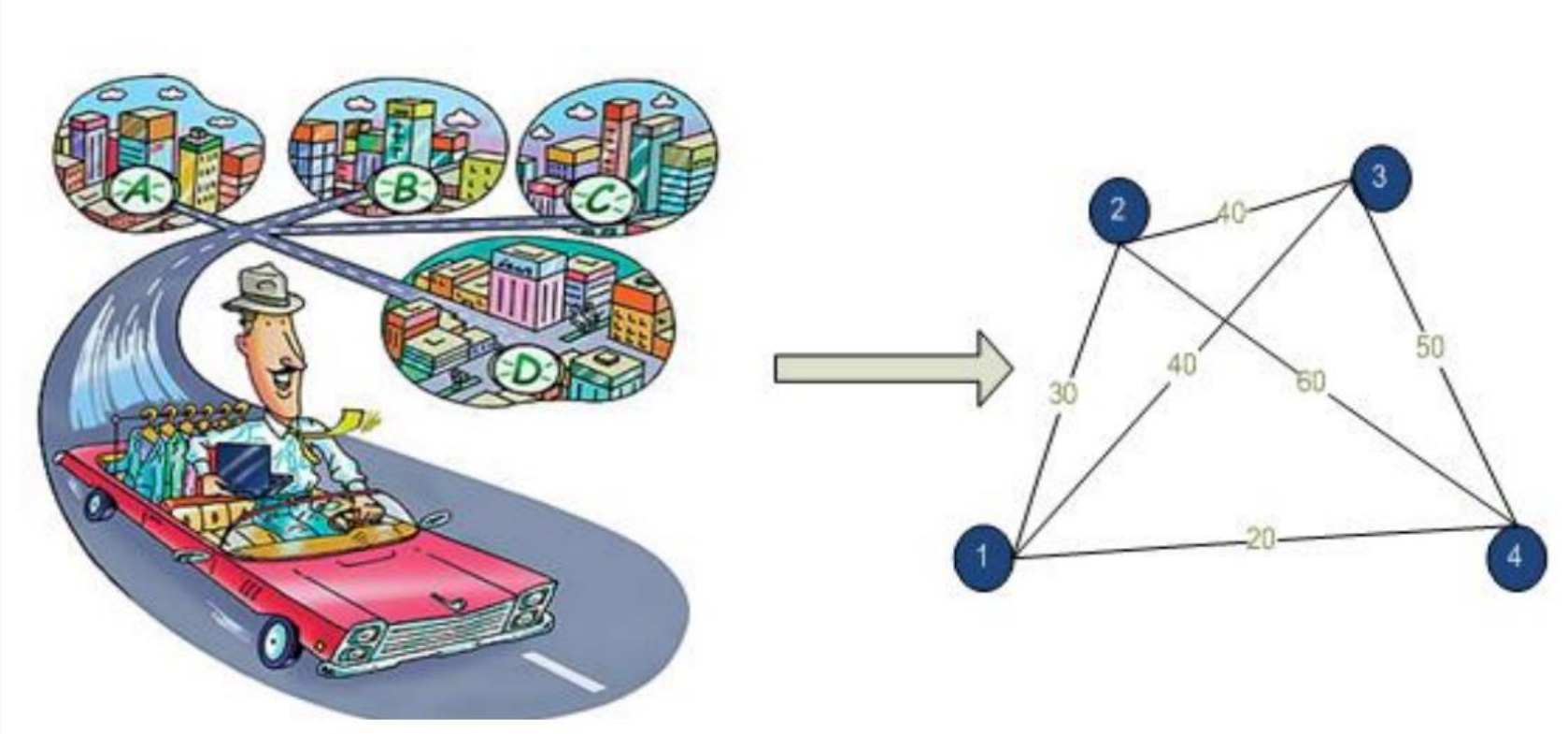
- $\tau_{A,B}, \tau_{A,C} = 0.1$
- $\eta_{A,B} = 1, \eta_{A,C} = 0.5$
- $\alpha = 1, \beta = 2$

$$P_{A,B} = \frac{(0.1)^1.(1)^2}{(0.1)^1.(1)^2 + (0.1)^1.(0.5)^2} = \frac{0.1}{0.35} \approx 0.2857$$

$$P_{A,C} = \frac{(0.1)^1.(0.5)^2}{(0.1)^1.(1)^2 + (0.1)^1.(0.5)^2} = \frac{0.05}{0.35} = 0.0175$$

Thuật toán tối ưu hóa đàn kiến
(Ant Colony Otimization - ACO)

ỨNG DỤNG VÀO BÀI TOÁN NGƯỜI BÁN HÀNG (TRAVELLING SALESMAN PROBLEM - TSP)



LÝ DO

Vì bài toán này so sánh được 1 cách tổng quan cho 2 thuật toán GA và ACO. Nó nhằm mục đích thử thách trong tối ưu hóa kinh điển, kiểm tra khả năng tìm kiếm và khai thác của thuật toán, ứng dụng thực tế, tối ưu hóa và cải tiến thuật toán.

Cùng với đó là nó còn cho biết được thuật toán GA cũng có thể áp dụng được với bài toán tìm đường đi và nó tổng quan hơn thuật toán ACO.

Ý TƯỞNG

Thuật toán Di truyền (Genetic Algorithms - GA)

Ý tưởng cơ bản: Mô phỏng quá trình tiến hóa sinh học, bao gồm các phép toán như chọn lọc tự nhiên, lai ghép và đột biến để tìm ra giải pháp tối ưu.

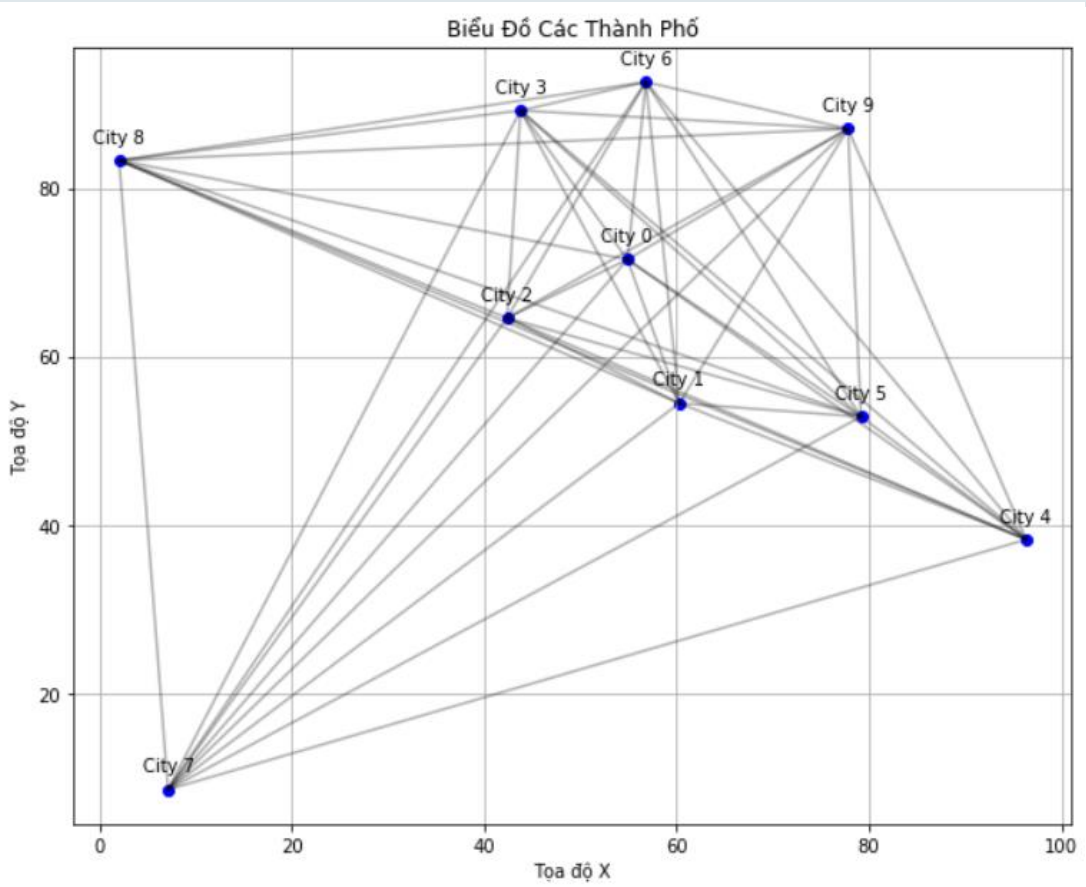
Thuật toán tối ưu hóa đàn kiến (Ant Colony Optimization - ACO)

Ý tưởng cơ bản: Mô phỏng hành vi tìm đường và xây dựng đường đi của kiến trong tự nhiên, dựa trên việc tích lũy pheromone trên đường đi để hướng đến giải pháp tối ưu

THỰC NGHIỆM

Với 10 thành phố có tên theo thứ tự từ 0 đến 9
Tìm quãng đường đi ngắn nhất từ 1 thành phố bất kỳ đi hết các thành phố rồi kết thúc tại thành phố đầu tiên

```
[  0.  18.  14.  21.  53.  31.  21.  79.  54.  28.]
[ 18.   0.  21.  38.  40.  19.  38.  70.  65.  37.]
[ 14.  21.   0.  25.  60.  39.  31.  66.  44.  42.]
[ 21.  38.  25.   0.  73.  51.  13.  88.  42.  34.]
[ 53.  40.  60.  73.   0.  23.  67.  94. 104.  52.]
[ 31.  19.  39.  51.  23.   0.  46.  85.  83.  34.]
[ 21.  38.  31.  13.  67.  46.   0.  97.  56.  22.]
[ 79.  70.  66.  88.  94.  85.  97.   0.  75. 105.]
[ 54.  65.  44.  42. 104.  83.  56.  75.   0.  76.]
[ 28.  37.  42.  34.  52.  34.  22. 105.  76.   0.]
```



VỚI 10 THÀNH PHỐ CỦA GA

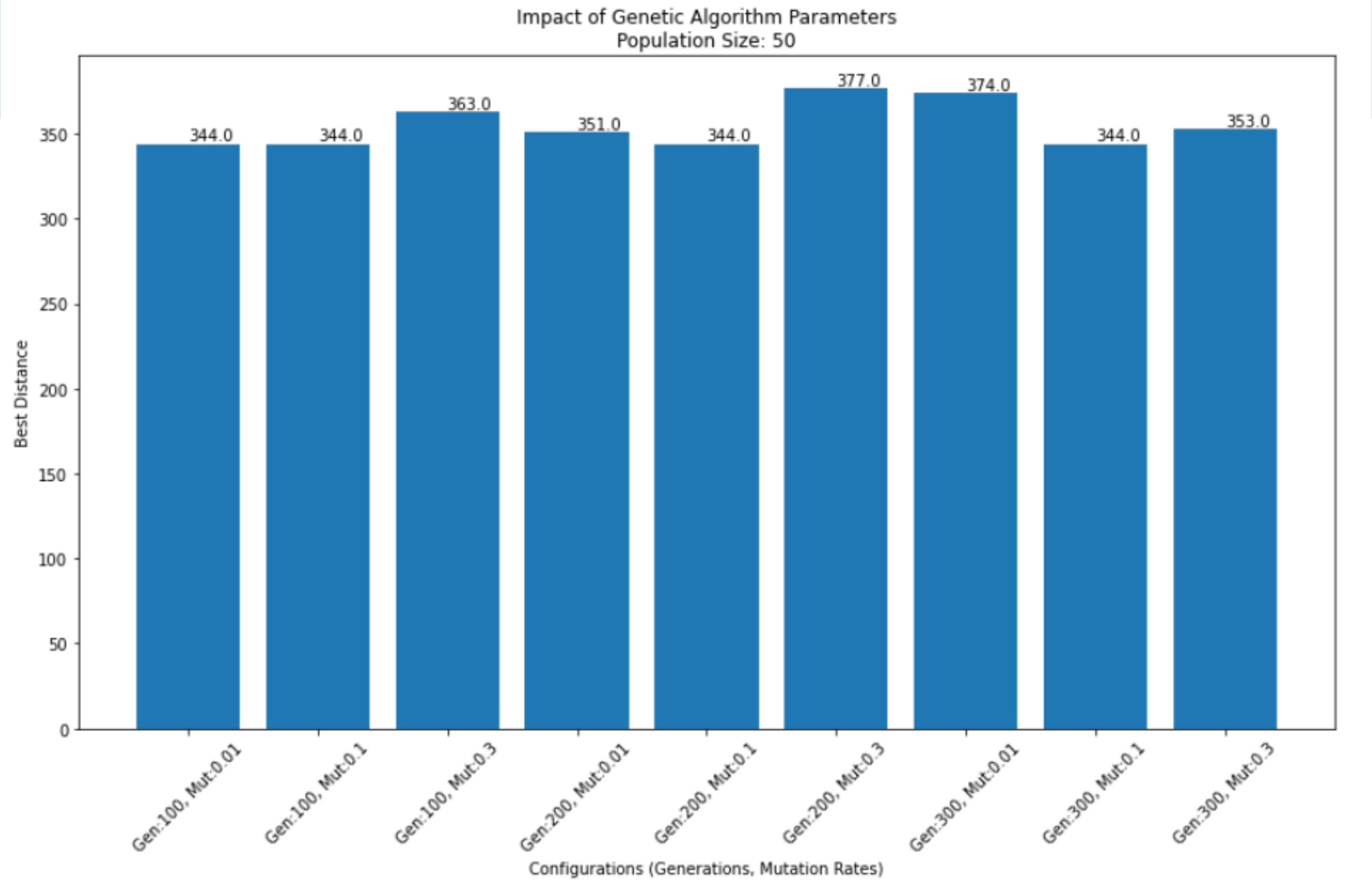
Khởi tạo ban đầu

populations = [50, 100, 200]

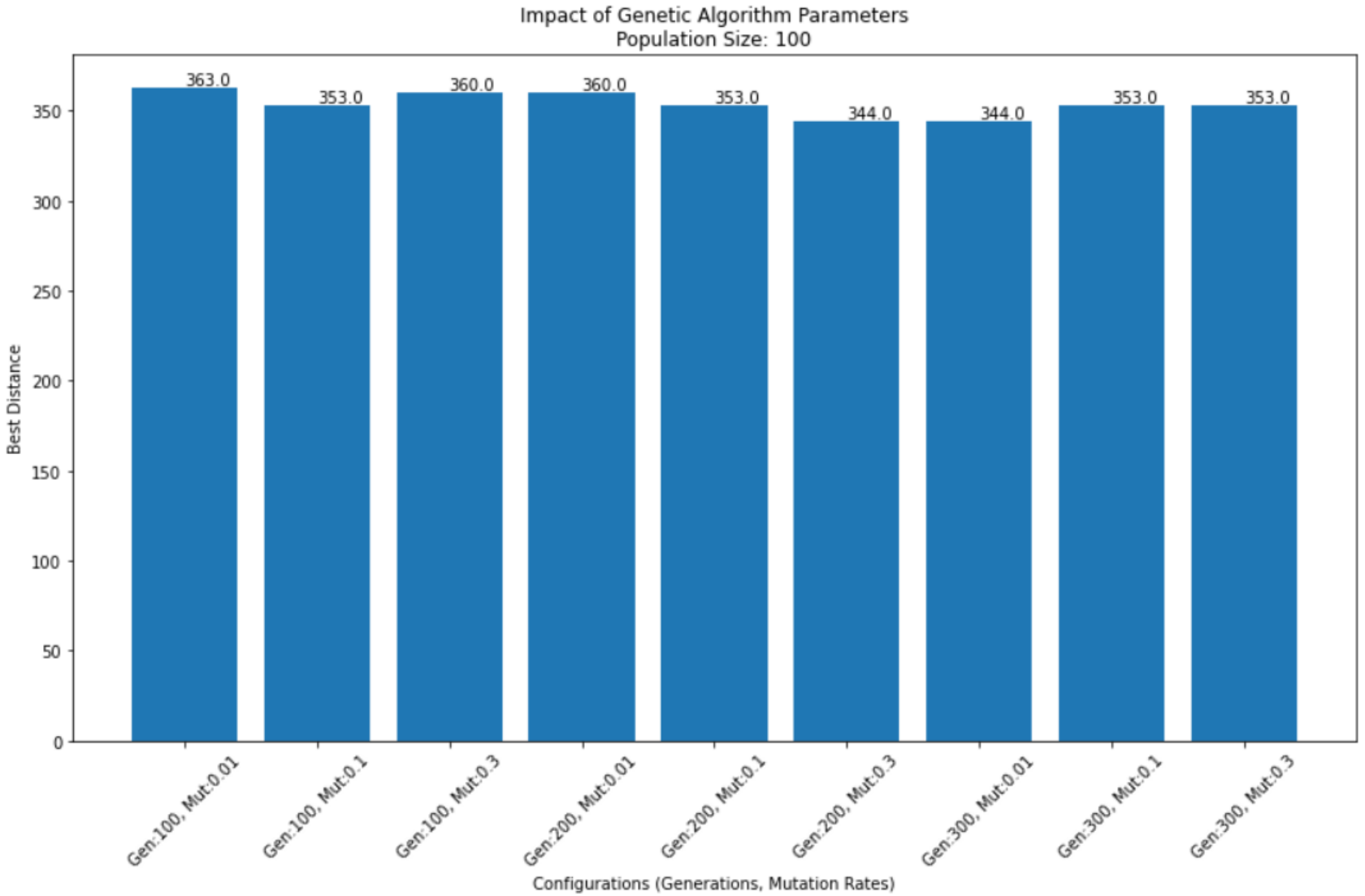
generations = [100, 200, 300]

mutation_rates = [0.01, 0.1, 0.3]

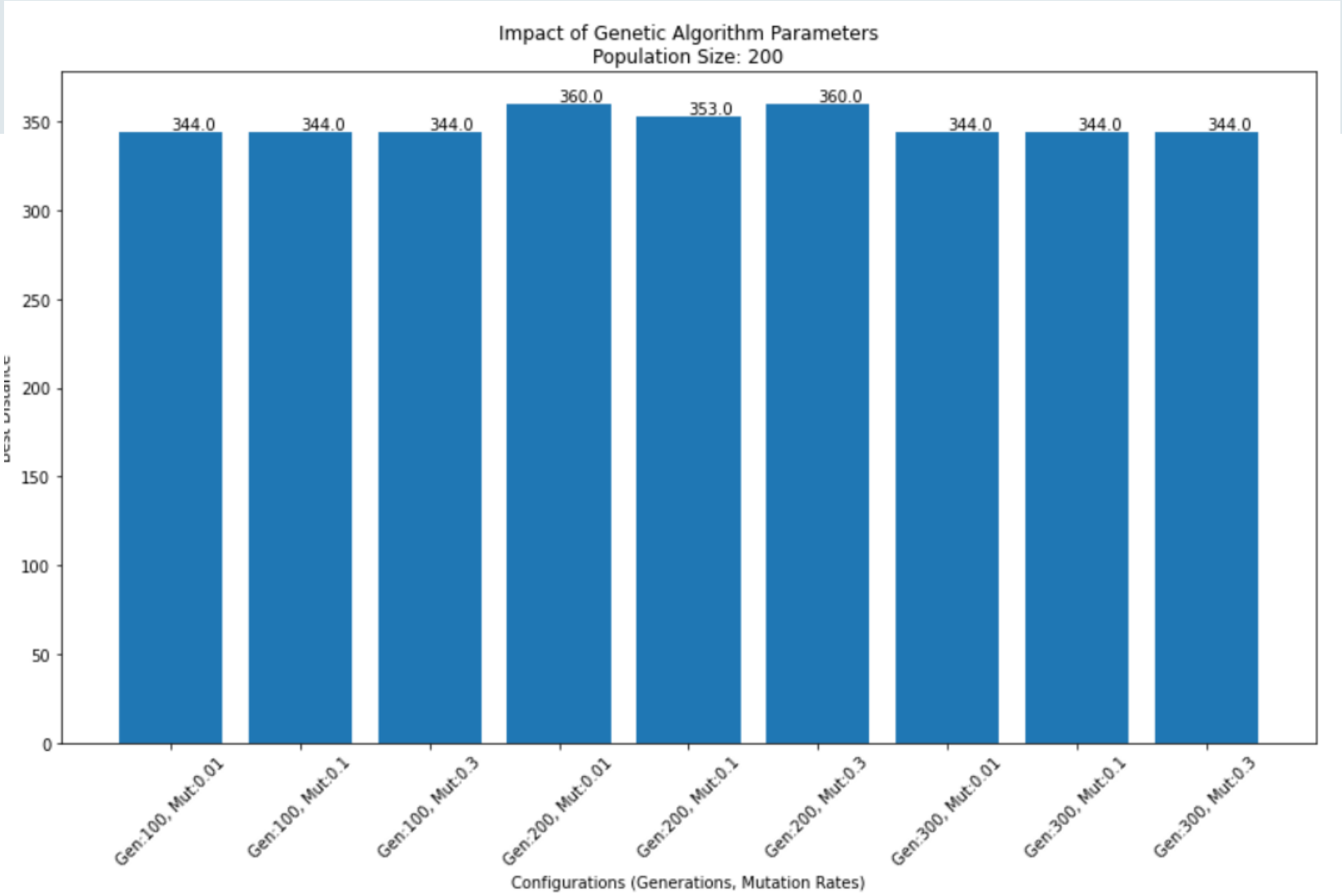
VỚI 10 THÀNH PHỐ CỦA GA



VỚI 10 THÀNH PHỐ CỦA GA



VỚI 10 THÀNH PHỐ CỦA GA



VỚI 10 THÀNH PHỐ CỦA ACO

Khởi tạo

n_ants_options = 20

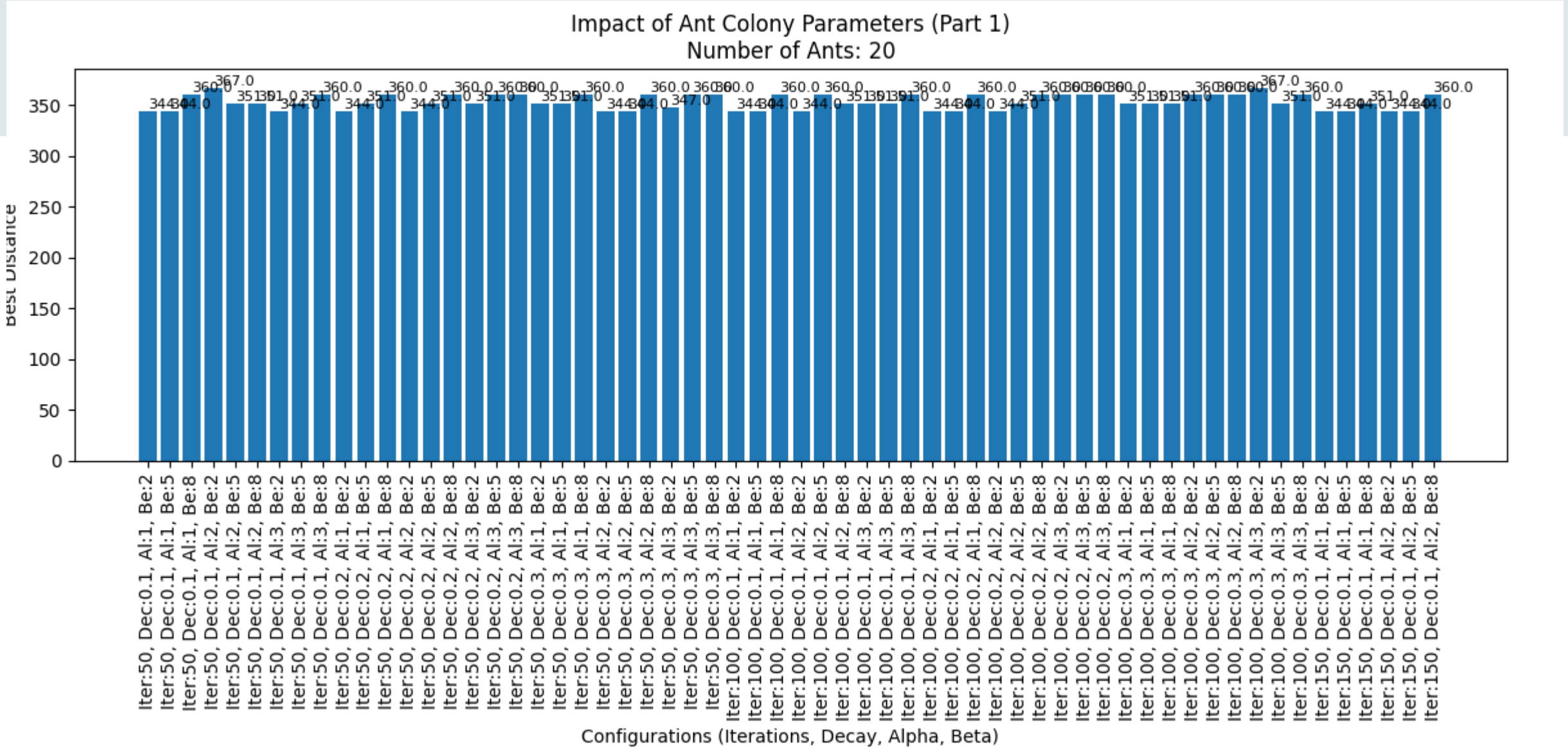
n_iterations_options = [50, 100, 150]

decay_options = [0.1, 0.2, 0.3]

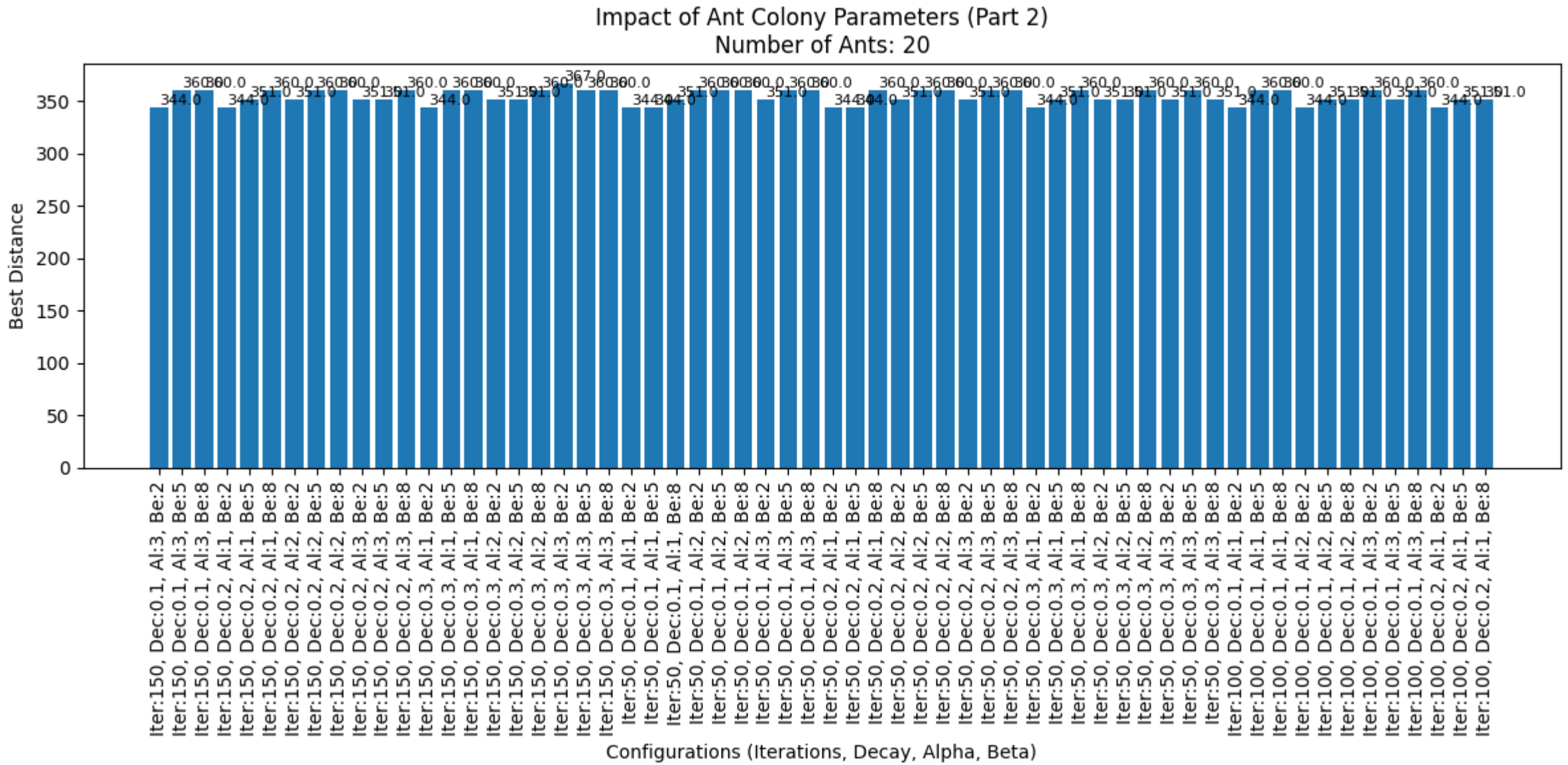
alpha_options = [1, 2, 3]

beta_options = [2, 5, 8]

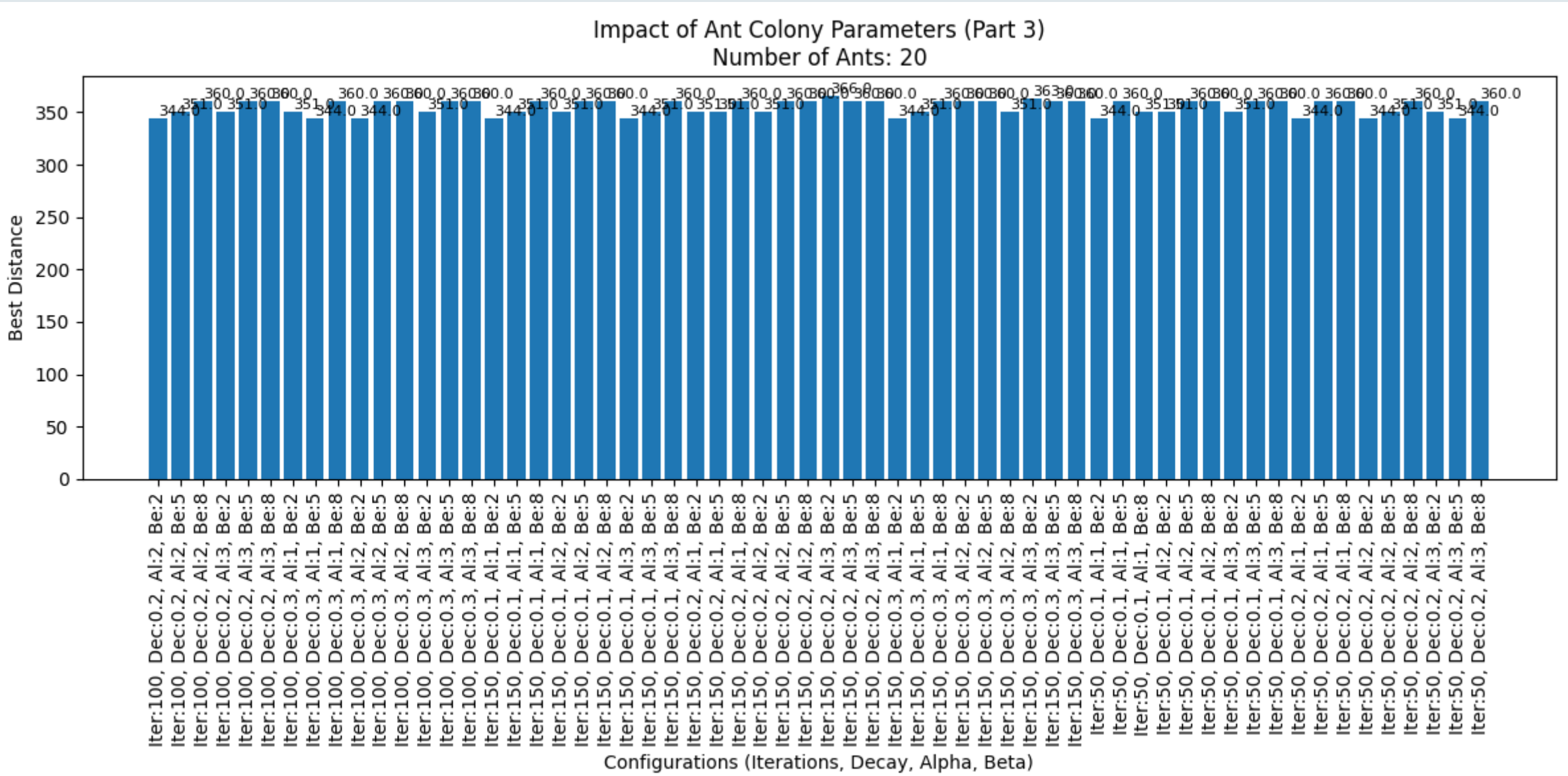
VỚI 10 THÀNH PHỐ CỦA ACO



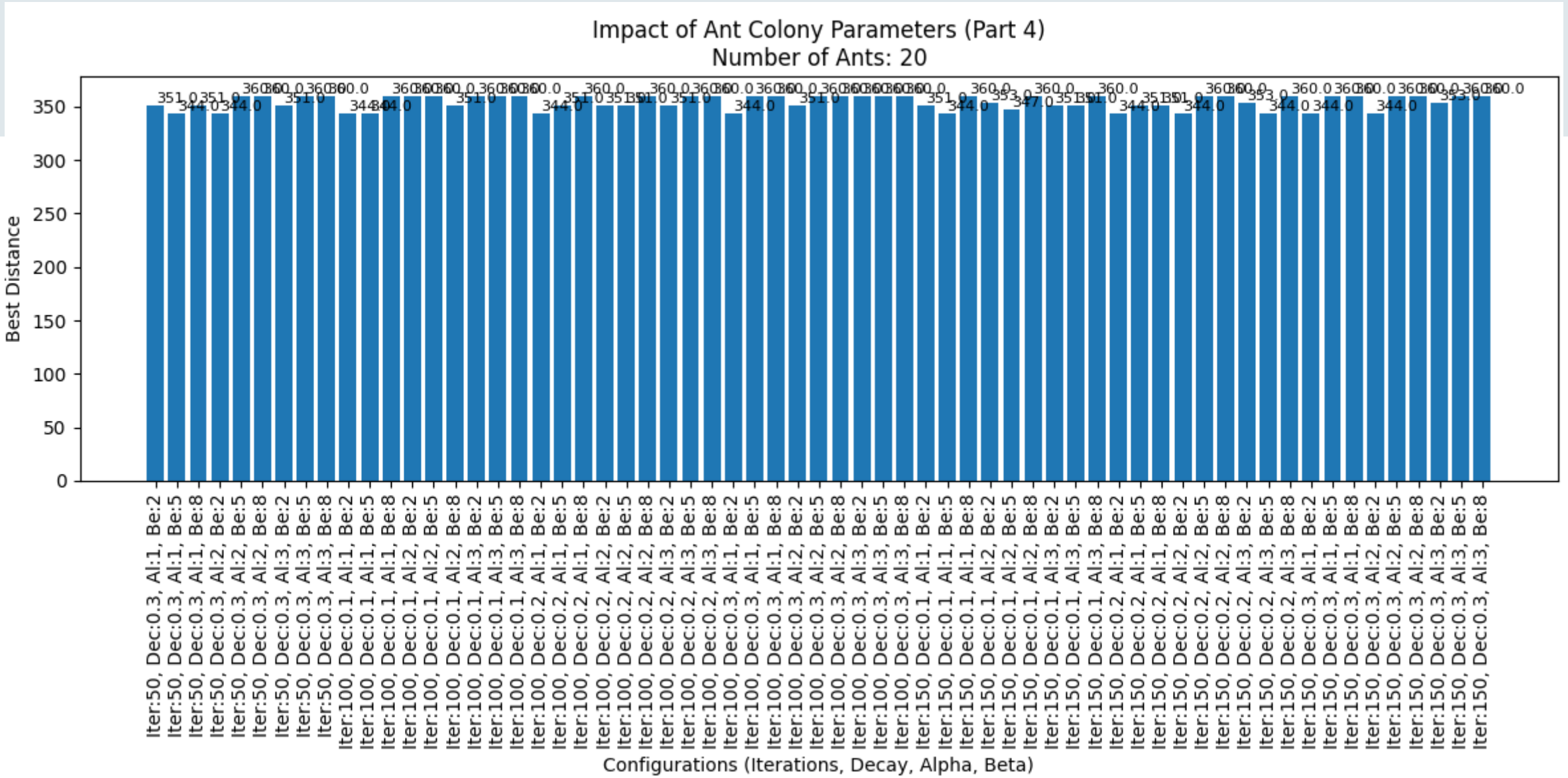
VỚI 10 THÀNH PHỐ CỦA ACO



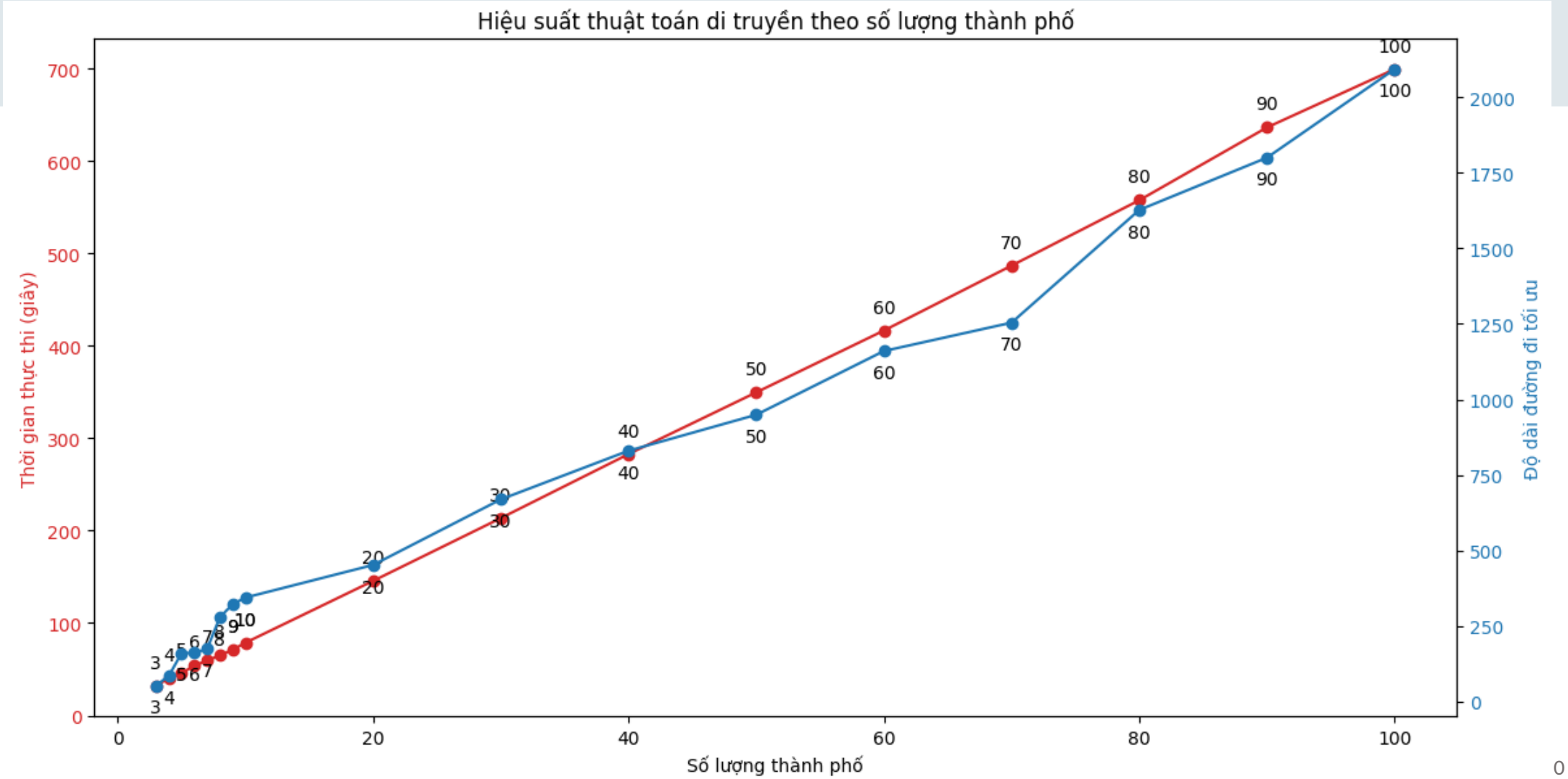
VỚI 10 THÀNH PHỐ CỦA ACO



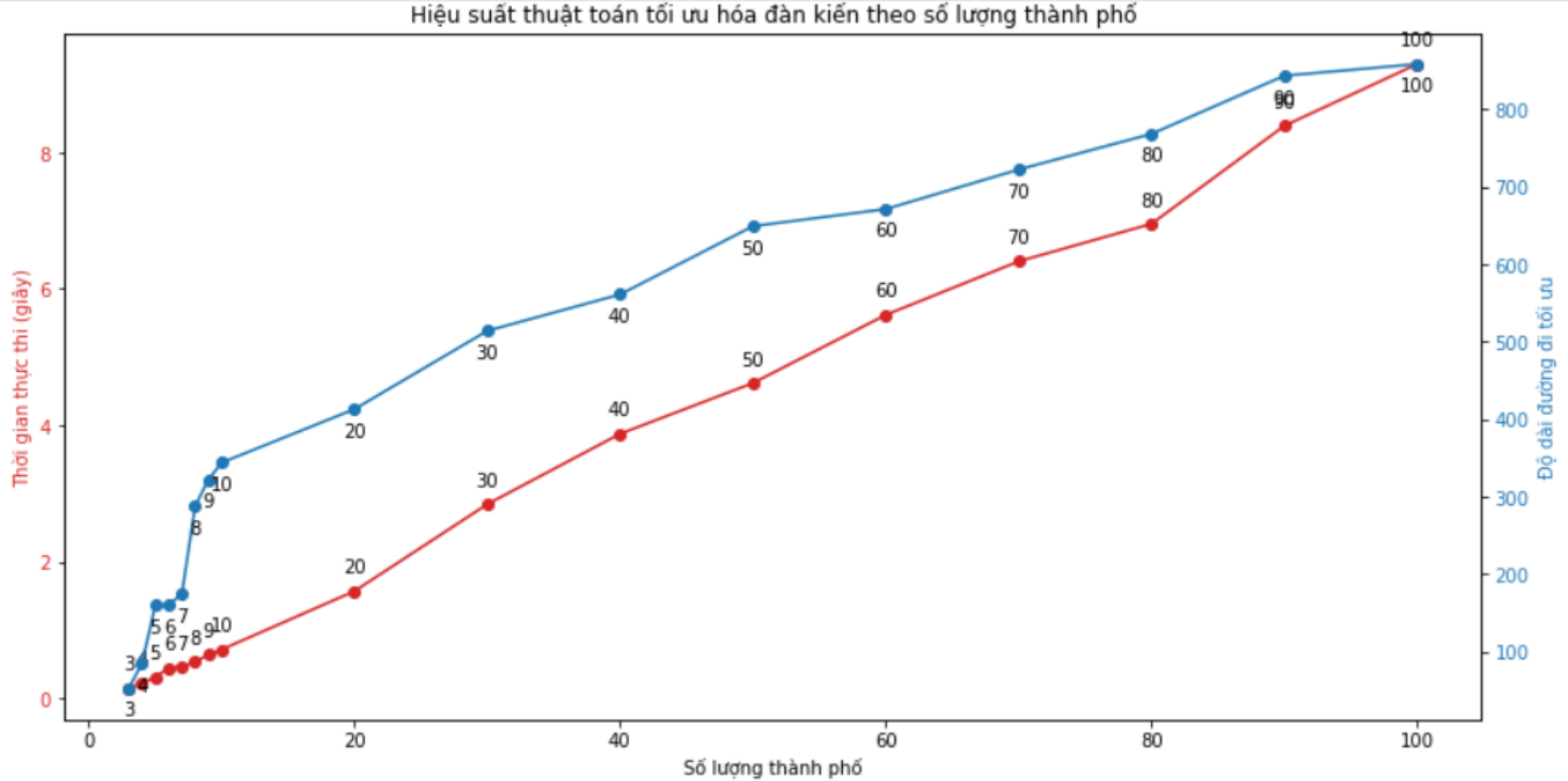
VỚI 10 THÀNH PHỐ CỦA ACO



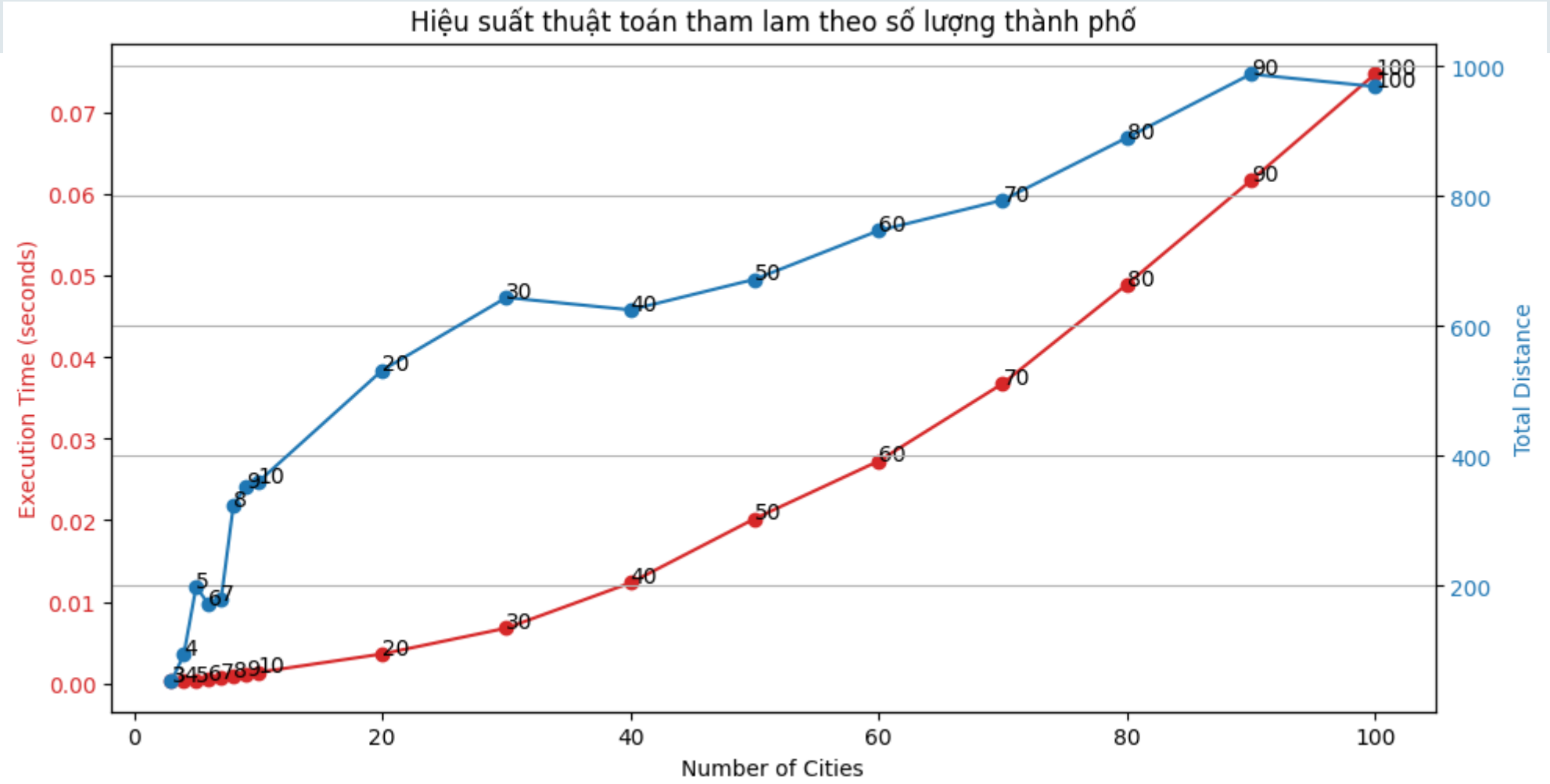
VỚI 0-100 THÀNH PHỐ CỦA GA



VỚI 0-100 THÀNH PHỐ CỦA ACO



VỚI 0-100 THÀNH PHỐ CỦA THUẬT TOÁN THAM LAM



ĐÁNH GIÁ

Cả ACO và GA đều là những công cụ mạnh mẽ và có những ưu điểm riêng biệt phù hợp với các loại bài toán khác nhau. ACO thường được ưu tiên khi cần một giải pháp tối ưu chính xác cho các bài toán đường đi phức tạp, trong khi GA là lựa chọn tốt cho các bài toán đòi hỏi khả năng khám phá không gian lớn và tính đa dạng cao. Việc lựa chọn giữa hai thuật toán phụ thuộc vào đặc điểm cụ thể của từng bài toán và mục tiêu tối ưu hóa.

CẢM ƠN THẦY CÔ ĐÃ CHÚ Ý LẮNG NGHE Ạ

Xin trân thành cảm ơn