

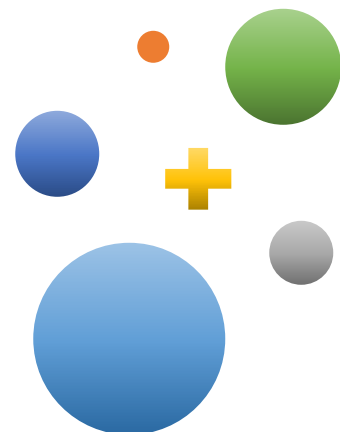
ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN



Báo cáo đồ án

Tìm hiểu GA & Bee cho Vehicle Routing Problem

GVHD: Võ Hoài Việt



Một số thuật ngữ sử dụng và ý nghĩa:

Thuật ngữ	Ý nghĩa
Genetic Algorithm (GA)	Thuật giải di truyền
Artificial Bee Colony Algorithm (ABC)	Thuật giải đàn ong nhân tạo
Vehicle Routing Problem (VRP)	Bài toán định tuyến xe chở hàng
Capacitated Vehicle Routing Problem (CVRP)	Bài toán định tuyến có tải trọng

A Thành viên nhóm:

Tên nhóm: Fusion

STT	MSSV	Họ tên	Email
1	1612174	Phùng Tiến Hào	tienhaophung@gmail.com
2	1612269	Võ Quốc Huy	voquochuy304@gmail.com

B Mức độ hoàn thành:

STT	Nội dung	Mức độ hoàn thành (%)
1	Cài đặt thuật giải ABC cho VRP	100
2	Kết hợp GA và ABC cho VRP	100
Tổng cộng:		100

C

Báo cáo:

I. Đặc tả:

Việc phân phối hàng hóa đến khách hàng là một vấn đề quan trọng của các công ty vận chuyển cũng như các công ty thương mại điện tử. Biết phân phối một cách tối ưu và tiết kiệm không chỉ giúp giảm giá thành, thời gian vận chuyển mà còn giúp nâng cao được sự tin tưởng của khách hàng. Việc lựa chọn chiến lược vận chuyển tốt nhất trong bất kỳ tổ chức nào thường sẽ được sàng lọc lại từ nhiều chiến lược thông qua các tiêu chí được tổ chức đó đặt ra ban đầu. Vì những điều này nên sự giúp ích của các thuật toán tự động tìm đường đi (Vehicle Routing) như Bee Colony hay Genetic là rất cần thiết.

Ở đây, nhóm thực hiện tìm hiểu và cài đặt cho bài toán CVRP (một biến thể của VRP). Tổng quan cách cài đặt đều dựa vào [1] nhưng đồng thời nhóm cũng có những hiệu chỉnh riêng để giúp thuật toán hội tụ nhanh hơn và cho kết quả tốt hơn.

1) Giới hạn bài toán

Thường khi nhắc đến bài toán vận chuyển thì sẽ có 3 giới hạn (tiêu chí) chính: quãng đường đi (distance), trọng lượng tối đa (weight) mà mỗi phương tiện vận chuyển có thể mang và thời gian vận chuyển (time). Ở đây, chúng tôi sẽ chỉ giới hạn lại việc giải quyết bài toán trên 2 tiêu chí là tổng đường đi của các phương tiện đạt ngắn nhất có thể và tổng khối lượng hàng hóa mà mỗi phương tiện mang đi không vượt quá nhiều so với tải trọng của phương tiện. Tất cả phương tiện vận chuyển được mặc định là giống nhau.

Mỗi giải pháp vận chuyển sẽ được biểu diễn bằng một chuỗi có thứ tự các chỉ số của khách hàng và chỉ số của trung tâm (mặc định là 0). Mỗi chỉ số này sẽ gắn liền với thông tin về vị trí của khách hàng (trung tâm) và trọng lượng món hàng tương ứng.

Ví dụ: có 9 khách hàng và 3 phương tiện vận chuyển, một giải pháp sẽ được biểu diễn bằng một solution vector như sau:

0	3	7	5	0	1	4	2	6	0	9	8
---	---	---	---	---	---	---	---	---	---	---	---

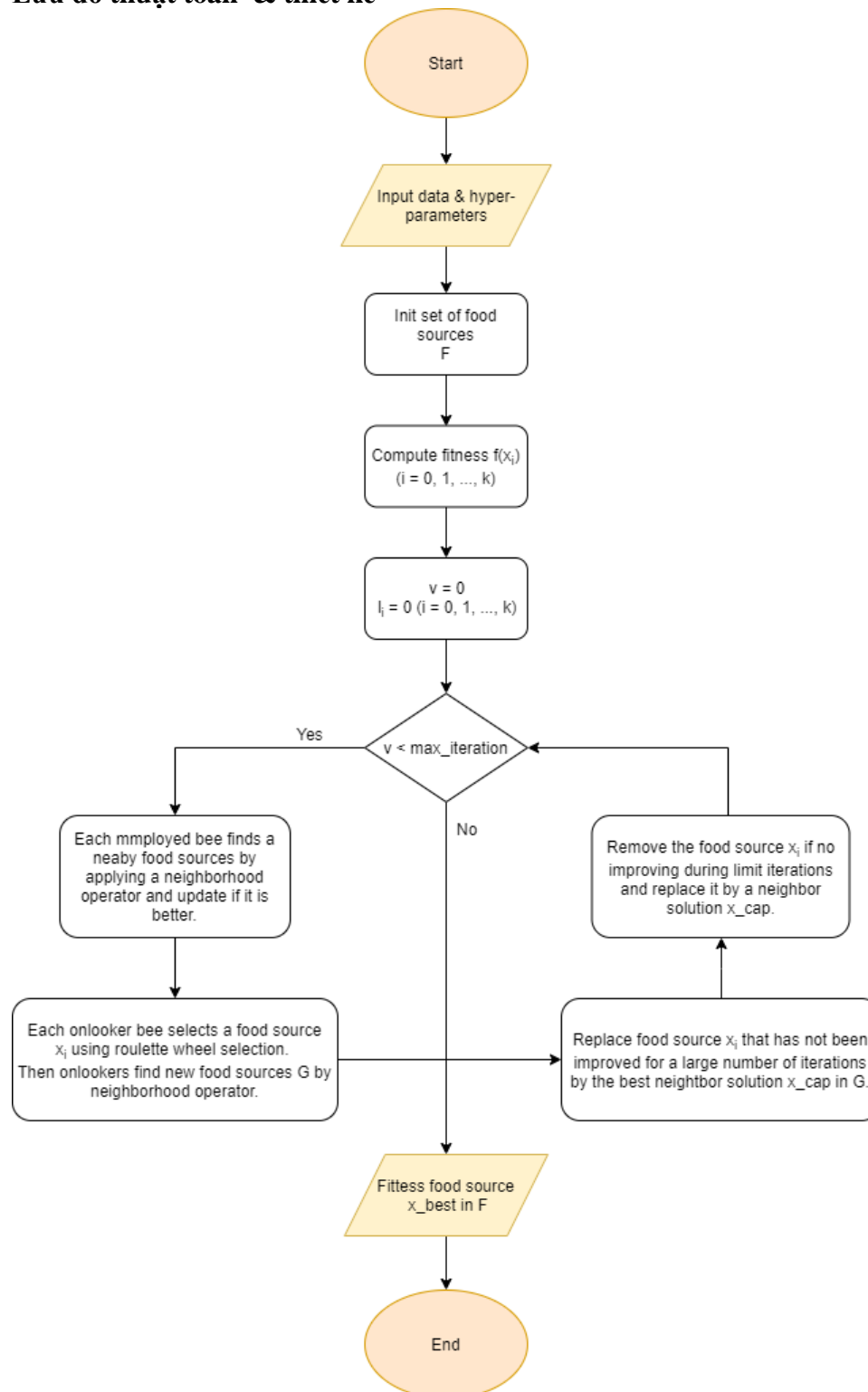
Với các chỉ số 0 tượng trưng cho vị trí bắt đầu của mỗi phương tiện. Mỗi phương tiện sẽ di chuyển từ vị trí 0 này lần lượt đến các vị trí ngay sau đó cho đến khi gặp một vị trí 0 khác hoặc đến cuối chuỗi thì nghĩa là báo hiệu quay về trung tâm (depot).

Cost function sử dụng là $z(x) = c(x) + \alpha \cdot q(x) + \gamma \cdot \text{diffcost}(x)$. Với:

- x : vector biểu diễn một giải pháp đường đi.

- $c(x)$: tổng quãng đường đi của các phương tiện theo giải pháp x .
- $q(x)$: tổng mức trọng lượng bị lệch hơn trọng lượng tối đa của mỗi phương tiện (hiệu của trọng lượng xe phải chở với trọng lượng tối đa của xe).
- α : chỉ số thể hiện mức độ quan trọng của khối lượng hàng hóa mà mỗi phương tiện phải chở.
- $\text{diffcost}(x)$: đây là phần cải tiến của nhóm. Diffcost là tổng các hiệu quãng đường của từng cặp phương tiện đối một tham gia vận chuyển. Mục đích của Diffcost là để giữ cho các quãng đường của các phương tiện bằng nhau nhất có thể.
- γ : để điều chỉnh mức độ ảnh hưởng của diffcost vào total cost $z(x)$, mặc định là 0.1.

2) Lưu đồ thuật toán & thiết kế



Ở đây, các thuật toán neighborhood operator sử dụng là random swap, reverse subsequence và random swap of reverse subsequence như [1] đề xuất. Nhưng nhóm có tinh chỉnh nhỏ để giúp hệ thống hiệu quả hơn.

Random swap: nhóm thực hiện chọn ngẫu nhiên 2 vị trí mà không phải phải kho hàng và hoán đổi vị trí cho chúng. Bởi việc hoán đổi vị trí của 2 kho hàng tương trưng cho vị trí bắt đầu lời giải của 2 xe chở hàng thì điều này hoàn toàn không có gì thay đổi trong kết quả cuối cùng.

Bảng 1: Vị trí có dấu gạch ngang là không chọn

θ	1	2	3	4	5	6	7	8	9
0	4	1	0	2	7	5	0	3	6

Reverse subsequence: Nhóm thực hiện chọn ngẫu nhiên 1 vị trí kho hàng trong chuỗi kết quả x trước giả sử là $icur_d$. Tiếp đến, ta sẽ chọn ngẫu nhiên vị trí $i1$ và $i2$ lần lượt từ $[iprev_d+1, icur_d-1]$ và $[icur_d+1, inext_d-1]$. Sau đó, ta thực hiện đảo chuỗi x từ vị trí $i1$ đến $i2$.

Ví dụ: $icur_d = 3, i1 = 2, i2 = 5$

Chuỗi x:

0	1	2	3	4	5	6	7	8	9
0	4	1	0	2	7	5	0	3	6

Kết quả:

0	1	2	3	4	5	6	7	8	9
0	4	7	2	0	1	5	0	3	6

Ví dụ: Do $icur_d = 0$ nên sẽ không có $iprev_d$. Chính vì thế, nhóm đã giải quyết trường hợp này bằng cách gấp cong vector x này lại. Chính vì thế vị trí 9 sẽ liền kề với vị trí 0 và thực hiện đảo chuỗi bình thường. Lưu ý: trong ví dụ này chuỗi con $[3, 6]$ không bị đảo vì nếu ta tưởng tượng việc gấp vector lại sẽ đảo thứ tự của chuỗi con này rồi nên khi đảo lần nữa thì thứ tự vẫn giữ nguyên.

Chuỗi x:

0	1	2	3	4	5	6	7	8	9
0	4	1	0	2	7	5	0	3	6

Kết quả:

0	1	2	3	4	5	6	7	8	9
0	3	6	1	0	2	7	5	0	4

Random swap of reversed subsequence:

Ý tưởng: Chọn ngẫu nhiên 2 subsequence không overlap nhau trong solution vector.

Mỗi subsequence có một giá trị xác suất tương ứng để xác định xem có cần reverse chúng không (xác suất lớn hơn 0.5 thì reverse). Cuối cùng là sẽ swap 2 subsequence này cho nhau. Lưu ý là mỗi subsequence phải bao gồm cả depot và customer.

Các yêu cầu khi swap mà nhóm đặt ra để không xảy ra xung đột khi chạy chương trình là:

- + Không được có 2 vị trí 0 nào kề nhau trong solution vector.
- + Không được có 2 vị trí 0 nào cách nhau 1 ô (nghĩa là khi đã có 0 ở vị trí i thì không được có 0 ở vị trí $i+1$) trong solution vector.
- + Không được có số 0 nằm ở cuối solution vector.

Các bước thực hiện:

- Bước 1: Chọn ngẫu nhiên 2 vị trí 0 (không được chọn vị trí đầu) trong solution vector input.
- Bước 2: Xác định vùng random cho 4 biến start1, end1, start2, end2. Với start1 và end1 là vị trí bắt đầu và kết thúc của subsequence 1, start2 và end2 là vị trí bắt đầu và kết thúc của subsequence 2.
- Bước 3: Random giá trị cho 4 biến trên sao cho trong mỗi subsequence chỉ có đúng 1 giá trị 0.
- Bước 4: Cuối cùng, tiến hành swap và reverse.

Hàm lỗi (cost function): được tính chính như sau

$$z(x) = c(x) + \alpha q(x) + \gamma d(x)$$

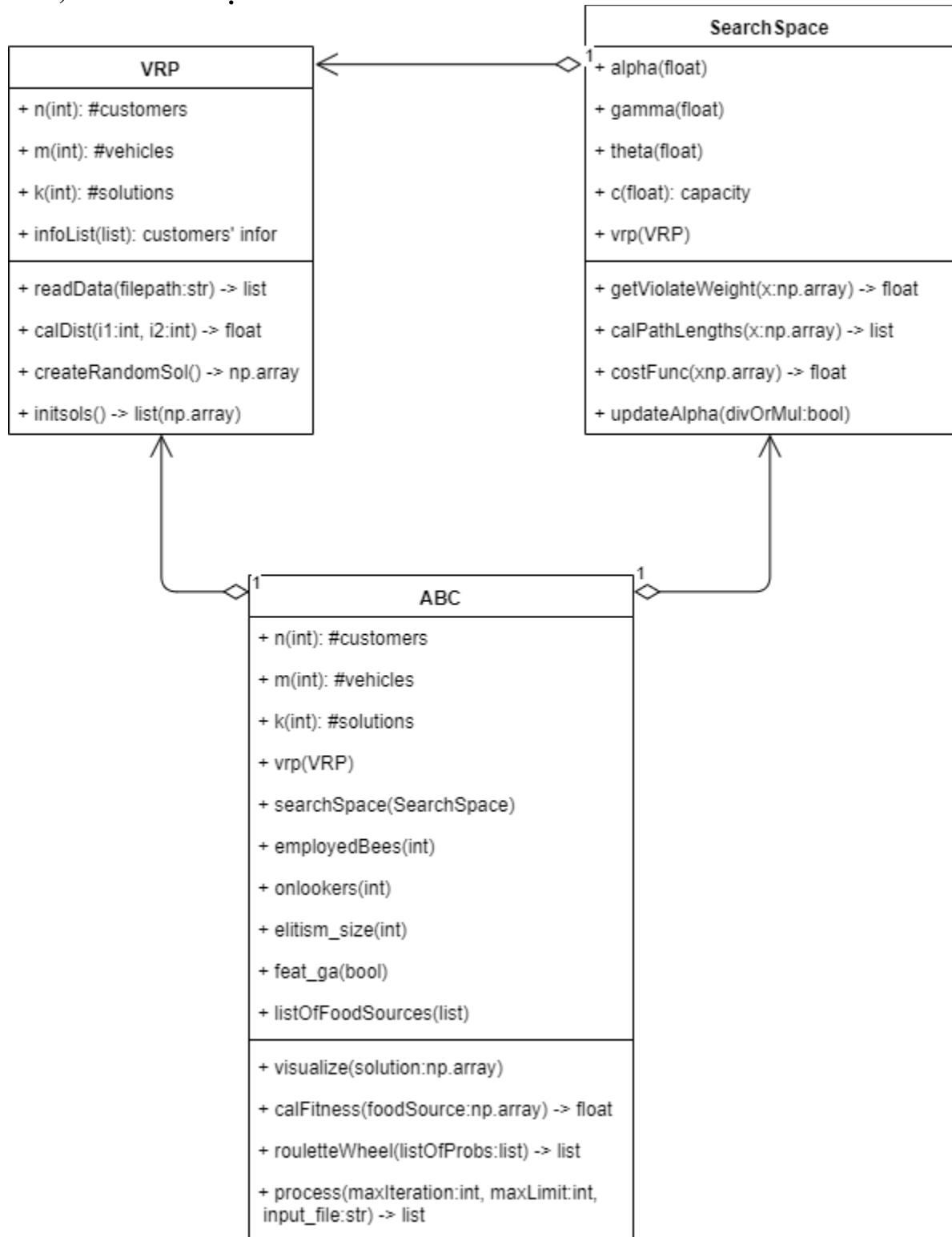
Với:

- $c(x) = \sum_{i,j} c_{i,j}$ (với $c_{i,j}$ là chi phí cạnh (i, j)) là chi phí di chuyển của lộ trình x.
- $q(x)$ là tổng mức độ vi phạm tải trọng trong lộ trình x. Điều kiện là $q(x) < Q$.
- $d(x)$ là tổng mức độ chênh lệch chi phí di chuyển của các m phương tiện.
- Hệ số α được cập nhật nếu số lượng lời giải vi phạm ràng buộc về tải trọng lớn hơn $\tau/2$ thì $\alpha = \frac{\alpha}{1+\delta}$, ngược lại thì $\alpha = \alpha \times (1 + \delta)$.

Kết hợp thêm Genetic cho Bee:

- Giải thích cách breed 2 solution vector x và y:
 - Bước 1: chọn random ngẫu nhiên một subsequence **q** trong x. Xóa tất cả các phần tử 0 (vehicle) trong subsequence **q** này.
 - Bước 2: chọn lại các phần tử tồn tại trong y mà không tồn tại trong **q** và giữ đúng thứ tự của chúng như trong y. Xóa các phần tử 0. => Có được subsequence **p**.
 - Bước 3 (quan trọng): gán lần lượt từng phần tử trong **p** vào vị trí thích hợp trong **q**. Để chọn vị trí thích hợp, cần duyệt mỗi phần tử u_i ($0 \leq i < \text{len}(p)$) trong p qua các vị trí của q. Với mỗi vị trí **j** ($0 \leq j < \text{len}(q)$) trong **q**, ta sẽ tính khoảng cách của khách hàng **ui** với khách hàng thứ **j** và khoảng cách của khách hàng **ui** với khách hàng thứ **j+1**. Vị trí được chọn để chèn vào là vị trí có tổng 2 khoảng cách trên là nhỏ nhất.=> Kết quả có được là chuỗi **children** thể hiện một giải pháp mới.
 - Bước 4: Thêm các giá trị 0 (vehicle) vào chuỗi **children**. Vị trí của mỗi vehicle sẽ được random trong một vùng xác định trước, các vùng random này giữa các vehicle sẽ cách một khoảng đều nhau.
- Thêm breed vào thuật toán Bee: Thay đổi bước c.ii) của bài báo. Thay vì tìm neighbor của các foodsource thì ta tìm neighbor của các children giữa các foodsource.

3) Chi tiết cài đặt:



Hình 1: Sơ đồ UML

a) *VRP*

Thuộc tính	Công dụng
n(int)	Số lượng khách hàng
m(int)	Số lượng phương tiện
k(int)	Số lượng lời giải xem xét
infoList(list)	Thông tin của các khách hàng. Với mỗi phần tử trong list là một tuple của (x, y, q) trong đó (x, y) tọa độ 2 chiều của khách hàng và q là khối lượng đơn hàng.

Phương thức	Công dụng
readData(filepath:str) -> list Tham số: <ul style="list-style-type: none"> filepath: đường dẫn của tập tin dữ liệu. 	Đọc dữ liệu truyền vào từ tập tin văn bản.
calDist(i1:int, i2:int) -> float Tham số: <ul style="list-style-type: none"> i1: index của khách hàng thứ nhất i2: index của khách hàng thứ hai 	Tính khoảng cách giữa 2 khách hàng
createRandomSol() -> np.array	Phát sinh một lời giải (food source) ngẫu nhiên
initSols() -> list(np.array)	Phát sinh k lời giải ngẫu nhiên

b) *SearchSpace*

Thuộc tính	Công dụng
alpha(float)	Hệ số kiểm soát mức độ vượt quá tải trọng của một lời giải x
gamma(float)	Hệ số kiểm soát mức độ chênh lệch khoảng cách giữa m phương tiện trong lời giải x
theta(float)	Hệ số để cập nhật alpha
c(float)	Tải trọng giới hạn của mỗi chiếc xe
vrp(VRP)	Đối tượng của lớp VRP

Phương thức	Công dụng
getViolateWeight(x:np.array) -> float	Trả về mức độ vi phạm tải trọng của lời giải x
calPathLength(x:np.array) -> list	Tính chi phí di chuyển của lời giải x dựa vào khoảng cách Euclide.
costFunc(x: np.array) -> list	Tính độ lỗi cho lời giải x
updateAlpha(divOrMul:bool)	Cập nhật hệ số alpha

c) ABC

Thuộc tính	Công dụng
n(int)	Số lượng khách hàng
m(int)	Số lượng phương tiện
k(int)	Số lượng lời giải xem xét
infoList(list)	Thông tin của các khách hàng. Với mỗi phần tử trong list là một tuple của (x, y, q) trong đó (x, y) tọa độ 2 chiều của khách hàng và q là khối lượng đơn hàng.
vrp(VRP)	Đối tượng của lớp VRP
searchSpace(SearchSpace)	Đối tượng của lớp SearchSpace
employedBees(int)	Số lượng employed bees
onlookers(int)	Số lượng onlookers
elitism_size(int)	Số lượng lời giải tốt được giữ lại cho mỗi vòng lặp
feat_ga(bool)	Flag để sử dụng thuật giải bee kết hợp ga do nhóm thiết kế.
listOfFoodSources(list)	Tập lời giải của hệ thống

Phương thức	Công dụng
visualize(solution:np.array)	Trực quan hóa lời giải
calFitness(foodSource:np.array) -> list	Tính độ thích nghi cho nguồn mật (hay lời giải)
rouletteWheel(listOfProbs:list) -> list	Thực hiện chọn lựa k lời giải theo phương pháp roulette wheel
process(maxIteration:int, maxLimit:int, input_file:str) -> str	Thực hiện chạy thuật giải

II. Kết quả thực nghiệm:

1) Giới thiệu về dữ liệu

Gồm 2 problem (2 tập input) lấy từ bài báo “An Algorithm for the Vehicle-dispatching Problem” [2] của N. Christofides và S. Eilon (1969). 2 problem được dùng để đánh giá là Problem 8 (50 khách hàng) và Problem 9 (75 khách hàng).

Vị trí của Depot (Trung tâm vận chuyển) và Customers (Các khách hàng) sẽ được thể hiện trên bản đồ theo tọa độ Euclid. Lưu ý là không có ràng buộc về đường đi nào giữa 2 vị trí bất kỳ trên bản đồ. Vị trí của Depot được mặc định ở (30, 40).

Mỗi tập input sẽ có n hàng, mỗi hàng tượng trưng cho thông tin của một khách hàng. Thông tin của khách hàng sẽ bao gồm 3 thành phần: hoành độ (x), tung độ (y) và trọng lượng của món hàng mà khách đó yêu cầu (q) (đơn vị cwt, biết rằng 1 cwt = 50.8 ~ 51kg). Tải trọng mặc định của mỗi phương tiện đều là 8000 kg.

2) Bảng thống kê và nhận xét

Để so sánh các giữa các bộ tham số khác nhau và các tùy chỉnh khác nhau của thuật toán Bee Colony, chúng tôi set random seed như trong hình

```
from numpy.random import seed
seed(1)
import random
random.seed(2)
```

Giải thích bộ tham số đầu vào của chương trình:

- n: số lượng khách hàng
- m: số lượng phương tiện vận chuyển
- k: số lượng lời giải trong không gian tìm kiếm tại mỗi giai đoạn
- c: trọng tải của mỗi phương tiện (mặc định theo standard benchmark là 8000 kg)
- loop: số lần lặp
- limit: để set giới hạn cho sau bao nhiêu lần lặp mà một food source vẫn không tốt lên thì cân nhắc thay thế food source đó với neighbor tốt hơn của nó.

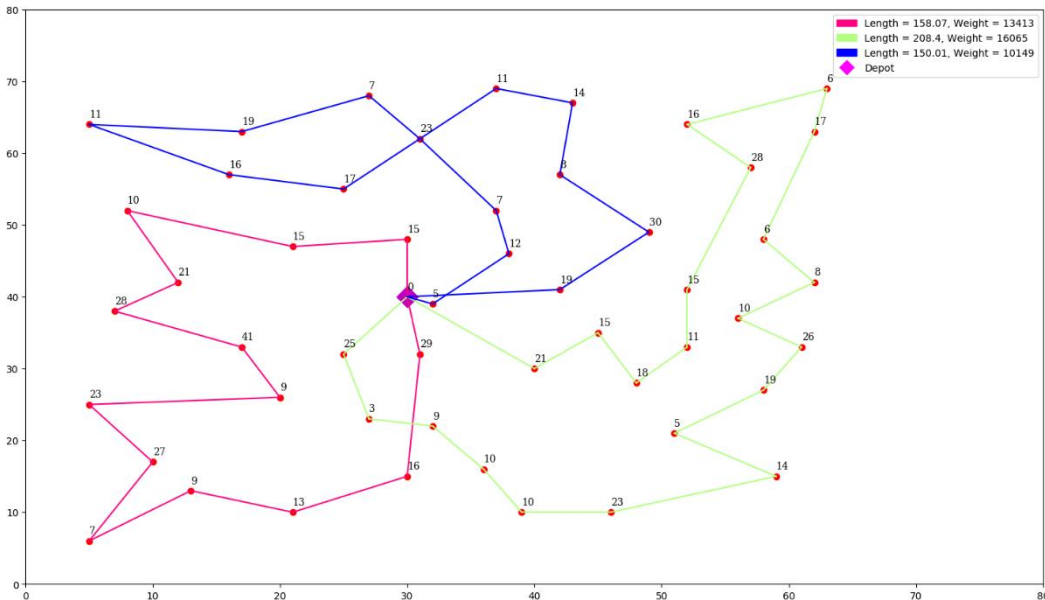
Các tham số thử nghiệm theo tác giả cung cấp và khảo nghiệm bởi nhóm:

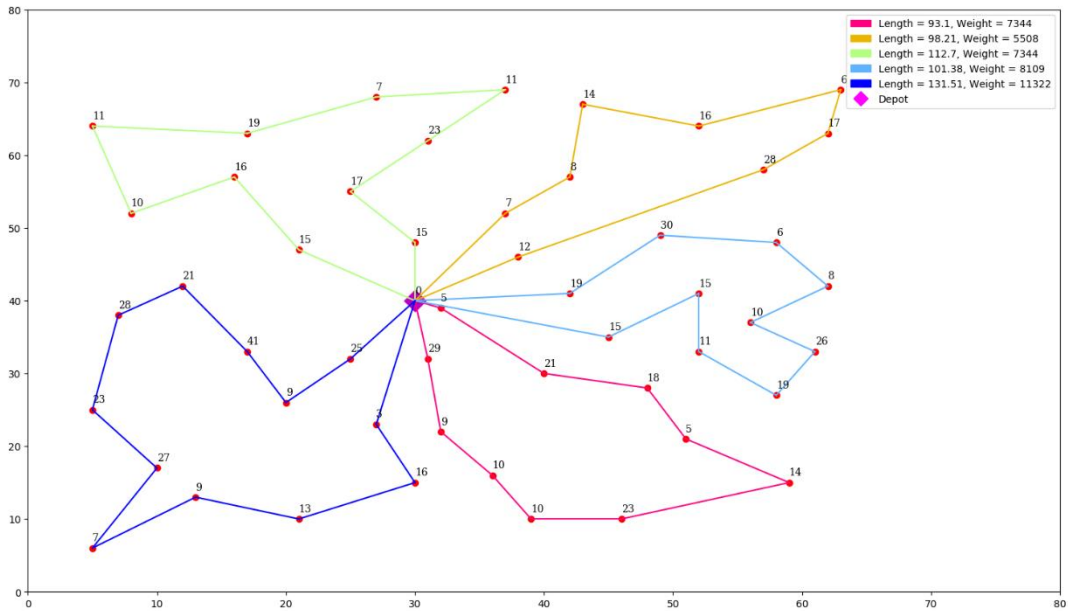
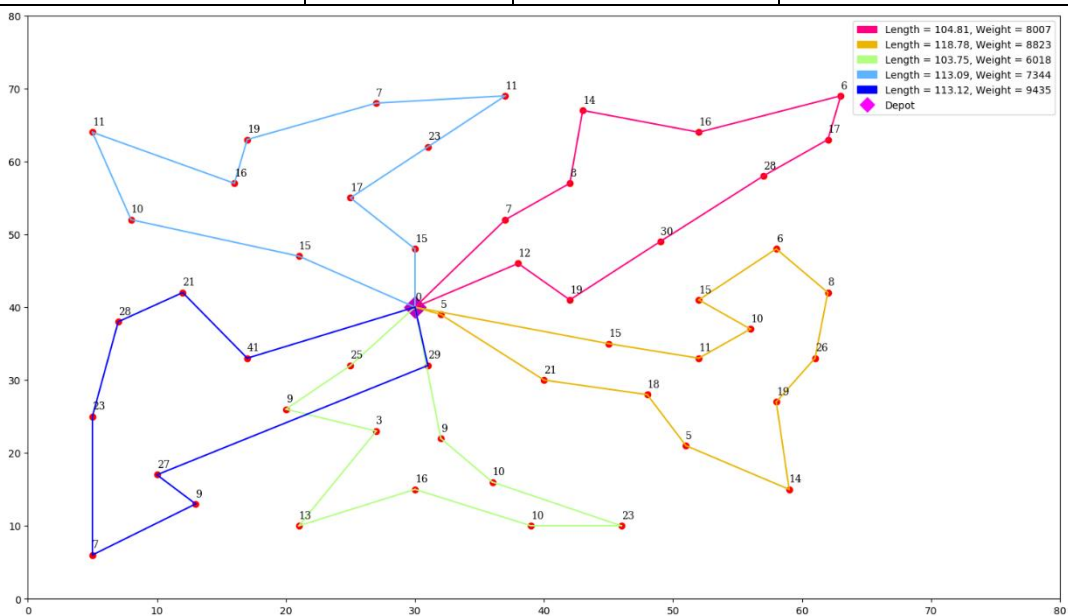
- Số lượng phương tiện: $m \geq 3$ và $(n+m) \geq 7$
- Số lượng lời giải (nguồn mật) xem xét: $k = 25$ (theo bài báo).
- Số lượng employed bees và lookers bằng k (theo bài báo)
- Hệ số $\alpha = 0.1$, $\gamma = 0.1$.
- Hệ số $\theta = 0.001$.
- Số lần lặp: $\text{maxIteration} = 2000n$ (theo bài báo) nhưng nhóm thử nghiệm từ 20000 trở lên thì cũng thấy rằng lời giải tương đối tốt.
- Số lần giới hạn vi phạm tải trọng: $\text{limit} = 50n$ (theo bài báo) nhưng nhóm thông thường thử nghiệm bằng 50.

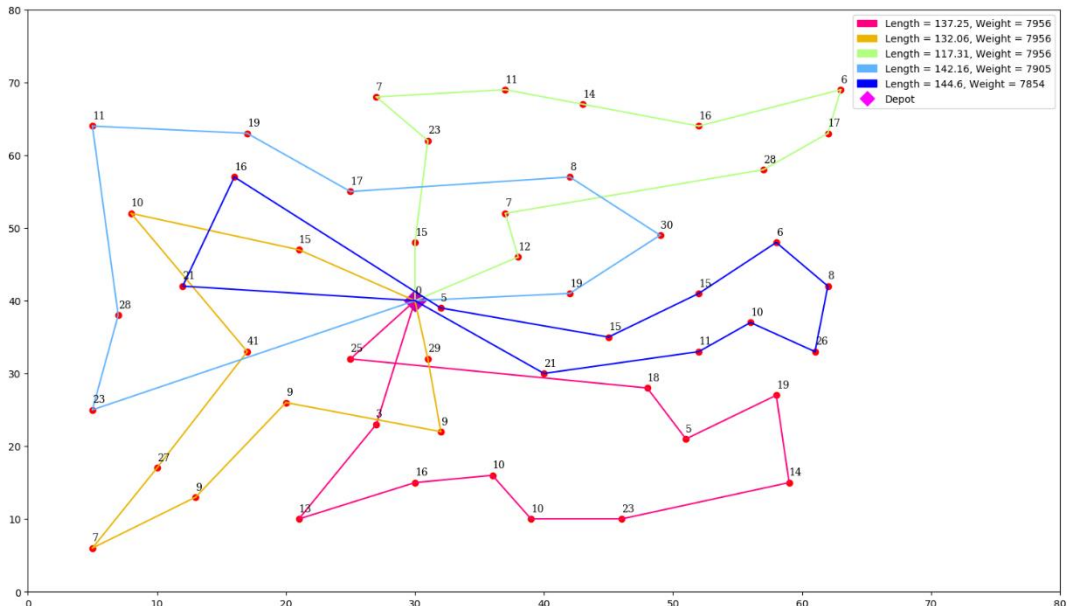
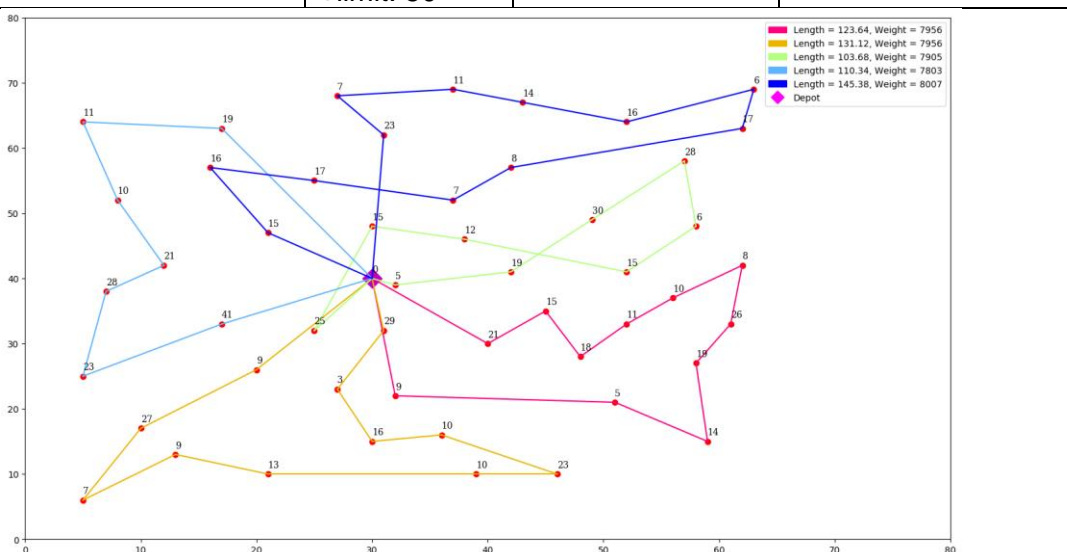
Giải thích ảnh bản đồ trong mỗi trường hợp bên dưới:

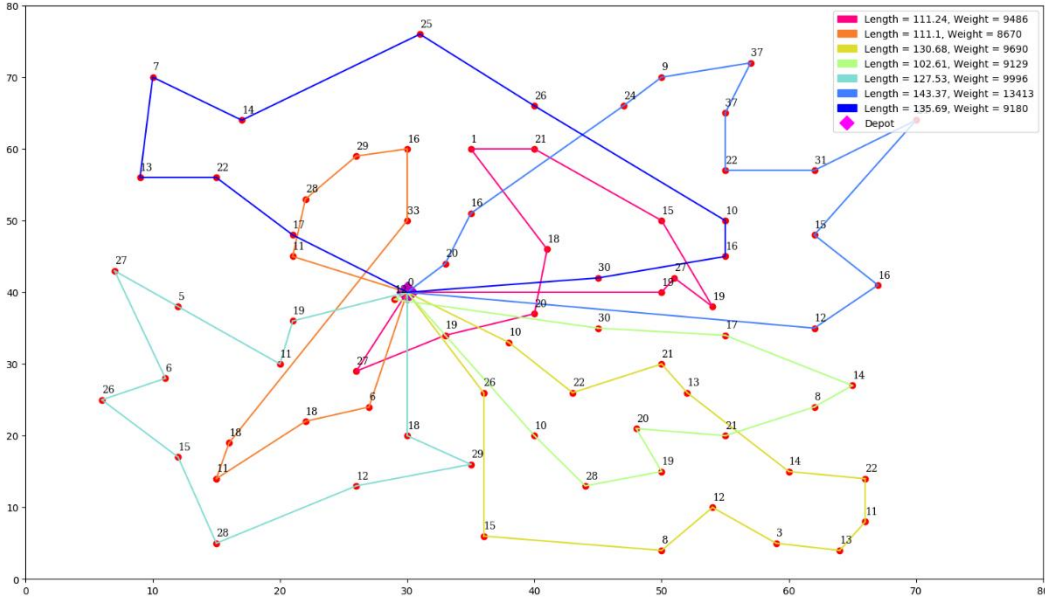
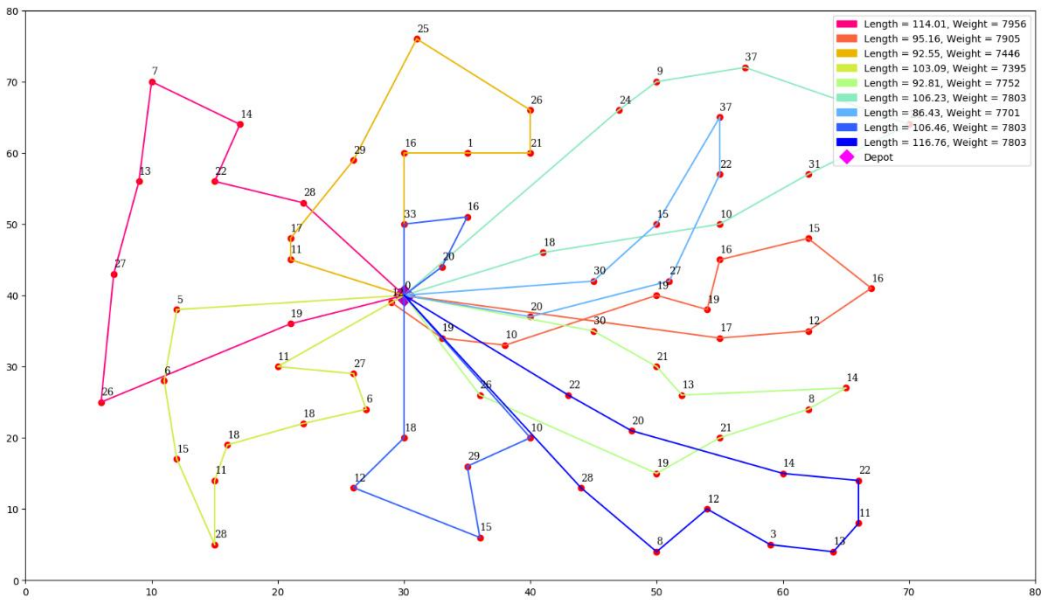
- Hình thoi màu tím là vị trí của depot, mặc định là ở tọa độ (30, 40). Là nơi bắt đầu và kết thúc của mỗi phương tiện.
- Các chấm màu đỏ là vị trí của khách hàng. Số nhỏ trên mỗi chấm là trọng lượng của món hàng mà khách hàng đó yêu cầu (theo đơn vị cwt).
- Đường đi của mỗi phương tiện sẽ được biểu diễn bằng một màu. Trong bảng chú thích, có thông tin về tổng quãng đường đi của mỗi phương tiện và tổng trọng lượng các món hàng mà phương tiện đó phải mang theo.
- Tổng quãng đường đi (Length) là tổng các khoảng cách Euclid giữa 2 điểm kề nhau trên đường đi đó.
- Tổng trọng lượng các món hàng (Weight) trên đường đi được thể hiện theo đơn vị kg trong bảng chú thích.

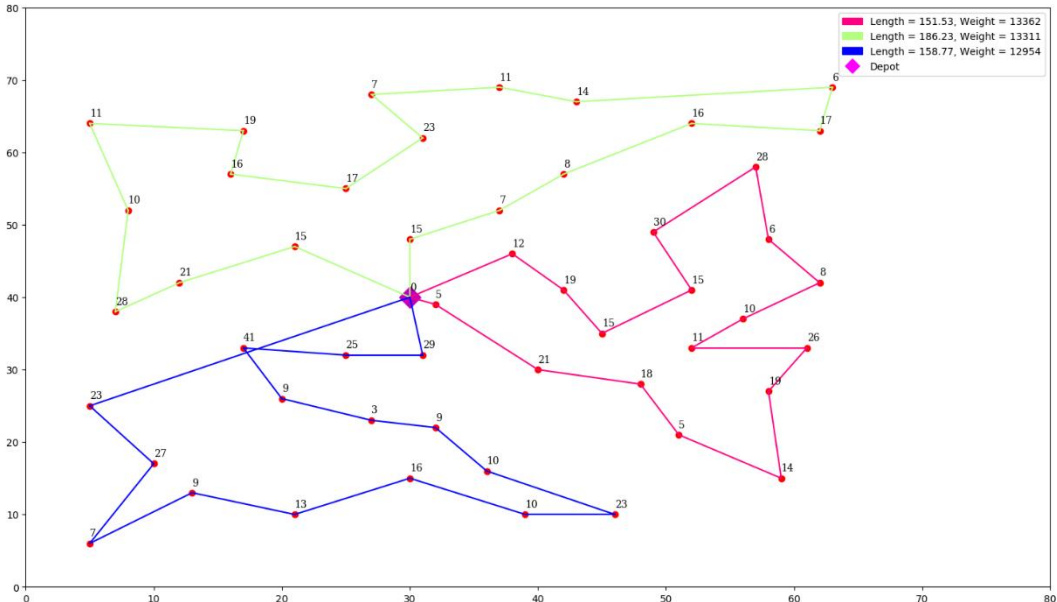
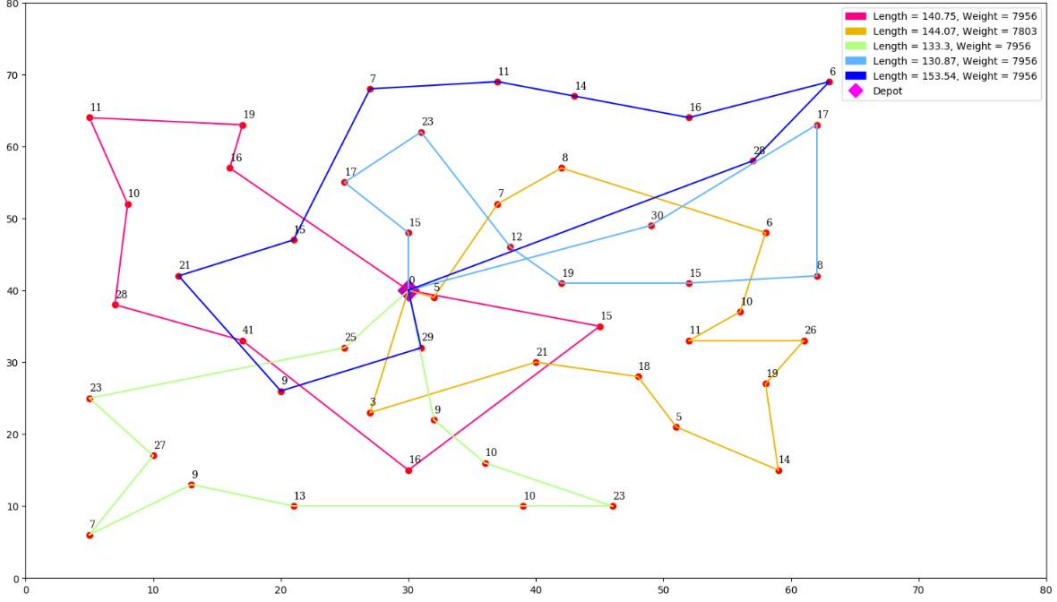
Nhận xét: Khi set cùng một bộ tham số, thì tất cả các ví dụ có kết hợp thêm Breed đều tìm ra đường đi tốt hơn khi chỉ sử dụng Bee Colony.

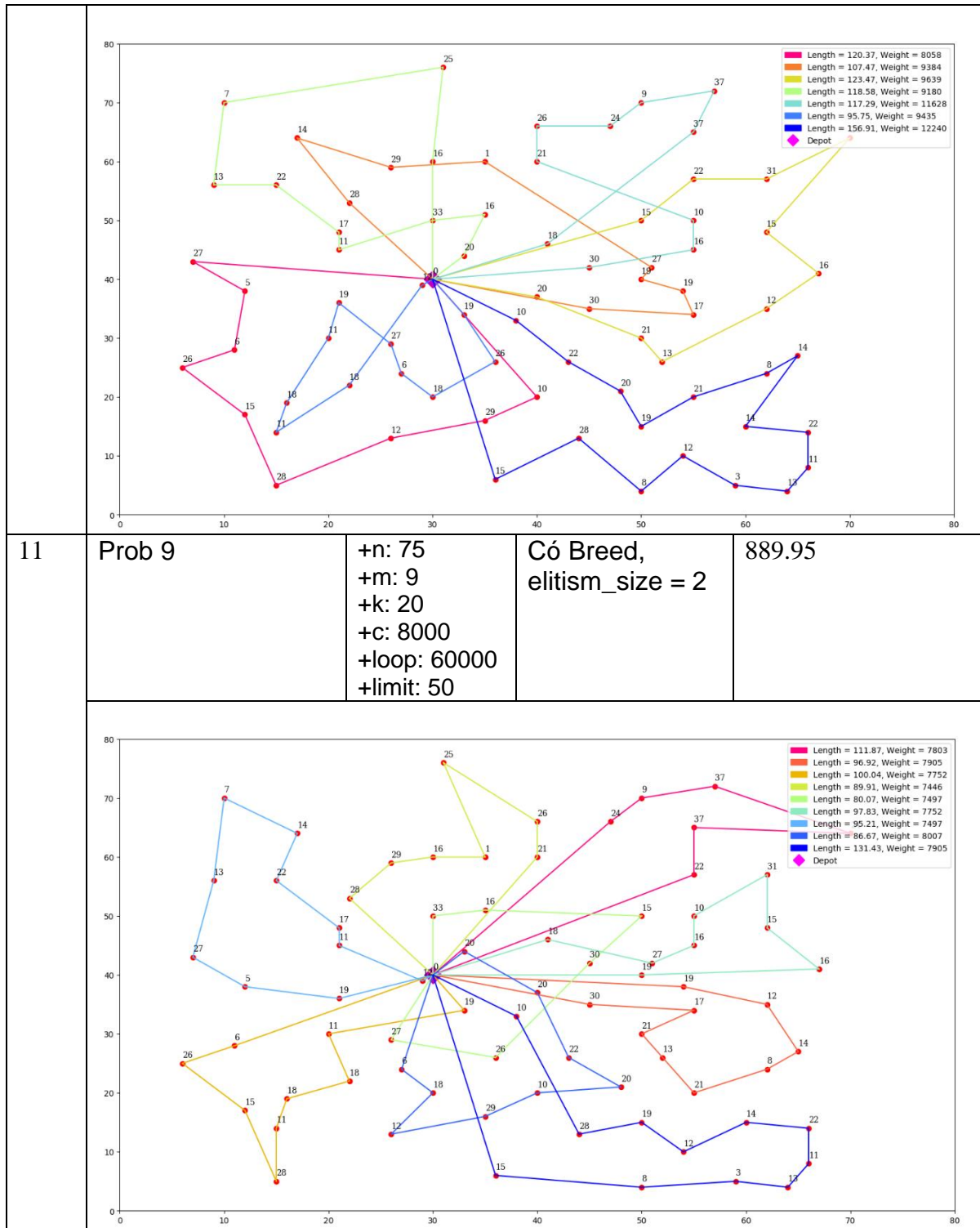
STT	Problem	Parameters	Có kết hợp thêm Genetic (Breed)	Tổng đường đi
1	Prob 8	+n: 50 +m: 3 +k: 10 +c: 8000 +loop: 20000 +limit: 50	Không có Breed	516.48
				
2	Prob 8	+n: 50 +m: 5 +k: 20 +c: 4000 +loop: 20000 +limit: 50	Không có Breed	536.9

				
3	Prob 8	+n: 50 +m: 5 +k: 20 +c: 6000 +loop: 20000 +limit: 50	Không có Breed	553.55
				
4	Prob 8	+n: 50 +m: 5 +k: 20 +c: 8000 +loop: 20000 +limit: 50	Không có Breed	673.38

				
5	Prob 8	+n: 50 +m: 5 +k: 20 +c: 8000 +loop: 50000 +limit: 50	Không có Breed	614.16
				
6	Prob 9	+n: 75 +m: 7 +k: 10 +c: 8000 +loop: 60000 +limit: 50	Không có Breed	862.22

				
7	<p>Prob 9</p>	<p>+n: 75 +m: 9 +k: 20 +c: 8000 +loop: 60000 +limit: 50</p>	<p>Không có Breed</p>	<p>913.5</p>
				
8	<p>Prob 8</p>	<p>+n: 50 +m: 3 +k: 10 +c: 8000 +loop: 20000 +limit: 50</p>	<p>Có Breed, elitism_size = 2</p>	<p>496.53</p>

				
9	Prob 8	+n: 50 +m: 5 +k: 20 +c: 8000 +loop: 50000 +limit: 50	Có Breed, elitism_size = 2	702.53
				
10	Prob 9	+n: 75 +m: 7 +k: 20 +c: 8000 +loop: 60000 +limit: 50	Có Breed, elitism_size = 2	839.84



D

Hướng dẫn sử dụng:

Xem tập tin ReadMe.md đính kèm.

F

Tham khảo:

[1]	W. Y. Szeto, Y. Wu, and S. C. Ho, "An artificial bee colony algorithm for the capacitated vehicle routing problem," European Journal of Operational Research, vol. 215, no. 1, pp. 126–135, 2011.
[2]	Christofides, Nicos, and Samuel Eilon. "An algorithm for the vehicle-dispatching problem." Journal of the Operational Research Society 20.3 (1969): 309-318.