# Ý tưởng

Ý tưởng của em áp dụng thuật toán GA để lai ghép các hành trình lịch sử giao hàng của target , lịch sử này được khởi tạo cẩn thận từ initsolution do đó kết quả xuất phát khá tiệm cận với kết quả tối đa của hàm mục tiêu

# Sơ đồ thuật toán chung

Sơ đồ thuật toán áp dụng GA cụ thể như sau

Chart, diagram, box and whisker chart

Description automatically generated

Hình 1 Sơ đồ thuật toán GA (cách 2)

Cụ thể các bước trên như sau

* Nhận các tham số của bài toán: Đọc từ file csv các thông số như ở bước mô hình hoá bài toán
* Tính khoảng cách giữa các khách hàng: Lập một ma trận khoảng cách giữa các khách hàng với nhau
* Kiểm tra điều kiện dừng: Nếu fitness không tăng sau 50 vòng lặp thì sẽ dừng

Các phần tiếp theo sẽ được giải thích kỹ ở các mục sau

## Mã hoá lời giải

Mã hoá lời giải: Mỗi lời giải gồm 2 trường sau đây:

* Lịch trình di chuyển drone: là lịch trình di chuyển của drone đi qua từng target và tại target đó giao bao nhiêu hàng. Cụ thể như sau
  + Ví dụ Drone\_1 = [[1,10],[2,20]],[[5,50][6,60]]
    - Phần màu đỏ là hành trình thứ nhất của drone 1 trong đó ghé thăm điểm khách hàng số 1 vào giao khối lượng 10, sau đó đến khách hàng 2 và giao khối lượng 20 rồi quay trở về kho
* Lịch sử nhận hàng của từng khách hàng: là lịch trình mà khách hàng đó nhận hàng của thiết bị nào và nhận trọng lượng là bao nhiêu. Cụ thể cách mã hoá được biểu diễn trong hình lịch sử nhận hàng bên dưới.
  + Ví dụ Customer\_1 = [[1,10], [2,10]]
    - Phần màu đỏ là ở lượt thứ nhất được thiết bị drone (id =1) giao khối lượng 10, sau đó đến drone (id = 2) giao khối lượng 10

Số liệu ở 2 trường đều đồng bộ với nhau, sở dĩ phải chia làm 2 trường để dễ dàng cho hàm kiểm soát giá trị rằng luôn thoả mãn ràng buộc của bài toán ở cả khách hàng và drone

## Init solution

Init solution gồm 3 cách khởi tạo:

### Init solution 1

Drone chọn điểm xuất phát là điểm gần nhất với nó, từ điểm đó di chuyển tới các điểm gần nhất và giao lower\_bound (nếu lower\_bound = 0 thì bỏ qua không cần giao) , dừng khi:

* Hết hàng để giao
* Hết thời gian duration của drone
* Hết thời gian working time
* Hết target

Sau đó tiếp tục đến xe tải, chọn điểm xa nhất làm điểm xuất phát rồi đi tới các điểm gần nhất cho đến khi đạt tới điều kiện dừng:

* Hết hàng để giao
* Hết thời gian working time
* Hết target

### Init solution 2

Drone chọn điểm xuất phát là điểm xa nhất với nó, từ điểm đó di chuyển tới các điểm gần nhất và giao lower\_bound (nếu lower\_bound = 0 thì bỏ qua không cần giao) , dừng khi:

* Hết hàng để giao
* Hết thời gian duration của drone
* Hết thời gian working time
* Hết target

Sau đó tiếp tục đến xe tải, chọn điểm gần nhất làm điểm xuất phát rồi đi tới các điểm gần nhất cho đến khi đạt tới điều kiện dừng:

* Hết hàng để giao
* Hết thời gian working time
* Hết target

### Init solution 3

Drone chọn điểm xuất phát là điểm bất kỳ, từ điểm đó di chuyển tới các điểm gần nhất và giao lower\_bound (nếu lower\_bound = 0 thì bỏ qua không cần giao) , dừng khi:

* Hết hàng để giao
* Hết thời gian duration của drone
* Hết thời gian working time
* Hết target

Sau đó tiếp tục đến xe tải cũng chọn điểm bất kỳ làm điểm xuất phát rồi đi tới các điểm gần nhất cho đến khi đạt tới điều kiện dừng:

* Hết hàng để giao
* Hết thời gian working time
* Hết target

Từ các init solution này em có thể tạo ra gần 100 init solution khác nhau trong khi triển khai. Kết quả của các init solution này cũng không tệ ( gần đạt feasible)

## Chọn lọc

### Độ thích nghi

Để tính độ thích nghi trước hết em dựa vào biến penalty để tính giá trị phạt, cụ thể hàm penalty em tính dưạ trên 3 thành phần chính:

* Phạt dựa trên việc giao cho target thiếu lower\_bound, nếu giao thiếu sẽ bị trừ trọng số nhân với tỷ lệ đã được chuẩn hoá theo trọng số max

Text

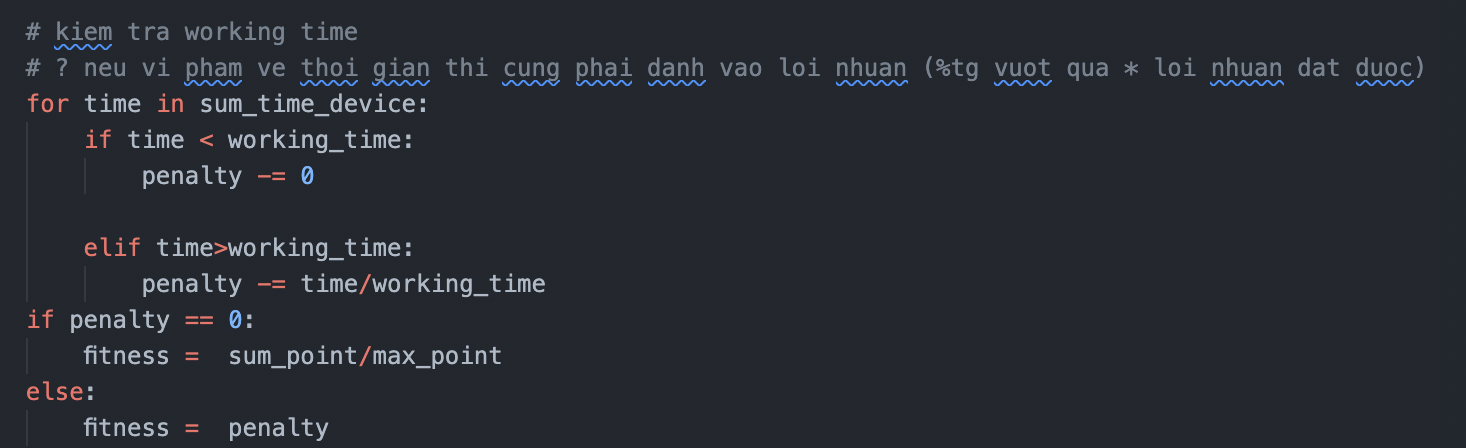
Description automatically generated

* Phạt dựa trên sự vi phạm của drone qua từng trip, nếu như trong 1 trip của drone mà vượt quá thời gian vi phạm cũng sẽ bị trừ điểm

Text

Description automatically generated

* Phạt dựa trên sự vi phạm working time của tất cả các thiết bị



Từ kết quả penalty tính ra em chia làm 2 trường hợp:

* Nếu penalty = 0 thì fitness sẽ bằng
  + Fitness = Giá trị chuẩn hoá hàm mục tiêu
* Nếu penalty < 0 thì fitness sẽ bằng
  + Fitness = penalty

### Chọn lọc

Dựa trên vòng quay roulette, chọn ngẫu nhiên các cá thể, các cá thể có thể xuất hiện nhiều lần trong quần thể mới.

* Đoạn code xây dựng vòng quay roulette như sau

Text

Description automatically generated

* Đoạn code lựa chọn ngẫu nhiên các cá thể trong vòng quay này

Graphical user interface, text, application

Description automatically generated

## Lai ghép

Chọn cá thể lai ghép có 2 cách, khi chọn có thể chọn ngẫu nhiên theo 2 cách sau:

* Dựa trên ma trận khác biệt
* Dựa trên chọn ngẫu nhiên

Lai ghép: Lai ghép giữa từng lịch sử nhận hàng của từng khách hàng ở cá thế này với từng lịch sử nhận hàng của từng khách hàng ở cá thể khác bằng cắch cắt từ một số ngâũ nhiên k ( điểm bắt đầu) và p( điểm kết thúc) và ném sang lịch sử nhận hàng phía bên kia cũng tại một số ngẫu nhiên p. Cụ thể như hình vẽ sau:

A picture containing schematic

Description automatically generated

Hình 2 Hình vẽ mô tả quá trình lai ghép

Ngoài ra trong khi triển khai em có đặt một tỷ lệ nhất định khi lai ghép chỉ cần chọn k, các điểm khác mặc định là điểm cuối cùng trong lịch sử để đảm bảo giữ được cấu trúc phân bố trong target nhiều nhất có thể

## Đột biến

Chọn ngẫu nhiên các cá thể để đột biến trong quần thể cũ, việc đột biến sẽ rơi vào một trong 3 trường hợp

* thực hiện đột biến khối lượng bất kỳ ở trong lịch sử giao hàng của từng khách hàng:
  + Ví dụ

|  |  |  |
| --- | --- | --- |
| Id target | Trước khi đột biến | Sau khi đột biến |
| 18 | [[0,25],[2,400]] | [[0,50],[2,400]] |

* Thực hiện đột biến tăng thêm trip mới cho target

|  |  |  |
| --- | --- | --- |
| Id target | Trước khi đột biến | Sau khi đột biến |
| 18 | [[0,25],[2,400]] | [[0,25],[2,400][3,10]] |

* Thực hiện đột biến xoá bớt trip cho target

|  |  |  |
| --- | --- | --- |
| Id target | Trước khi đột biến | Sau khi đột biến |
| 18 | [[0,25],[2,400]] | [[0,25]] |

## Education

Lựa chọn cá thể phải education: Chọn cá thể có fitness kém nhất

Hàm education học theo 2 hướng:

* Theo target: nếu target nào đang bị thiếu lowerbound thì sẽ được tăng khối lượng, nếu bị vượt quá upper\_bound sẽ được giảm khối lương
* Theo thiết bị: nếu target nào đang bị vượt quá giới hạn tải trọng thì tiến hành giảm khối lượng ở turn có số lượng hàng lớn nhất

# Lựa chọn tham số

# Kết quả thu được

Kết quả trên bộ 20.10.1

Chart, box and whisker chart

Description automatically generated

Hành trình tính ra như sau

Chart, line chart

Description automatically generated

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Data Name | working time | number truck | number drone | Feasible | best solution | số thế hệ |
| 6.5.1 | 60 | 2 | 2 | Có | 2100 | 3 |
| 10.5.1 | 60 | 2 | 2 | Có | 28700 | 1 |
| 20.5.1 | 60 | 2 | 2 | Có | 53100 | 25 |
| 20.10.1 | 60 | 2 | 2 | Có | 56725 | 45 |
| 50.10.1 | 60 | 4 | 5 | Có | 95750 | 100 |
| 100.10.1 | 60 | 8 | 6 | Có | 165650 | 10 |
| 200.10.1 | 60 | 20 | 30 | Có | 396250 | 10 |

# Kết luận