

The background of the entire image is a solid red color. Overlaid on this background is a large, stylized arch made of numerous small, light-red dots. The dots are arranged in a way that they form a continuous, flowing line that curves from the left side, peaks in the upper center, and curves down towards the right side. The dots are more densely packed in the center of the arch and become sparser towards the edges.

# HUST

**ĐẠI HỌC BÁCH KHOA HÀ NỘI**  
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.



**ĐẠI HỌC  
BÁCH KHOA HÀ NỘI**  
HANOI UNIVERSITY  
OF SCIENCE AND TECHNOLOGY

# Bài Tập Lớn Tối Ưu Lập Kế Hoạch

GHVD: TS. Bùi Quốc Trung

Nhóm: 21

Sinh viên thực hiện:

Nguyễn Long Nhật - 20215440

Nguyễn Bá Minh Đạt - 20215342

Nguyễn Đức Nghĩa - 20204674

Trần Quốc Nam Phi - 20200459

ONE LOVE. ONE FUTURE.

1. Giới thiệu bài toán
2. Các phương pháp được áp dụng
3. Kết quả thử nghiệm và Đánh giá

# 1. Giới thiệu bài toán

- Có  $N$  đơn hàng  $1, \dots, N$ . Trong đó, mỗi đơn hàng  $j$  có 2 giá trị cần quan tâm là trọng lượng  $c(j)$  và giá trị  $p(j)$
- Có  $K$  phương tiện  $1, 2, \dots, K$  để phục vụ các đơn hàng trong đó mỗi phương tiện  $i$  có cận dưới của tải trọng  $l(i)$  (lower bound) và cận trên của tải trọng  $u(i)$  (upper bound)
- Bài toán yêu cầu tìm lời giải sắp xếp các đơn hàng vào các phương tiện phục vụ sao cho thoả mãn:
  - Mỗi đơn hàng chỉ được phục vụ bởi 1 phương tiện
  - Tổng trọng lượng các đơn hàng trên mỗi phương tiện phải nằm trong khoảng cho phép (từ cận dưới tới không vượt quá cận trên của phương tiện)
  - Tổng giá trị các đơn hàng được phục vụ là lớn nhất

# 1. Giới thiệu bài toán

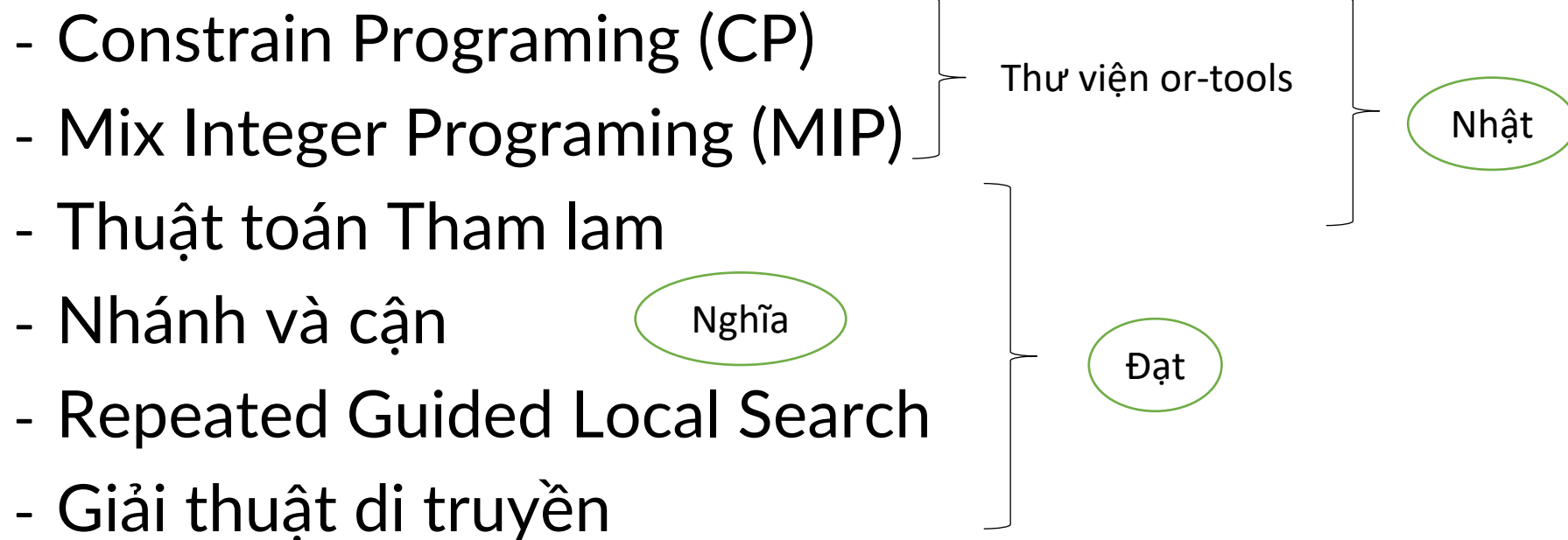
Input:

- $N$ : Số đơn hàng ( $1 \leq N \leq 1000$ )
- $K$ : Số phương tiện ( $1 \leq K \leq 100$ )
- $c_j$ : Trọng lượng của đơn hàng  $j$   $\forall j \in [1, N]$
- $p_j$ : Giá trị của đơn hàng  $j$   $\forall j \in [1, N]$
- $l_i$ : Cận dưới của tải trọng xe  $i$   $\forall i \in [1, K]$
- $u_i$ : Cận trên của tải trọng xe  $i$   $\forall i \in [1, K]$

Output:

- Một số nguyên  $m$  biểu thị số đơn hàng được phục vụ
- $m$  dòng tiếp theo là chi tiết đơn hàng  $j$  được phục vụ bởi xe  $i$   $\forall i \in [1, K], \forall j \in [1, N]$

## 2. Các phương pháp được áp dụng



# “Nhánh và cận” Giá trị lớn nhất có thể thêm được

Ta sẽ tính toán xem tổng giá trị còn lại có thể thêm được có lớn hơn giá trị hiện có hay không.

Tại lần duyệt nhánh cận thứ  $m$ , ta sắp xếp lại trọng tải của các đơn hàng theo thứ tự tăng dần để thu được mảng

$$C = \{c_i \mid m \leq i < n \ \& \ c_i < c_j \ (\forall i < j)\}.$$

Tương tự, ta sắp xếp giá trị của các đơn hàng theo thứ tự giảm dần, ta sẽ thu được mảng

$$P = \{p_i \mid m \leq i < n \ \& \ (p_i > p_j)(\forall i < j)\}.$$

Với 2 mảng trên, ta khởi tạo biến ***iterator\_C = 0***, ***iterator\_P = P.size() - 1***. Hai biến trên để đánh dấu trọng tải nhỏ nhất và giá trị lớn nhất.

Cuối cùng, ta duyệt qua danh sách các xe. Nếu xe thứ  $i$  có đủ trọng tải để chứa thêm ***C[iterator\_C]***, ta sẽ tăng ***iterator\_C*** lên 1 và giảm ***iterator\_P*** đi 1. Sau đó cập nhật lại tổng giá trị trên xe.

Thuật toán dừng lại khi ***duyệt hết mảng C, P hoặc duyệt qua hết tất cả các xe.***

Nếu tổng giá trị sau khi thực hiện thuật toán nhỏ hơn tổng giá trị hiện tại. Ta sẽ cắt "cận" tại đây.

Nghĩa

# 2.1 Constraint Programing (CP)

## Mô hình hoá bài toán:

Biến:

$X[i, j] = 0$  nếu xe  $i$  không chứa kiện hàng  $j$

$$\forall i \in [1, K], j \in [1, N]$$

$X[i, j] = 1$  nếu xe  $i$  chứa kiện hàng  $j$

$$\forall i \in [1, K], j \in [1, N]$$

Ràng buộc:

- Tổng trọng lượng của các đơn hàng trong mỗi phương tiện phải nằm trong khoảng cho phép

$$l_i \leq \sum_{j=1}^N X[i, j] * c_j \leq u_i \quad \forall i \in [1, K]$$

- Mỗi đơn hàng chỉ được phục vụ bởi 1 xe

$$\sum_{j=1}^N X[i, j] \leq 1 \quad \forall i \in [1, K]$$

Hàm mục tiêu:  $Max : \sum_{i=1}^K \sum_{j=1}^N X[i, j] * p_i$

Nhật



# 2.1 Constraint Programing (CP)

- Khai báo solver: SAT - Sử dụng thư viện Or-tools của Google
- Xây dựng các biến quyết định, ràng buộc, hàm mục tiêu theo mô hình hoá ở trên

- Biến:

```
# Variables
x = {}
for i in range(N):
    for k in range(K):
        x[i, k] = model.NewBoolVar(f'x[{i},{k}]')
```

- Ràng buộc:

```
# Constraints
#Sum of quantity of orders loaded (served) in a vehicle must be between
# the low-capacity and up-capacity of that vehicle
for k in range(K):
    load = sum(x[i, k] * orders[i][0] for i in range(N))
    model.Add(load >= vehicles[k][0])
    model.Add(load <= vehicles[k][1])
#Each order is served by at most one vehicle
for i in range(N):
    model.Add(sum(x[i, k] for k in range(K)) <= 1)
```

- Hàm mục tiêu

```
# Objective: Maximize the total value of the items assigned
total_value = sum(x[i, k] * orders[i][1] for i in range(N) for k in range(K))
model.Maximize(total_value)
```

Nhật

## 2.2 Mix Integer Programing (MIP)

- Tái sử dụng lại mô hình hoá ở trên do có tất cả các ràng buộc và hàm mục tiêu đều tuyến tính
- Khai báo solver với "SCIP":
- Thêm các biến quyết định, ràng buộc, hàm mục tiêu theo mô hình hoá giống dùng với CP model

Nhật



## 2.3 Thuật toán tham lam - 1

- Ý tưởng chính:

- Làm thoả mãn các ràng buộc được ưu tiên hơn trước tiên (ràng buộc về cận dưới của tải trọng phương tiện)
- Tham lam lựa chọn đơn hàng có "giá trị" cao để giao cho phương tiện phục vụ

Nhật



## 2.3 Thuật toán tham lam - 1

- Lựa chọn các tiêu chí để tham lam:
- Đánh giá các đơn hàng theo mức độ ưu tiên giảm dần về giá trị của các tiêu chí:
  - Hiệu quả =  $\text{value} / \text{weight}$
  - Value
  - Weight
- Đánh giá các phương tiện theo mức độ ưu tiên tăng dần về giá trị của các tiêu chí:
  - Lower\_bound: Cận dưới
  - Upper\_bound: Cận trên
  - Sự chênh lệch: Cận trên - cận dưới

Nhật




## 2.3 Thuật toán tham lam - 1

- Greedy: 2 phần
- Phần 1: Đảm bảo thoả mãn ràng buộc ưu tiên (cận dưới)
- Duyệt qua tổ hợp các tiêu chí ưu tiên về đơn hàng và phương tiện và xem xét trong không gian tìm kiếm lời giải, nếu phương tiện  $k$  có tải trọng thực sự đang chứa chưa đạt đủ cận dưới và phương tiện  $k$  này phục vụ thêm đơn hàng  $i$  thì tải trọng vẫn chưa vượt quá cận trên thì ta ngay lập tức gán đơn hàng  $i$  cho phương tiện  $k$  phục vụ
  - $\text{real\_load}[\text{veh\_k}] < \text{lower\_bound}[\text{veh\_k}]$  and  $\text{real\_load}[\text{veh\_k}] + \text{quantity\_ord}[\text{ord\_i}] \leq \text{upper\_bound}[\text{veh\_k}]$
- Phần 2: Nhét hàng lên xe sao cho không vượt quá cận trên
- Sau phần 1, duyệt qua các phương tiện, nếu tồn tại một phương tiện nào đó chưa thoả mãn ràng buộc về cận dưới thì trả về lỗi vì không thể thoả mãn điều kiện cận dưới tải trọng
  - $\text{real\_load}[\text{veh\_k}] < \text{lower\_bound}[\text{veh\_k}]$
- Sau khi đảm bảo về điều kiện cận dưới của các phương tiện, ta duyệt qua các đơn hàng còn lại và các phương tiện theo các tiêu chí ưu tiên. Nếu tải trọng thực sự của phương tiện  $k$  có thể chứa thêm đơn hàng  $i$  (chưa được gán cho phương tiện nào) mà vẫn đảm bảo ràng buộc về tải trọng tối đa thì gán đơn hàng  $i$  cho phương tiện  $k$ 
  - $\text{real\_load}[\text{veh\_k}] + \text{quantity\_ord}[\text{ord\_i}] \leq \text{upper\_bound}[\text{veh\_k}]$






Nhật


## 2.3 Thuật toán tham lam - 1

### Message

Evaluated 

### Test cases

Point	Message	Graded	Detail
0	jury solution = 2037 participant solution = 0	Y	
0	jury solution = 9395 participant solution = 0	Y	
100	Perfect, jury solution = 1065 participant solution = 1139	Y	
100	Perfect, jury solution = 5109 participant solution = 5226	Y	
0	jury solution = 8536 participant solution = 0	Y	

5 hàng  |< < 1-5 của 10 > >|

### Submission details

**Status**  
Evaluated

**Pass**  
\_ / 10 test cases

**Point**  
700

**Language**  
PYTHON3

**Total runtime**  
1 038 ms

**Submitted by**  
nhat.nl215440@sis.hust.edu.vn

**Submitted at**  
11/06/2024 14:54:25

**Last modified**  
11/06/2024 14:55:15

**Problem**  
[BIN\\_PACKING\\_LOW\\_UP\\_CAPACITY](#)

**Contest**  
[20232\\_TULKH\\_149488\\_Project](#)

Nhật


## 2.3 Thuật toán tham lam - 2

- Thuật toán tham lam thứ 2 có ý tưởng như sau:
- Bước 1: Sắp xếp đơn hàng
  - Sắp xếp các đơn hàng theo giá trị giảm dần (value).
  - Nếu hai đơn hàng có giá trị bằng nhau, ưu tiên đơn hàng có trọng lượng (weight) nhỏ hơn.
- Bước 2: Phân phối đơn hàng vào phương tiện
  - Duyệt qua các đơn hàng theo thứ tự đã sắp xếp.
  - Xếp từng đơn hàng lên các phương tiện lần lượt cho đến khi phương tiện đạt tải trọng tối đa (upper bound).
    - $\text{real\_load}[\text{veh\_k}] + \text{quantity\_ord}[\text{ord\_i}] \leq \text{upper\_bound}[\text{veh\_k}]$
  - Chuyển sang phương tiện tiếp theo nếu phương tiện hiện tại không thể chứa thêm đơn hàng nào.
- Bước 3: Kiểm tra ràng buộc tải trọng
  - Duyệt qua các phương tiện đã được xếp đơn hàng.
  - Kiểm tra ràng buộc tải trọng nằm trong có nằm trong đoạn từ lower bound đến upper bound hay không
    - $\text{real\_load}[\text{veh\_k}] \geq \text{lower\_bound}[\text{veh\_k}] \ \&\& \ \text{real\_load}[\text{veh\_k}] \leq \text{upper\_bound}[\text{veh\_k}]$ 
      - Nếu phương tiện thỏa mãn ràng buộc tải trọng, thêm vào không gian lời giải.
      - Nếu không thỏa mãn, không thêm phương tiện và các đơn hàng trên phương tiện đó khỏi không gian lời giải.






Nhật


## 2.3 Thuật toán tham lam - 2

### Message

Evaluated 

### Test cases

Point	Message	Graded	Detail
100	Perfect, jury solution = 2037 participant solution = 2049	Y	
99	jury solution = 9395 participant solution = 9362	Y	
100	Perfect, jury solution = 5109 participant solution = 5193	Y	
98	jury solution = 8536 participant solution = 8429	Y	
99	jury solution = 1065 participant solution = 1063	Y	

5 hàng  |< < 1-5 của 10 > >|

### Submission details

**Status**  
Evaluated

**Pass**  
\_ / 10 test cases

**Point**  
994

**Language**  
PYTHON3

**Total runtime**  
584 ms

**Submitted by**  
nhat.nl215440@sis.hust.edu.vn

**Submitted at**  
29/06/2024 21:24:12

**Last modified**  
29/06/2024 21:25:01

**Problem**  
[BIN\\_PACKING\\_LOW\\_UP\\_CAPACITY](#)

**Contest**  
[20232\\_TULKH\\_149488\\_Project](#)

Nhật



# “Nhánh và cận” Khả thi

Ta mong muốn cắt bớt càng nhiều nhánh của cây tìm kiếm càng tốt.

Thay vì sắp xếp tất cả các đơn hàng vào các xe; ta xếp chúng vào từng xe.

Tức là ta giải lần lượt K bài toán con với 1 xe và (một phần) các đơn hàng

Các “cận” được áp dụng để dừng nhánh gồm:

- Tổng trọng lượng các đơn còn lại không đủ xếp vào các xe

$$remain < \sum_{i=1}^k (1 - \tau_i) l_i$$

$\tau_i = 1$  nghĩa là xe  $i$  đã xếp xong

- Không còn đơn hàng có trọng lượng thỏa mãn tải trọng

$$- accum + c_i > u_j,$$

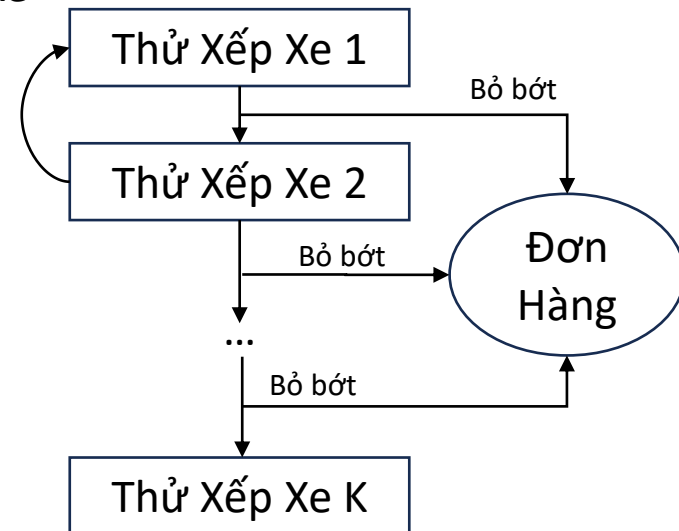
(Xét các đơn hàng theo thứ tự tăng dần của  $c_i$ )

$accum$  là tổng tải trọng tính đến hiện tại của xe  $j$

- Đã sắp xếp được toàn bộ đơn hàng, xác định bởi

$$best = \sum_{i=1}^n p_i$$

với  $best$  là giá trị hàm mục tiêu của lời giải tốt nhất tính đến hiện tại



Đạt

# Mô hình lời giải mới

Sử dụng cấu trúc Assignment để thành lập mô hình lời giải mới là một chuỗi các assignments như sau:

Assignment:

- orderID
- vehicleID

$$X = \{x_1, x_2, \dots, x_n\}$$

Ý nghĩa đơn hàng nào gán cho phương tiện nào

$x_i = j$  nghĩa là đơn hàng  $i$  được xếp vào xe  $j$

Chiều đọc để “giải mã”

Assignment	Assignment	...	Assignment
-orderID: 1	-orderID: 2		-orderID: n
-vehicleID: 1	-vehicleID: 1		-vehicleID: k

Một danh sách các Assignment, để lấy được lời giải cuối cùng: ta đọc các phần tử theo thứ tự từ đầu đến cuối của danh sách.

Thứ tự các Assignment không nhất thiết phải giống thứ tự các đơn hàng hay phương tiện

Đạt

# Hàm sửa chữa tìm kiếm cục bộ - ý tưởng cơ bản

Sau khi thu được kết quả lời giải của một số thuật toán, chúng em nhận thấy một số xe có dư đơn hàng trong khi một số lại thiếu

Do đó sử dụng hàm sửa chữa để đẩy phần dư của xe này sang cho xe khác

Ta quan tâm 2 loại phương tiện:

+ Các xe đã đủ đơn hàng: loại 1

+ Các xe chưa đủ đơn hàng: loại 2

Định nghĩa:

-  $o_j = w_j - l_j$ , là phần đang “dư” ra

-  $s_j = u_j - w_j$ , là phần còn có thể thêm vào

Hàm sửa chữa – chính là tìm kiếm 1 phần hàng xóm:

Function Repair():

For xe j loại 2:

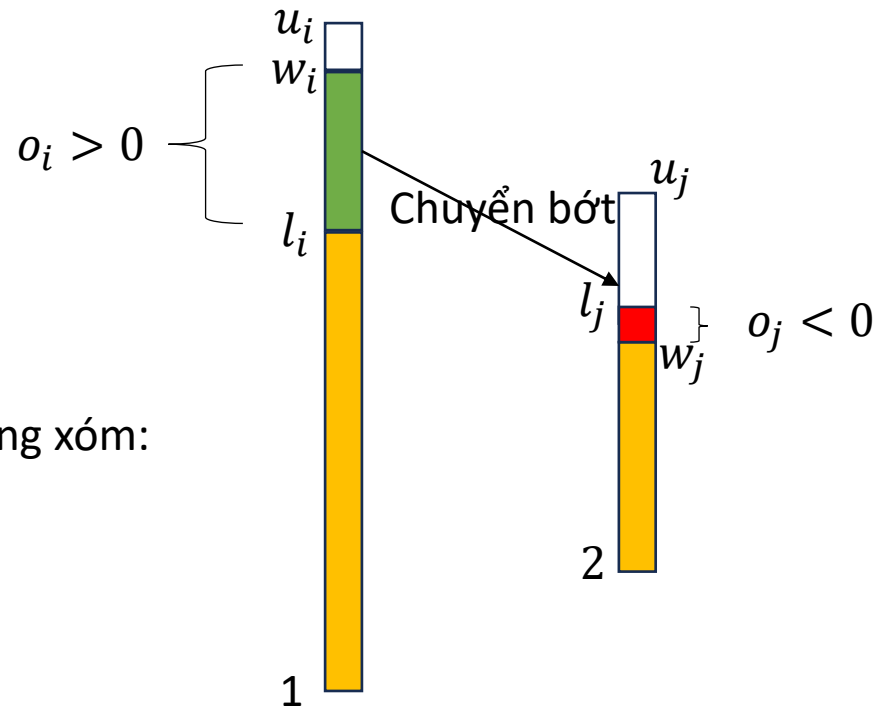
    Tính  $s_j$

For xe i loại 1:

    Tính  $o_i$

    Xét toàn bộ đơn hàng của xe có  $c < o_i$

        Nếu  $c < \min(o_i, s_j)$  thì chuyển c sang j và giảm  $o_i, s_j$  đi c



Đạt

# Thuật toán tham lam thứ ba

Sau khi thu được kết quả từ lời giải của thuật toán tham lam; ta sẽ “sửa chữa” nó bằng hàm tìm kiếm cục bộ.

Thuật toán tham lam xếp các đơn hàng vào các phương tiện theo thứ tự sau:

- Các đơn hàng:  $p_i$  giảm dần  $\rightarrow c_i$  tăng dần
- Các phương tiện:  $u_j - l_j$  tăng dần  $\rightarrow l_j$  tăng dần

Khi không thể xếp 1 đơn hàng vào 1 phương tiện nữa thì chuyển sang phương tiện khác

Greedy Insert()

Greedy Repair()

**Tất cả cố định thứ tự !**

Hàm sửa chữa được áp dụng:

Function Repair():

For xe j loại 2:

Tính  $s_j$

For xe i loại 1:

Tính  $o_i$

Xét toàn bộ đơn hàng của xe có  $c < o_i$

Nếu  $c < \min(o_i, s_j)$  thì chuyển c sang j và giảm  $o_i, s_j$  đi c

Chiều đọc  $\equiv$  chiều gán  $\rightarrow$

Assignment	Assignment
-orderId: ?	-orderId: ?
-vehicleID: ?	-vehicleID: ?

$s_j$  giảm dần  
 $\rightarrow o_j$  tăng dần

$o_i$  giảm dần  
 $\rightarrow$  số đơn hàng

# Tìm kiếm cục bộ ngẫu nhiên

Giải mã lời giải và thực hiện một số thay đổi trong đó để tìm được lời giải tốt hơn

Khởi tạo lời giải ngẫu nhiên

Chiều dọc để “giải mã”

Assignment	Assignment	...	Assignment
-orderId: 1	-orderId: 2	...	-orderId: n
- Random(1;k)	- Random(1;k)	...	- Random(1;k)

Chỉ quan tâm  $u_j$

Giai đoạn 1:

Đọc lời giải và xếp các đơn hàng vào xe tương ứng.

Nếu không xếp được thì gán vehicleID  $\leftarrow 0$

Giai đoạn 2:

Gán các đơn hàng chưa được xếp vào một xe

Giai đoạn 3:

Kích hoạt hàm sửa chữa Repair()

Danh sách các phương tiện vehicles:

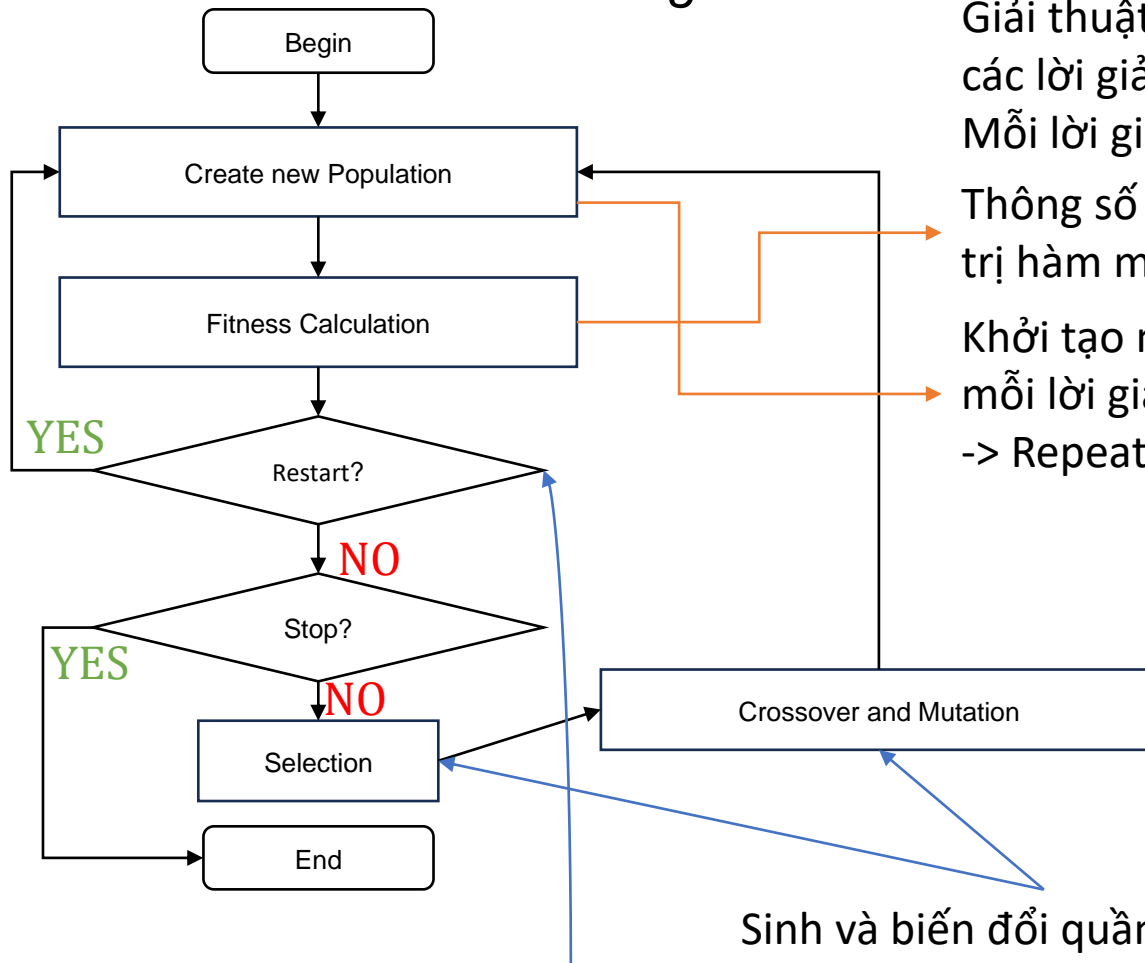
- $l_j$  và  $u_j$
- Tổng  $c_i$  đơn hiện tại
- Danh sách đơn

Ngẫu nhiên: Shuffle vehicles và assignments ở trước mỗi giai đoạn

Đạt

# Repeated Local Search và Genetic Algorithm

## Genetic Algorithm



Giải thuật di truyền xoay quanh một tập hợp các lời giải gọi là quần thể.  
Mỗi lời giải chính là 1 danh sách assignments.  
Thông số đánh giá lời giải được chọn là giá trị hàm mục tiêu  
Khởi tạo một tập hợp lời giải mới, sau đó mỗi lời giải thực hiện Local Search  
-> Repeated Local Search

Đã tìm được kết quả tối ưu toàn cục !

Sinh và biến đổi quần thể mới

Quần thể hiện tại đã đạt “giới hạn”

Đạt

# Kết quả thử nghiệm



	Jury Solution	CP		MIP		Tham lam – 1	Tham lam -2
		Objective	Time(s)	Objective	Time(s)	Objective	Objective
N = 5, K = 2	27	27	0.0472	27	0.0404	27	27
N = 10, K = 2	57	57	0.0404	57	0.0500	57	57
N = 50, K = 3	257	257	0.0440	257	0.0462	257	257
N = 100, K = 5	1065	1139	0.1050	1139	0.0526	1139	1063
N = 200, K = 10	2037	2102	0.3041	2102	1.1309		2049
N = 300, K = 10	2971	3033	0.2399	3033	1.4996	3033	3002
N = 500, K = 50	5109	5227	3.2490	5227	379.8108	5226	5193
N = 800, K = 80	8536	8559	22.0531	8559	3440.9531		8429
N = 900, K = 90	9395	9526	101.3593	9526	1060.7996		9362
N = 1000, K = 100	10368	10380	19.6820	N/A	N/A	10379	10255

Ký hiệu	Màu
Tìm được lời giải sai	
Không tìm được lời giải do giới hạn thời gian	

# Kết quả thực nghiệm

	Jury Solution	Tham lam - 3	Repeated Local Search	Feasibility "Branch&Bound"	Branch&Bound
N = 5, K = 2	27	27	27	27	27
N = 10, K = 2	57	57	57	57	57
N = 50, K = 3	257	257	257	257	257
N = 100, K = 5	1065		1139	1139	
N = 200, K = 10	2037	2102	2102		
N = 300, K = 10	2971		3033		
N = 500, K = 50	5109	5227	5227		
N = 800, K = 80	8536		8559		
N = 900, K = 90	9395		9526		
N = 1000, K = 100	10368	10380	10380		

Giới hạn thời gian: 30s  
RLS lặp lại 10000 lần, lấy BO10

Ký hiệu	Màu
Tìm được lời giải sai	
Không tìm được lời giải do giới hạn thời gian	



# Đánh giá

- Kết quả của các thuật toán khác nhau rất phụ thuộc vào sự phân phối các  $c_i$  cũng như  $[l_j, u_j]$
- Hiện tại các thuật toán đang tập trung nhiều hơn vào tính “khả thi”
  - ➔ Cần có thêm dữ liệu để giải bài toán “tối ưu”
  - ➔ Thử sinh dữ liệu và thử nghiệm lại
- + Mục tiêu của dữ liệu mới:
  - + Chắc chắn phải có đơn hàng không được xếp vào xe trong mọi trường hợp sắp xếp. Điều này có thể được thể hiện qua tổng  $\sum_{i=1}^N c_i > \text{tổng } \sum_{j=1}^K u_j$
  - + Giá trị  $u_j - l_j$  phải lớn hơn các test cases sẵn có, để tăng số lượng lời giải khả thi.

Hoặc: thử kết hợp giải thuật: Tham lam 2 và Repair()

A large, stylized graphic on the left side of the slide. It consists of a red background with a circular pattern of white dots of varying sizes, creating a sense of depth and movement. The word "HUST" is written in white, bold, sans-serif capital letters in the center of this graphic.

**HUST**

**THANK YOU !**