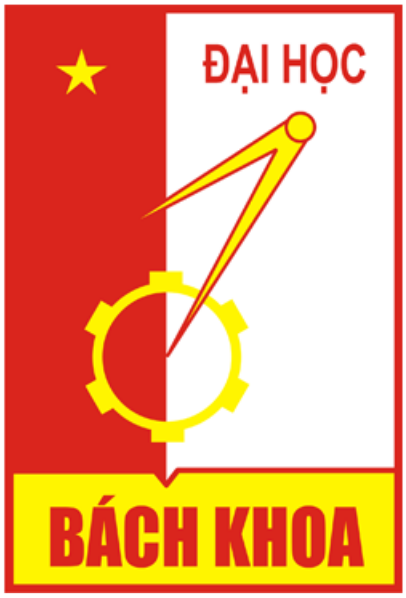

BÁO CÁO BÀI TẬP LỚN
IT3940: PROJECT III

Parallel VRPD-time window (Thuật toán Genetic Programming)

Giảng viên hướng dẫn: TS. Nguyễn Khánh Phương

Sinh viên thực hiện: Phạm Thanh An
Mã số sinh viên: 20224911

Hà Nội, tháng 2 năm 2026

Mục lục

Chương 1
Giới thiệu

1.1 Đặt vấn đề

Trong thời đại 4.0 hiện nay, sự dịch chuyển của ngành logistic trong kỷ nguyên số không chỉ dừng lại ở việc tối ưu lộ trình mà còn là khả năng thích ứng thời gian thực với các biến động của thị trường. Bài toán vận tải truyền thống vốn dựa trên các tài nguyên tĩnh đã không còn đáp ứng được tính phức tạp khi xuất hiện các nhu cầu giao hàng động (Dynamic). Trong bối cảnh đó, mô hình phối hợp song song giữa xe tải và thiết bị bay không người lái (Parallel Truck-Drone) nổi lên như một giải pháp tối ưu, tận dụng được thế mạnh về tải trọng của xe tải và tính linh hoạt về tốc độ của drone.

Tuy nhiên, thách thức lớn nhất nằm ở việc điều phối các thực thể này trong môi trường không chắc chắn, nơi các yêu cầu khách hàng xuất hiện rời rạc theo khung thời gian mà không có thông tin báo trước. Các thuật toán tối ưu hóa cổ điển thường tập trung tìm kiếm một lời giải duy nhất cho một kịch bản cố định, điều này dẫn đến sự thiếu hiệu quả khi môi trường thay đổi.

Để giải quyết vấn đề này, đề tài tiếp cận theo hướng **Genetic Programming Hyper-heuristic**. Trong cách tiếp cận này, thay vì tìm kiếm một lộ trình cụ thể, mục tiêu là tiến hóa các quy tắc điều phối (dispatching policies) để cuối cùng tìm ra tập hợp các chính quy tắc tối ưu không bị trị, đảm bảo sự cân bằng giữa khả năng phục vụ khách hàng tối đa và việc tối thiểu hóa tổng thời gian hoàn thành nhiệm vụ.

1.2 Phạm vi đề tài

Mục tiêu của bài toán là xác định lộ trình cho xe tải và drone sao cho tối thiểu hóa thời gian hoàn thành nhiệm vụ (Minimizing Makespan) cùng với đó tối đa lượng khách hàng phục vụ (Maximizing Served) với điều kiện môi trường động, các request xuất hiện ở thời điểm không xác định.

Để giải quyết bài toán phức tạp này, sinh viên đề xuất sử dụng khung giải thuật Genetic Programming. Đây là một phương pháp tìm kiếm metaheuristic mạnh mẽ, một meta-heuristic thuộc nhóm Evolutionary Algorithms, sử dụng cơ chế tiến hoá để tìm kiếm không gian chương trình mà không phụ thuộc vào bài toán cụ thể, phù hợp cho bài toán động bởi giải thuật cung cấp chiến lược ứng phó với sự xuất hiện của yêu cầu mới. Các công việc cần làm bao gồm:

- Tìm hiểu giải thuật Genetic Programming
- Đề xuất chiến lược giải quyết bài toán động (dynamic)
- Giải quyết vấn đề tối ưu đa mục tiêu
- Đánh giá kết quả trên tập dữ liệu có sẵn

Chương 2

Mô tả bài toán

2.1 Các thành phần của hệ thống

Hệ thống giao hàng bao gồm:

- Depot (kho) điểm xuất phát của tất cả phương tiện
- Phương tiện:
  - Tập hợp K truck đồng nhất với tải trọng M_T
  - Tập hợp D drone đồng nhất với tải trọng M_D và bán kính bay tối đa L_D
- Khách hàng (Requests): Tập khách hàng C được chia làm 2 loại:
  - C_1: Khách hàng bắt buộc phục vụ bởi xe tải (do kích thước hàng lớn hoặc quá xa tầm bay drone)
  - C_2: Khách hàng có thể phục vụ bởi cả drone hoặc xe tải

2.2 Đặc điểm của Request

Mỗi khách hàng i có các thuộc tính sau:

- Thời điểm xuất hiện r_i (dynamic)
- Khung thời gian phục vụ $[e_i, l_i]$ (time window)
- Khối lượng gói hàng d_i
- Thời gian đợi tối đa L_w

2.3 Ký hiệu và tham số

a, Tập hợp

- $C = \{1, 2, \ldots, n\}$: Tập hợp các khách hàng (request)
- $C_1 \subset C$: Tập khách hàng bắt buộc phải phục vụ bằng xe tải
- $C_2 \subset C$: Tập khách hàng có thể phục vụ bằng xe tải hoặc drone
- $N = C \cup \{0\}$: Tập các điểm, trong đó 0 là Depot.
- K: Tập hợp tất cả các phương tiện

  - K_T: Tập hợp Truck
  - K_D: Tập hợp Drone
- $\mathcal{R}_k$: Tập các chuyến đi của phương tiện k
- $pickup_k^r \subseteq C$: Tập các đơn hàng mà phương tiện k đang mang theo trong chuyến đi r
- $served$: Tập các đơn hàng được xử lý thành công (về kho)

b, Tham số

- d_i: Nhu cầu hàng hóa (demand) của khách i
- $[e_i, l_i]$: Khung thời gian phục vụ (Time Window) của khách hàng i
  - e_i: Thời điểm sớm nhất có thể phục vụ
  - l_i: Thời điểm muộn nhất phải phục vụ
- t_{ij}^k: Thời gian di chuyển từ i đến j của phương tiện k
- M_T: Tải trọng tối đa của xe tải
- M_D: Tải trọng tối đa của drone
- L_D: Phạm vi bay tối đa của drone cho một chuyến (Trip)
- L_w: Thời gian chờ đợi tối đa của khách hàng (từ khi đơn của họ được lấy cho đến khi phương tiện về kho)

c, Biến quyết định

- $a_i^k \in R^+$: Thời điểm phương tiện k đến phục vụ khách hàng i

2.4 Ràng buộc

- Ràng buộc về sức chứa: Tổng khối lượng hàng hóa đang mang theo trong chuyến đi không được vượt quá tải trọng tối đa của phương tiện tương ứng.

$$\sum_{i \in pickup_k^r} d_i \leq \begin{cases} M_T & \text{nếu } k \in K_T \\ M_D & \text{nếu } k \in K_D \end{cases}, \quad \forall r \in \mathcal{R}_k,\ \forall k \in K$$

- Ràng buộc về khung thời gian:

- Phương tiện k phải đến khách hàng i trước thời gian đóng cửa l_i. Nếu đến sớm hơn e_i, phương tiện phải chờ:

$$a_i^k \leq l_i, \forall i \in C, \forall k \in K$$

- Mối quan hệ giữa hai điểm liên tiếp i và j trên hành trình: Giả sử j theo sau i trong chuyến r của k, thì :

$$a_j^k = \max(a_i^k, e_i) + t_{ij}^k$$

- Ràng buộc về bán kính bay Drone:

Giả sử chuyến bay r của drone k bao gồm tập các cung đường $E_r = \{(0, i_1), (i_1, i_2), \ldots, (i_m, 0)\}$ thì:

$$\sum_{(u,v)\in E_r} t^k_{uv} \leq L_D$$

- Ràng buộc về tính tương thích: Khách hàng thuộc nhóm C_1 không được phục vụ bởi Drone.

$$pickup^r_k \cap C_1 = \emptyset, \forall k \in K_D, \forall r \in \mathcal{R}_k$$

- Ràng buộc về thời gian chờ đợi: Đối với mỗi khách hàng i $\notin$ served, thời gian từ khi đơn i được đưa vào $pickup^r_k$ (bắt đầu chờ) đến khi phương tiện k hoàn thành chuyến r (về depot) không vượt quá L_w.

$$t^k_{\text{depot}} - t^k_{\text{pickup}_i} \leq L_w, \quad \forall i \notin served,\ \forall k \in K,\ \forall r \in \mathcal{R}_k \text{ với } i \in pickup^r_k$$

Chương 3

Mô hình hóa giải pháp với GPHH

3.1 Nút lá cho biểu diễn cây GP

Thay vì giải bài toán tối ưu tĩnh, em sử dụng GPHH để tìm ra các hàm ưu tiên (Priority Functions). Hệ thống được phân rã thành hai bài toán con quyết định: Định tuyến (Routing) và Sắp xếp (Sequencing)

3.1.1 Routing Rule (Quy tắc định tuyến – cây R)

Quy tắc này quyết định phương tiện nào sẽ được giao nhiệm vụ phục vụ một yêu cầu mới khi yêu cầu xuất hiện.

Khi một yêu cầu i xuất hiện tại thời điểm hiện tại, hệ thống tính giá trị ưu tiên $R(k, i)$ cho từng phương tiện k khả dụng. Yêu cầu i sẽ được gán cho phương tiện k^* có giá trị $R(k^*, i)$ lớn nhất (ưu tiên cao nhất).

Tập terminal (các biến đầu vào) cho cây Routing:

- **RT0**: Tỷ lệ số lượng yêu cầu đang chờ trong hàng đợi của phương tiện (ít việc hơn được ưu tiên).
- **RT1**: Tỷ lệ sức chứa còn lại của phương tiện (còn nhiều chỗ trống được ưu tiên).
- **RT2**: Thời gian di chuyển từ vị trí trung tâm (median) của các yêu cầu đang chờ đến yêu cầu mới (càng gần càng tốt).
- **RT3**: Thời gian di chuyển từ vị trí hiện tại của phương tiện đến yêu cầu mới (càng gần càng tốt).
- **RT4**: Tỷ lệ kích thước gói hàng (demand) của yêu cầu mới so với tổng nhu cầu đang chờ (ưu tiên gói hàng lớn).
- **RT5**: Ưu tiên loại phương tiện (1.0 nếu là Drone, 0.0 nếu là Truck).

3.1.2 Sequencing Rule (Quy tắc sắp xếp – cây S)

Quy tắc này quyết định phương tiện đang rảnh sẽ phục vụ yêu cầu nào tiếp theo trong hàng đợi của chính nó.

Khi phương tiện k rảnh (sẵn sàng nhận nhiệm vụ mới), hệ thống tính giá trị ưu tiên $S(k, i)$ cho từng yêu cầu i đang chờ trong hàng đợi của k. Yêu cầu có giá trị $S(k, i)$ **nhỏ nhất** và thỏa mãn các ràng buộc khả thi sẽ được chọn để phục vụ tiếp theo.

Tập terminal cho cây Sequencing:

- **ST0**: Thời gian di chuyển từ vị trí hiện tại đến yêu cầu (ngắn nhất được ưu tiên).

- **ST1**: Thời gian khách hàng đã chờ kể từ khi yêu cầu xuất hiện (ưu tiên khách chờ lâu).
- **ST2**: Độ khẩn cấp – thời gian còn lại đến thời điểm đóng cửa sổ (Time until close) (ưu tiên yêu cầu sắp hết hạn).
- **ST3**: Kích thước gói hàng (demand) – càng lớn càng được ưu tiên.
- **ST4**: Độ trễ so với thời gian mở cửa sổ (thời gian sớm nhất có thể phục vụ được ưu tiên trước).
- **ST5**: Thời điểm yêu cầu xuất hiện (ưu tiên yêu cầu đến trước – FIFO cơ bản).

3.2 Nút trong cho biểu diễn cây GP

Các nút không phải lá (internal nodes) trong cây biểu diễn chương trình được xây dựng từ tập hàm sau:

add (+), **sub (-)**, **mul (*)**, **protected division (div – chia bảo toàn)**, **min**, **max**.

Lý do lựa chọn tập hàm này:

- **Các phép toán số học cơ bản (+, , ×, ÷ bảo toàn)**: Đây là tập hợp tối thiểu cần thiết để biểu diễn hầu hết các hàm ưu tiên thực tế trong bài toán dynamic truck-drone. Chúng cho phép tạo ra các biểu thức tuyến tính và phi tuyến đơn giản như tổng trọng số, chênh lệch thời gian, tỷ lệ khoảng cách/thời gian, v.v. Protected division được sử dụng để tránh lỗi chia cho 0 (trả về giá trị mặc định an toàn, thường là 1 hoặc giá trị tử số).
- **min và max**: Đây là hai hàm phi tuyến quan trọng nhất trong bài toán có ràng buộc thời gian và thứ tự ưu tiên. Chúng giúp mô hình hóa các quyết định dạng "ngưỡng" và "chọn cái tốt nhất/tồi tệ nhất" một cách tự nhiên.
- **Lý do ưu tiên tập hàm nhỏ gọn và phi tuyến đơn giản**:
  - Phù hợp với đặc trưng của bài toán **quyết định thời gian thực**: Các chính sách dispatching trong môi trường dynamic VRPDTW thường dựa trên so sánh (min/max) và kết hợp tuyến tính/tỷ lệ hơn là các hàm phức tạp như sin, exp, log.
  - Dễ diễn giải và phân tích: Tập hàm {+, , ×, ÷, min, max} sinh ra các biểu thức có thể dễ dàng chuyển thành các quy tắc if-then-else dễ hiểu cho con người – một đặc tính rất quan trọng trong hyper-heuristic.

Chương 4

Sơ đồ thuật toán và quy trình thực hiện

Thuật toán sử dụng là NSGA-II (Non-dominated Sorting Genetic Algorithm II) để tối ưu đa mục tiêu, tích hợp với Genetic Programming (GP) để tiến hóa cây quy tắc (routing rule, sequencing rule)

4.1 Các thuật toán tiến hóa

Algorithm 1 Thuật toán tiến hóa

```
Population ← CreateGreedyInitialPopulation(PopSize, MaxDepth, Problem)
Evaluate(Population, Problem)
while gen < MaxGens do
    Offspring ← []
    while len(Population) < PopSize do
        Parent1 ← NSGAIIBinaryTournamentSelection(Population, TourSize)
        Parent2 ← NSGAIIBinaryTournamentSelection(Population, TourSize)
        if random() < c_rate then
            Child1, Child2 ← SubtreeCrossover(Parent1, Parent2, MaxDepth)
        else
            Child1, Child2 ← Copy(Parent1), Copy(Parent2)
        end if
        if random() < m_rate then
            Child1 ← SubtreeMutation(Child1, MaxDepth)
        end if
        if random() < m_rate then
            Child2 ← SubtreeMutation(Child2, MaxDepth)
        end if
        Offspring ← Offspring ∪ {Child1, Child2}

        Evaluate(Offspring, Problem)
        Combined ← Population ∪ Offspring
        Population ← NSGAIISurvivalSelection(Combined, PopSize)
    end while
    return Population
```

Algorithm 2 Thuật toán khởi tạo quần thể tham lam

Require: Kích thước quần thể P, độ sâu tối đa D
Ensure: Quần thể ban đầu $\mathcal{P}$

$\mathcal{P} \leftarrow \emptyset$
$P_g \leftarrow \lfloor P/3 \rfloor$
$P_w \leftarrow \lfloor 2P/3 \rfloor$
▷ Giai đoạn 1: Cá thể mạnh được thiết kế thủ công
for mỗi cặp heuristic (R_i, S_i) trong danh sách heuristic mạnh **do**
    if $|\mathcal{P}| \geq P_g$ **then**
        break
    end if
    $T_R \leftarrow$ BuildTreeFromString(R_i)
    $T_S \leftarrow$ BuildTreeFromString(S_i)
    Thêm cá thể (T_R, T_S) vào $\mathcal{P}$
end for
▷ Giai đoạn 2: Cá thể ngẫu nhiên có trọng số
while $|\mathcal{P}| < P_w$ **do**
    $T_R \leftarrow$ GenerateWeightedRandomTree(D, R)
    $T_S \leftarrow$ GenerateWeightedRandomTree(D, S)
    Thêm cá thể (T_R, T_S) vào $\mathcal{P}$
end while
▷ Giai đoạn 3: Cá thể ngẫu nhiên hoàn toàn
while $|\mathcal{P}| < P$ **do**
    Chọn ngẫu nhiên $grow \in \{true, false\}$
    $T_R \leftarrow$ GenerateRandomTree$(D, grow, R)$
    $T_S \leftarrow$ GenerateRandomTree$(D, grow, S)$
    Thêm cá thể (T_R, T_S) vào $\mathcal{P}$
end while
return $\mathcal{P}$ =0

Algorithm 3 Thuật toán lai ghép cây con (Subtree Crossover)

Require: Hai cá thể cha mẹ P_1, P_2, độ sâu tối đa $D_{\max}$
Ensure: Hai cá thể con C_1, C_2

$C_1 \leftarrow \text{Copy}(P_1)$
$C_2 \leftarrow \text{Copy}(P_2)$
▷ Chọn ngẫu nhiên cây Routing hoặc Sequencing để lai
if random() < 0.5 **then**
    $T_1 \leftarrow C_1.R\_tree$, $T_2 \leftarrow C_2.R\_tree$
    $type \leftarrow R$
else
    $T_1 \leftarrow C_1.S\_tree$, $T_2 \leftarrow C_2.S\_tree$
    $type \leftarrow S$
end if
$n_1 \leftarrow$ số node của T_1
$n_2 \leftarrow$ số node của T_2
for $k = 1$ to 10 **do**
    $i \leftarrow \text{RandomInteger}(0, n_1 - 1)$
    $j \leftarrow \text{RandomInteger}(0, n_2 - 1)$
    $sub_1 \leftarrow$ subtree tại chỉ số i của T_1
    $sub_2 \leftarrow$ subtree tại chỉ số j của T_2
    if $\text{depth}(T_1) - \text{depth}(sub_1) + \text{depth}(sub_2) \leq D_{\max}$
and
$\text{depth}(T_2) - \text{depth}(sub_2) + \text{depth}(sub_1) \leq D_{\max}$ **then**
        if $type = R$ **then**
            Thay subtree tại i của $C_1.R\_tree$ bằng sub_2
            Thay subtree tại j của $C_2.R\_tree$ bằng sub_1
        else
            Thay subtree tại i của $C_1.S\_tree$ bằng sub_2
            Thay subtree tại j của $C_2.S\_tree$ bằng sub_1
        end if
        break
    end if
end for
return C_1, C_2

Algorithm 4 Đột biến cây con (Subtree Mutation)

Require: Cá thể Ind, độ sâu tối đa $MaxDepth$
Ensure: Cá thể mới Ind'
$Ind' \leftarrow$ **Copy**(Ind)
Chọn ngẫu nhiên cây mục tiêu $T \in \{r\_tree, s\_tree\}$
$size \leftarrow$ số lượng nút của cây T
for $i = 1$ **to** 10 **do**
    $idx \leftarrow$ chỉ số nút ngẫu nhiên trong $[0, size - 1]$
    $Sub \leftarrow$ cây ngẫu nhiên với độ sâu từ 1 đến 3
    $T' \leftarrow$ thay thế cây con tại vị trí idx của T bằng Sub
    if độ sâu của $T' \leq MaxDepth$ **then**
        cập nhật T trong Ind'
        return Ind'
    end if
end for
return Ind' ▷ không thay đổi nếu đột biến thất bại

4.2 Các thuật toán phân bổ sự kiện và điều phối xe

Algorithm 5 Giả lập sự kiện rời rạc cho đánh giá cá thể GP

Require: Problem P, Individual Ind
Ensure: f_1 (tỷ lệ phục vụ), f_2 (điểm makespan)

$P' \leftarrow \text{DeepCopy}(P)$
Reset hàng đợi và trạng thái các vehicle trong P'
$EventQueue \leftarrow \emptyset$
$PendingRequests \leftarrow \emptyset$
$curTime \leftarrow 0$
▷ Khởi tạo sự kiện ARRIVE và END
for each request $r \in P.requests$ **do**
    Push $(r.release\_time, \text{“ARRIVE”}, r.id)$ vào $EventQueue$
end for
Push $(P.depotClose, \text{“END”},)$ vào $EventQueue$
while $EventQueue$ không rỗng **do**
    $(time, type, payload) \leftarrow$ Pop sự kiện sớm nhất
    $curTime \leftarrow time$
    if $type =$ “END” **then**
        break
    end if
    if $type =$ “ARRIVE” **then**
        $req \leftarrow$ bản sao request theo $payload$
        $assigned \leftarrow \textsc{TryAssign}(req, P'.vehicles, Ind.r\_tree)$
        if not $assigned$ **then**
            Thêm req vào $PendingRequests$
        end if
    end if
    if $type =$ “VEH_FREE” **then**
        $veh \leftarrow$ vehicle theo $payload$
        for each $r \in PendingRequests$ **do**
            if r còn hợp lệ về time window **then**
                if $\textsc{TryAssign}(r, P'.vehicles, Ind.r\_tree)$ **then**
                    Xóa r khỏi $PendingRequests$
                end if
            end if
        end for
        $\textsc{DispatchVehicle}(veh, Ind.s\_tree, curTime)$
    end if
end while
▷ Tính toán hàm mục tiêu
$served \leftarrow$ số request được phục vụ
$total \leftarrow$ tổng request
$makespan \leftarrow \max(v.busy\_until)$
$f_1 \leftarrow served/total$
$f_2 \leftarrow \max(0, 1 - makespan/P.depotClose)$
return $(f_1, f_2, served, makespan)$

Algorithm 6 Gán request cho xe sử dụng R-tree

Require: Request r, danh sách xe V, R-tree, thời gian hiện tại t, số lượng gán tối đa $Assign_n$

Ensure: Request r được gán cho tối đa $Assign_n$ xe (nếu thỏa điều kiện)

$Candidates \leftarrow \emptyset$
for each xe $v \in V$ **do**
    if v là DRONE **và** r không cho phép drone **then**
        continue
    end if
    if Tổng nhu cầu hàng đợi của $v + r.demand > v.capacity$ **then**
        continue
    end if
    if v là DRONE **và** không thể bay tới vị trí của r **then**
        continue
    end if
    $score \leftarrow$ R-tree.evaluate(v, r, t)
    $Candidates \leftarrow Candidates \cup \{(score, v)\}$
end for
if $Candidates = \emptyset$ **then**
    return False ▷ Không có xe thỏa điều kiện
end if
Sắp xếp $Candidates$ theo $score$ giảm dần
$assigned \leftarrow 0$
$assigned\_any \leftarrow$ False
for each $(score, v) \in Candidates$ **do**
    if r đã được phục vụ **then**
        break
    end if
    $t_{start} \leftarrow \max(t, v.busy\_until)$
    $t_{arrive} \leftarrow t_{start} + v.moving\_time(r.location)$
    if $t_{arrive} > r.time\_window_{end}$ **then**
        continue
    end if
    Gán r vào hàng đợi của v
    $assigned\_any \leftarrow$ True
    if $v.busy\_until \leq t$ **then**
        Điều phối xe v thực hiện request
    end if
    $assigned \leftarrow assigned + 1$
    if $assigned \geq Assign_n$ **then**
        break
    end if
end for
return $assigned\_any$

Algorithm 7 Điều phối xe

Require: Xe v, thời gian hiện tại t, S-tree
Ensure: Thực hiện hành động tiếp theo cho xe v

Loại bỏ các request đã được phục vụ hoặc đã được lấy khỏi hàng đợi của v
$ready\_time \leftarrow \max(t, v.busy\_until)$
$depot\_close \leftarrow$ thời điểm đóng depot
if $v.picked\_up\_orders \neq \emptyset$ **then**
    $t_{return} \leftarrow ready\_time + time(v.current\_location \rightarrow depot)$
    for each đơn $p \in v.picked\_up\_orders$ **do**
        if $t_{return} - p.pickup\_time > p.l_w$ **then**
            Thực hiện chuỗi trả hàng thất bại (vi phạm l_w)
            return
        end if
    end for
end if
$Candidates \leftarrow \emptyset$
for each request $r \in v.req\_queue$ **do**
    if r đã được lấy hoặc đã phục vụ **then**
        continue
    end if
    $t_{arrive} \leftarrow ready\_time + time(v \rightarrow r)$
    $t_{service} \leftarrow \max(t_{arrive}, r.time\_window_{start})$
    if $t_{arrive} > r.time\_window_{end}$ **then**
        continue
    end if
    $t_{depot} \leftarrow t_{service} + time(r \rightarrow depot)$
    if $t_{depot} > depot\_close$ **then**
        continue
    end if
    if bất kỳ đơn $p \in v.picked\_up\_orders$ vi phạm l_w khi về depot **then**
        continue
    end if
    if $t_{depot} - t_{service} > r.l_w$ **then**
        continue
    end if
    if v là DRONE và vượt quá tầm bay cho phép **then**
        continue
    end if
    $score \leftarrow$ S-tree.evaluate(v, r, t)
    $Candidates \leftarrow Candidates \cup \{(score, r)\}$
end for
if $Candidates \neq \emptyset$ **then**
    Chọn $r^* = \arg\min score$
    if $v.remaining\_capacity \geq r^*.demand$ **then**
        Thực hiện lấy hàng r^* và cập nhật trạng thái xe
        return
    end if
end if
if $v.current\_location \neq depot$ **then**
    Điều phối xe v quay về depot
end if

Algorithm 8 Thực hiện lấy đơn hàng

Require: Xe v, request r, $ready\_time$, $travel\_time$, $service\_start$
Ensure: Cập nhật trạng thái xe và request sau khi lấy hàng

$arrival\_time \leftarrow ready\_time + travel\_time$
▷ Khởi tạo tuyến mới nếu xe xuất phát từ depot
if $v.current\_location = depot$ **then**
    Thêm tuyến rỗng mới vào $v.routes$
end if
▷ Cập nhật tầm bay nếu là DRONE
if $v.type = DRONE$ **then**
    $v.remaining\_range \leftarrow v.remaining\_range - travel\_time$
end if
▷ Cập nhật trạng thái request
$r.is\_picked\_up \leftarrow$ True
$r.pickup\_time \leftarrow service\_start$
▷ Cập nhật trạng thái xe
$v.remaining\_capacity \leftarrow v.remaining\_capacity - r.demand$
Thêm r vào danh sách $v.picked\_up\_orders$
$v.current\_location \leftarrow r.location$
$v.busy\_until \leftarrow service\_start$
Loại bỏ r khỏi hàng đợi $v.req\_queue$
▷ Lập lịch sự kiện xe rảnh
Đưa sự kiện $(v.busy\_until, VEH\_FREE, v.id)$ vào hàng đợi sự kiện

Algorithm 9 Quay về depot và hoàn tất đơn hàng

Require: Xe v, thời điểm sẵn sàng $ready\_time$
Ensure: Hoàn tất các đơn đang chở và cập nhật trạng thái xe

▷ Tính thời gian quay về depot
$travel\_time \leftarrow$ Thời gian di chuyển từ $v.current\_location$ về depot
$arrival\_time \leftarrow ready\_time + travel\_time$
▷ Ghi nhận sự kiện quay về depot
if Logging được bật **then**
    Ghi log sự kiện `RETURN_DEPOT`
end if
▷ Sạc lại drone nếu cần
if $v.type = DRONE$ **then**
    Thực hiện sạc lại pin cho v
end if
▷ Hoàn tất tất cả đơn hàng đang chở
for mỗi request r trong $v.picked\_up\_orders$ **do**
    $r.is\_served \leftarrow$ True
end for
▷ Reset trạng thái xe tại depot
$v.picked\_up\_orders \leftarrow \emptyset$
$v.busy\_until \leftarrow arrival\_time$
$v.current\_location \leftarrow depot$
$v.remaining\_capacity \leftarrow v.capacity$
▷ Lập lịch sự kiện xe rảnh
Thêm sự kiện $(v.busy\_until, VEH\_FREE, v.id)$ vào hàng đợi sự kiện

Algorithm 10 Trả hàng bắt buộc (Failed Return)

Require: Xe v, request khẩn cấp r, thời điểm sẵn sàng $ready\_time$, ghi chú $note$
Ensure: Hoàn tác việc lấy hàng và cập nhật trạng thái hệ thống

▷ Tính thời gian quay lại khách hàng
$travel\_time \leftarrow$ Thời gian di chuyển từ $v.current\_location$ đến $r.location$
$arrival\_time \leftarrow ready\_time + travel\_time$
▷ Hoàn tác trạng thái request
$r.is\_picked\_up \leftarrow$ False
$r.pickup\_time \leftarrow$ None
▷ Nếu vẫn còn trong time window thì đưa lại vào hàng chờ
if $arrival\_time \leq r.time\_window^{end}$ **then**
    Thêm r vào tập request chờ xử lý
end if
▷ Cập nhật trạng thái xe
$v.current\_location \leftarrow r.location$
$v.busy\_until \leftarrow arrival\_time$
$v.remaining\_capacity \leftarrow v.remaining\_capacity + r.demand$
Loại bỏ r khỏi danh sách đơn đang chở
▷ Lập lịch sự kiện xe rảnh
Thêm sự kiện $(v.busy\_until, VEH\_FREE, v.id)$ vào hàng đợi sự kiện

Chương 5

Thực nghiệm và kết quả

5.1 Tham số thực nghiệm Genetic Programming

Để đánh giá hiệu quả của thuật toán Genetic Programming, em thực hiện thí nghiệm trên nhiều bộ tham số khác nhau. Bảng 5.1 dưới đây tóm tắt 7 bộ cấu hình được sử dụng cho 7 loại bộ dữ liệu.

Bảng 5.1: Các bộ tham số cấu hình Genetic Programming

Tham số	**config_6**	**config_10**	**config_12**	**config_20**	**config_50**	**config_100**
pop_size	50	50	50	50	60	80
max_gen	80	80	80	80	80	80
max_depth	5	5	5	5	5	5
c_rate	0.8	0.8	0.8	0.8	0.8	0.8
m_rate	0.2	0.2	0.2	0.2	0.2	0.2
tourn_size	4	4	4	4	4	4
Random seed	42	42	42	42	42	42
assignment_n	1	1	1	2	3	4

5.2 Kết quả thực nghiệm trên các instance

Bảng dưới đây tổng hợp kết quả chạy thực nghiệm với thuật toán GPHH + NSGA-II trên các bộ dữ liệu khác nhau. Mỗi instance có số lượng cá thể không trội trong quần thể cuối cùng (Pareto Count), số lượng phục vụ (Served), makespan và thời gian thực thi trung bình (Avg Execution Time – đơn vị giây).

Bảng 5.2: Kết quả thực nghiệm trên các instance Parallel VRPD-time window

Instance	**Pareto Count**	**Served**	**Makespan**	**Avg Execution Time (s)**
6.5.1	50	6	753.5815	2.477
6.5.2	50	5	546.882	2.7739
6.5.3	50	5	687.2092	2.6947
6.5.4	50	4	611.8268	2.5305
6.10.1	50	5	1058.9239	2.1617
6.10.2	50	5	1169.0393	2.4986
6.10.3	50	5	1162.8643	2.9062

Instance	Pareto Count	Served	Makespan	Avg Execution Time (s)
6.10.4	50	4	997.8432	2.9831
6.20.1	50	5	4306.6813	2.5509
6.20.2	50	6	4202.1827	2.2895
6.20.3	50	5	6233.2145	2.2328
6.20.4	50	5	6174.7861	2.2906
10.5.1	50	8	979.9087	3.6434
10.5.2	50	8	709.2037	3.7021
10.5.3	50	7	775.6905	3.5072
10.5.4	50	9	753.1453	3.3137
10.10.1	50	8	1901.1013	3.5888
10.10.2	50	10	2060.5354	3.5261
10.10.3	50	5	1508.8685	3.4069
10.10.4	50	7	1373.2675	3.6308
10.20.1	50	6	5738.0668	3.325
10.20.2	50	6	6203.1798	3.298
10.20.3	50	5	6468.8684	3.089
10.20.4	50	5	5015.2341	3.3396
12.5.1	50	10	877.8337	4.5616
12.5.2	50	10	755.2642	4.4439
12.5.3	50	11	1128.751	4.0148
12.5.4	50	8	974.2947	4.5237
12.10.1	50	10	1725.2414	4.2566
12.10.2	50	12	1731.7758	4.3261
12.10.3	50	11	1901.2105	4.6493
12.10.4	50	8	1664.698	4.3017
12.20.1	50	8	6705.1533	4.1253
12.20.2	50	6	6365.2014	3.7361
12.20.3	50	8	4945.0698	4.5414
12.20.4	50	6	6009.3265	4.2781
20.5.1	50	15	711.7336	8.5715
20.5.2	50	15	801.0408	8.78
20.5.3	50	15	576.1912	10.5179
20.5.4	50	18	629.165	8.885
20.10.1	50	16	1436.3767	9.1622
20.10.2	50	15	1325.8383	12.7183
20.10.3	50	13	1202.4461	9.9026
20.10.4	50	16	1360.8	7.6799
20.20.1	50	14	4420.6347	6.7886
20.20.2	50	18	4213.7226	7.4526
20.20.3	50	11	5491.3251	6.9105
20.20.4	50	12	2440.0265	7.05
50.10.1	60	40	1979.7255	54.3897

Instance	Pareto Count	Served	Makespan	Avg Execution Time (s)
50.10.2	60	34	1713.3307	32.8125
50.10.3	60	37	1665.6566	50.2701
50.10.4	60	37	1551.665	34.2889
50.20.1	60	30	3919.3287	36.0902
50.20.2	60	26	4111.6578	33.6495
50.20.3	60	25	4404.5782	31.5926
50.20.4	60	19	4941.4782	26.4583
50.30.1	60	19	9010.177	23.0868
50.30.2	60	21	7987.9054	23.5384
50.30.3	60	21	8783.3467	20.6473
50.30.4	60	16	8208.8034	19.4392
50.40.1	60	24	14160.1392	22.6226
50.40.2	60	20	14480.6177	22.5409
50.40.3	60	20	13166.9166	22.5623
50.40.4	60	19	13180.8472	26.6073
100.10.1	80	65	1776.4963	284.4609
100.10.2	80	84	1798.1246	150.6617
100.10.3	80	78	1761.5812	184.7923
100.10.4	80	71	1718.58	154.266
100.20.1	80	34	4451.3342	102.6948
100.20.2	80	51	3989.7156	150.4607
100.20.3	80	32	5037.5123	105.2364
100.20.4	80	43	4568.0367	133.4885
100.30.1	80	28	8554.7242	73.6698
100.30.2	80	40	8721.025	87.2097
100.30.3	80	32	8726.862	78.9565
100.30.4	80	30	8784.8097	78.9406
100.40.1	80	33	14982.3807	76.2431
100.40.2	80	36	15066.4212	80.8628
100.40.3	80	51	15361.3954	87.1002
100.40.4	80	34	14617.2005	77.119

Chương 6
Kết luận

Qua quá trình thực hiện đồ án Project III với đề tài "Parallel VRPD - time window" sử dụng giải thuật Genetic Programming, sinh viên đã có cơ hội tiếp cận sâu sắc với bài toán lập lịch vận tải động trong môi trường không chắc chắn, kết hợp giữa xe tải và drone–một hướng nghiên cứu đang rất được quan tâm trong lĩnh vực logistics hiện đại.

Các kết quả và đóng góp chính đạt được bao gồm:

- Xây dựng mô hình toán học hoàn chỉnh cho bài toán Parallel Vehicle Routing Problem with Time Windows với các ràng buộc phức tạp.
- Đề xuất và triển khai thành công khung giải pháp dựa trên **Genetic Programming Hyper-heuristic (GPHH)** kết hợp NSGA-II, trong đó:
  - Tiến hóa đồng thời hai cây quy tắc riêng biệt:
  - **Routing Rule (cây R)** – quyết định phương tiện nào được phân bổ để phục vụ yêu cầu mới xuất hiện.
  - **Sequencing Rule (cây S)** – quyết định thứ tự phục vụ các yêu cầu đang chờ trong hàng đợi của từng phương tiện.

  Tập terminal được thiết kế phong phú, phản ánh các yếu tố thực tế quan trọng. Tập hàm được chọn gọn nhẹ nhưng mạnh mẽ nhằm đảm bảo tính diễn giải cao và khả năng ứng dụng trong môi trường thời gian thực.
- Tích hợp NSGA-II để giải quyết bài toán tối ưu đa mục tiêu, cơ chế giả lập sự kiện rời rạc.
- Thực hiện thực nghiệm trên các bộ dữ liệu với quy mô từ 10 đến 100 yêu cầu, thử nghiệm nhiều cấu hình tham số (pop_size, assignment_n, ...), và thu được tập Pareto front chất lượng, chứng minh khả năng cân bằng hiệu quả giữa hai mục tiêu chính trong môi trường động.

Những kết quả đạt được không chỉ khẳng định tính khả thi của cách tiếp cận GPHH trong bài toán VRPD động mà còn mở ra tiềm năng ứng dụng thực tế trong lĩnh vực giao hàng nhanh và logistics đô thị.

Trong tương lai, em dự định tiếp tục phát triển theo các hướng sau:

- Mở rộng tập terminal và function set để tăng khả năng biểu diễn của các quy tắc tiến hóa.
- Thử nghiệm kết hợp chuyển giao kiến thức giữa các instance bài toán khác nhau.
- Thêm các ràng buộc thực tế hơn như ảnh hưởng của thời tiết đến tốc độ drone, nhiều depot, hoặc tải trọng thay đổi theo tuyến đường.

- Tối ưu hóa hiệu suất tính toán (song song hóa đánh giá cá thể, giảm số lần giả lập) để áp dụng cho quy mô lớn hơn.
- Khai thác khả năng ứng dụng thực tế trong hệ thống giao hàng thông minh tại Việt Nam.

Cuối cùng, em xin cảm ơn TS. Nguyễn Khánh Phương và cô Trần Thị Huế đã tận tình hướng dẫn, định hướng và hỗ trợ em trong suốt quá trình thực hiện đồ án ạ!