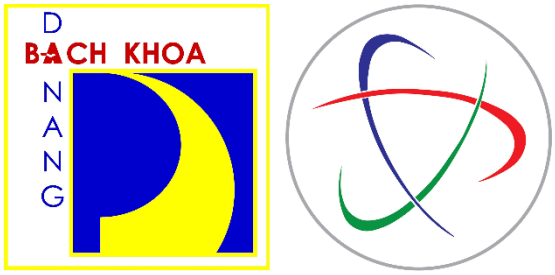
ĐẠI HỌC ĐÀ NẴNG

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA ĐIỆN TỬ VIỄN THÔNG



BÁO CÁO CUỐI KỲ

MÔN: CHUYÊN ĐỀ 2

Đề tài: Nghiên cứu và triển khai bài toán gán vị trí lưu trữ và định tuyến người lấy hàng (SLAPRP) trong kho hàng

GVHD : TS. Nguyễn Văn Hiếu

SVTH : Trần Văn Sáng - 106210051 - 21KTMT

Võ Văn Tín - 106210254 - 21KTMT2

ĐÀ NẴNG, 2025

LỜI NÓI ĐẦU

Trong các hệ thống kho hàng hiện đại, đặc biệt là kho phục vụ thương mại điện tử, hoạt động lấy hàng theo đơn đóng vai trò quan trọng và chiếm tỷ trọng lớn trong chi phí vận hành. Hiệu quả của quá trình này phụ thuộc trực tiếp vào cách gán vị trí lưu trữ cho hàng hóa và lộ trình di chuyển của người lấy hàng.

Trong thực tế, hai bài toán gán vị trí lưu trữ và định tuyến người lấy hàng thường được giải quyết riêng rẽ hoặc dựa trên các phương pháp heuristic, dẫn đến hiệu quả tối ưu chưa cao khi xét trên toàn hệ thống. Do đó, việc nghiên cứu một cách tiếp cận tích hợp để giải đồng thời hai bài toán này là cần thiết.

Xuất phát từ yêu cầu đó, đề tài "Nghiên cứu và triển khai bài toán gán vị trí lưu trữ và định tuyến người lấy hàng (SLAPRP) trong kho hàng" được thực hiện nhằm phân tích mô hình toán học của bài toán SLAPRP và áp dụng thuật toán Branch-Cut-and-Price để giải quyết bài toán một cách tối ưu. Nội dung báo cáo tập trung vào mô hình hóa, phương pháp giải và đánh giá kết quả của thuật toán thông qua triển khai thực nghiệm.

MỤC LỤC

Chương 1: Phân công công việc trong nhóm

STT	Họ tên	Nhiệm vụ	Tỷ lệ
1	Trần Văn Sáng	Nghiên cứu cơ sở lý thuyết của bài toán gán vị trí lưu trữ và định tuyến người lấy hàng. Phân tích thuật toán Branch-Cut-and-Price. Tham gia xây dựng và hiệu chỉnh mô hình toán học của bài toán. Thực hiện cài đặt, chạy thử nghiệm thuật toán và phân tích kết quả. Viết và chỉnh sửa các phần mô hình hóa, phương pháp giải và đánh giá kết quả trong báo cáo.	50%
2	Võ Văn Tín	Tham gia phân tích bài toán SLAPRP và tìm kiếm tài liệu liên quan. Xây dựng mô hình toán học và nghiên cứu phương pháp giải. Triển khai thuật toán và đánh giá hiệu quả thuật toán. Cài đặt và thực nghiệm thuật toán với một số bộ dữ liệu. Viết và chỉnh sửa các phần phương pháp giải, độ phức tạp và thực nghiệm trong báo cáo.	50%

Chương 2: Mô tả nội dung chính của dự án

2.1 Yêu cầu bài toán và tính cấp thiết

Trong các hệ thống kho hàng hiện đại, đặc biệt là kho phục vụ thương mại điện tử (e-commerce), hoạt động lấy hàng theo đơn chiếm tỷ trọng lớn nhất trong chi phí vận hành và thời gian xử lý đơn hàng. Hai quyết định quan trọng ảnh hưởng trực tiếp đến hiệu quả của hoạt động này là:

- Gán vị trí lưu trữ (Storage Location Assignment): quyết định mỗi đơn hàng được đặt tại vị trí nào trong kho.
- Định tuyến người lấy hàng (Picker Routing): quyết định lộ trình di chuyển của người lấy hàng để thu thập nhiều loại hàng trong một đơn.

Trong thực tế, hai bài toán này thường được giải tách rời:

- Việc gán vị trí lưu trữ được thực hiện trước, mang tính chiến thuật.
- Việc định tuyến người lấy hàng được thực hiện sau, mang tính tác nghiệp.

Cách tiếp cận này không đảm bảo tối ưu toàn cục, đặc biệt trong bối cảnh kho e-commerce, nơi:

- Nhu cầu thay đổi liên tục theo ngày/giờ.
- Hàng hóa được bổ sung và sắp xếp lại nhiều lần trong ngày.
- Mỗi đợt lấy hàng thường xử lý các batch đơn đã biết trước.

Do đó, bài toán tích hợp gán vị trí lưu trữ và định tuyến người lấy hàng (SLAPRP) trở nên cấp thiết, với mục tiêu tối thiểu hóa tổng quãng đường di chuyển của người lấy hàng, từ đó:

- Giảm thời gian xử lý đơn
- Giảm chi phí vận hành
- Tăng hiệu suất kho hang

2.2 Mô hình hệ thống và cách thức hoạt động

2.2.1 Mô hình hệ thống kho hàng

Hệ thống kho hàng được mô hình hóa như một không gian lưu trữ gồm các thành phần sau:

- Khu vực lưu trữ được chia thành các vị trí lưu trữ, mỗi vị trí có dung lượng hữu hạn và có thể chứa một hoặc nhiều SKU (Stock Keeping Unit).
- Tập SKU đại diện cho các mặt hàng cần được lưu trữ và lấy theo đơn.
- Tập đơn hàng (hoặc batch đơn hàng), trong đó mỗi đơn yêu cầu lấy một tập SKU xác định.

- Điểm xuất phát/kết thúc (drop-off point), nơi người lấy hàng bắt đầu và kết thúc lộ trình.

Hệ thống được mô hình hóa dưới dạng đồ thị có hướng, trong đó:

- Các đỉnh biểu diễn các vị trí lưu trữ và điểm drop-off.
- Các cạnh biểu diễn khả năng di chuyển giữa các vị trí trong kho.
- Chi phí di chuyển giữa các đỉnh phụ thuộc vào bố cục kho và chính sách định tuyến được áp dụng.

Mô hình đồ thị này cho phép mô tả linh hoạt nhiều cấu trúc kho khác nhau, từ kho một block, nhiều block đến các cấu hình kho phức tạp trong thực tế.

2.2.2 Các quyết định trong hệ thống

Hệ thống cần đồng thời đưa ra hai nhóm quyết định chính:

Quyết định gán vị trí lưu trữ:

- Mỗi SKU được gán vào đúng một vị trí lưu trữ.
- Tổng số SKU tại mỗi vị trí không vượt quá dung lượng cho phép.
- Một số SKU có thể bị ràng buộc phải nằm ở các vị trí cố định, ví dụ do đặc tính hàng hóa hoặc yêu cầu vận hành.

Quyết định định tuyến người lấy hàng:

- Với mỗi đơn hàng, hệ thống xác định một lộ trình di chuyển hợp lệ.
- Lộ trình phải đi qua tất cả các vị trí chứa SKU của đơn hàng.
- Lộ trình bắt đầu và kết thúc tại điểm drop-off.
- Mỗi lộ trình có chi phí tương ứng với tổng quãng đường di chuyển.

2.2.3 Cách thức hoạt động của hệ thống

Quá trình hoạt động của hệ thống được mô tả theo các bước sau:

❖ Nhận dữ liệu đầu vào

Hệ thống tiếp nhận các thông tin gồm:

- Danh sách SKU cần lưu trữ.
- Dung lượng các vị trí lưu trữ.
- Danh sách đơn hàng (đã được batch).
- Các ràng buộc vị trí cố định (nếu có).

❖ Xây dựng mô hình tối ưu

Từ dữ liệu đầu vào, hệ thống xây dựng mô hình toán học tích hợp, trong đó:

- Các biến quyết định mô tả việc gán SKU vào các vị trí lưu trữ.
- Các biến quyết định mô tả việc lựa chọn lộ trình lấy hàng cho từng đơn.

❖ Giải bài toán tối ưu tích hợp

Thay vì giải tuần tự, hệ thống giải đồng thời hai bài toán gán vị trí lưu trữ và định tuyến người lấy hàng, nhằm đảm bảo tối ưu toàn cục cho tổng quãng đường di chuyển.

❖ Xuất kết quả

Kết quả đầu ra của hệ thống bao gồm:

- Phương án gán vị trí lưu trữ cho từng SKU.
- Lộ trình lấy hàng tối ưu cho mỗi đơn.
- Tổng chi phí di chuyển tương ứng.

2.3 Cách tiếp cận và điểm mới của dự án

2.3.1 Cách tiếp cận của dự án

Dự án tiếp cận bài toán SLAPRP theo hướng tối ưu chính xác, thay vì sử dụng các phương pháp heuristic, với các đặc điểm chính sau:

- Tích hợp hoàn toàn hai bài toán SLAP và PRP trong cùng một mô hình tối ưu.
- Sử dụng mô hình toán học mở rộng để tách phần định tuyến ra khỏi mô hình gán vị trí.
- Áp dụng phân rã Dantzig–Wolfe, trong đó:
    - Mỗi lộ trình lấy hàng hợp lệ được xem như một "cột" trong mô hình.
    - Không cần liệt kê toàn bộ các lộ trình khả thi ngay từ đầu.

Để giải mô hình mở rộng này, dự án sử dụng thuật toán Branch-Cut-and-Price, kết hợp ba kỹ thuật:

- Branch-and-Bound để đảm bảo nghiệm nguyên.
- Column Generation để sinh động các lộ trình cần thiết.
- Cutting Planes để tăng cường chất lượng nghiệm của bài toán LP.

2.3.2 Giải bài toán định tuyến bằng Column Generation

Trong khuôn khổ column generation:

Bài toán chính quyết định:

- Gán SKU vào vị trí lưu trữ.
- Lựa chọn một lộ trình lấy hàng cho mỗi đơn.

Bài toán con có nhiệm vụ:

- Tìm các lộ trình lấy hàng có chi phí giảm (reduced cost âm).
- Bài toán con được mô hình hóa dưới dạng Elementary Shortest Path Problem with Resource Constraints (ESPPRC).

Cách tiếp cận này giúp:

- Tránh hiện tượng bùng nổ số biến trong mô hình.
- Chỉ sinh ra các lộ trình thực sự cần thiết cho nghiệm tối ưu.

2.3.3 Điểm mới của dự án

Các điểm mới và nổi bật của dự án bao gồm:

- Tối ưu tích hợp thay vì tuần tự: Giải quyết đồng thời việc gán vị trí lưu trữ và định tuyến người lấy hàng, tránh nghiệm cục bộ.
- Định tuyến tối ưu chính xác: Không sử dụng các chính sách định tuyến heuristic phổ biến, mà giải bài toán định tuyến ở mức tối ưu.
- Áp dụng Branch-Cut-and-Price cho SLAPRP: Một phương pháp mạnh, phù hợp để giải các bài toán NP-hard có cấu trúc lớn.
- Mô hình linh hoạt và dễ mở rộng: Cho phép thay đổi cấu trúc kho, chính sách định tuyến và các ràng buộc bổ sung mà không cần thay đổi toàn bộ mô hình.
- Gần với bối cảnh kho e-commerce thực tế: Việc giả định các đơn hàng đã được batch trước và quyết định lưu trữ ở mức tác nghiệp phản ánh sát hoạt động của các kho hiện đại.

2.4 Phạm vi triển khai trong dự án

Trong khuôn khổ dự án, nhóm tập trung vào:

- Nghiên cứu mô hình toán học của bài toán SLAPRP.
- Triển khai và sử dụng mã nguồn Julia để giải bài toán bằng thuật toán Branch-Cut-and-Price.
- Thực nghiệm trên các bộ dữ liệu mô phỏng kho hàng để đánh giá hiệu quả của mô hình và thuật toán.

Chương 3: Mô tả và giải quyết bài toán

3.1 Mô hình hóa bài toán dưới dạng công thức toán học

Bài toán gán vị trí lưu trữ và định tuyến người lấy hàng (Storage Location Assignment and Picker Routing Problem – SLAPRP) được mô hình hóa như một bài toán tối ưu tổ hợp tích hợp, trong đó các quyết định về lưu trữ và định tuyến được đưa ra đồng thời nhằm tối thiểu hóa tổng chi phí di chuyển trong kho.

❖ Mô hình kho và dữ liệu đầu vào

Kho hàng được biểu diễn bằng một đồ thị có hướng

$$G = (V_0, E)$$

trong đó:

- v_0 là điểm drop-off, nơi người lấy hàng bắt đầu và kết thúc lộ trình.
- V là tập các nút lưu trữ, mỗi nút tương ứng với một vị trí có thể chứa một SKU.
- $V_0 = V \cup \{v_0\}$.
- E là tập các cung biểu diễn khả năng di chuyển giữa các nút trong kho.

Các nút lưu trữ được nhóm thành các vị trí lưu trữ. Ký hiệu:

- L: tập các location.
- $V(l) \subseteq V$: tập các nút thuộc location l.
- $K_l = \mid V(l) \mid$: dung lượng của location l.

Tập dữ liệu của bài toán gồm:

- S: tập các SKU, với $\mid S \mid \leq \mid V \mid$.
- O: tập các đơn hàng.
- $S(o) \subseteq S$: tập SKU cần lấy trong đơn hàng o.
- $F \subseteq S \times L$: tập các cặp gán cố định, trong đó $(s, l) \in F$ nghĩa là SKU s bắt buộc phải được gán vào location l.

Một lộ trình lấy hàng (route) cho đơn o là một tour bắt đầu và kết thúc tại v_0, đi qua các nút lưu trữ chứa SKU thuộc $S(o)$. Chi phí của lộ trình được ký hiệu là $f(x^o)$, phụ thuộc vào chính sách định tuyến và bố cục kho.

❖ Biến quyết định

Bài toán sử dụng các biến quyết định sau:

- Biến gán vị trí lưu trữ

$$\xi_{ls} = \begin{cases} 1 & \text{nếu SKU } s \text{ được gán vào location } l \\ 0 & \text{ngược lại} \end{cases}$$

- Biến định tuyến (compact formulation)

$$x_{ij}^{o} = \begin{cases} 1 & \text{nếu lộ trình của đơn } o \text{ đi qua cung } (v_i, v_j) \\ 0 & \text{ngược lại} \end{cases}$$

- Biến phụ trợ loại subtour (MTZ)

$$u_i^o \geq 0$$

biểu diễn thứ tự ghé thăm nút v_i trong lộ trình của đơn o.

❖ Hàm mục tiêu

Mục tiêu của bài toán là tối thiểu hóa tổng chi phí di chuyển của tất cả các đơn hàng:

$$\min \sum_{o \in O} f(x^o) \qquad (1)$$

❖ Các ràng buộc gán vị trí lưu trữ

- Dung lượng location:

$$\sum_{s \in S} \xi_{ls} \leq K_l \qquad \forall l \in L \qquad (2)$$

- Mỗi SKU gán đúng một location:

$$\sum_{l \in L} \xi_{ls} = 1 \qquad \forall s \in S \qquad (3)$$

- Gán vị trí cố định:

$$\xi_{ls} = 1 \qquad \forall\,(s,l) \in F \qquad (4)$$

❖ Các ràng buộc định tuyến cho mỗi đơn hàng

- Bắt đầu và kết thúc tại drop-off:

$$\sum_{v_j \in V} x_{0j}^{o} = 1, \qquad \sum_{v_i \in V} x_{i0}^{o} = 1 \quad \forall o \in O \qquad (5)$$

- Bảo toàn luồng tại mỗi nút lưu trữ:

$$\sum_{v_j \in V_0} x_{ij}^{o} = \sum_{v_j \in V_0} x_{ji}^{o} \qquad \forall v_i \in V, \forall o \in O \qquad (6)$$

- Loại bỏ subtour (MTZ):

$$u_j^o \geq u_i^o + 1 - |\,S(o)\,|\,(1 - x_{ij}^o) \qquad \forall v_i, v_j \in V, \quad \forall o \in O \qquad (7)$$

$$0 \leq u_i^o \leq |\,S(o)\,| - 1 \forall v_i \in V, \qquad \forall o \in O \qquad (8)$$

- Số lần ghé thăm tương ứng số SKU cần lấy:

$$\sum_{v_i \in V_0} \sum_{v_j \in V} x_{ij}^{o} = |\,S(o)\,| \qquad \forall o \in O \qquad (9)$$

❖ Ràng buộc liên kết giữa gán vị trí và định tuyến

Để đảm bảo rằng lộ trình của đơn hàng phải ghé qua các location chứa SKU của đơn đó, ràng buộc liên kết được xây dựng như sau:

$$\sum_{v_i \in V_0} \sum_{v_j \in V(l)} x_{ij}^{o} \;\geq \sum_{s \in S(o)} \xi_{ls} \qquad \forall l \in L, \forall o \in O \qquad (10)$$

Ràng buộc này đảm bảo rằng nếu nhiều SKU của cùng một đơn được gán vào cùng một location, lộ trình phải ghé location đó đủ số lần tương ứng.

❖ Miền biến (compact formulation)

$$x_{ij}^{o} \in \{0,1\} \quad \forall o \in O, \ (v_i, v_j) \in E \qquad (11)$$
$$\xi_{ls} \in \{0,1\} \quad \forall l \in L, \quad s \in S \qquad (12)$$

❖ Mô hình mở rộng (Dantzig–Wolfe formulation)

Do số lượng lộ trình khả thi tăng theo cấp số mũ, mô hình compact có LP relaxation yếu. Vì vậy, bài toán được mở rộng bằng phân rã Dantzig–Wolfe.

Ký hiệu:

- R_o: tập các lộ trình hợp lệ cho đơn o.
- $\rho_{or} \in \{0,1\}$: bằng 1 nếu đơn o sử dụng lộ trình $r \in R_o$.
- c_{or}: chi phí của lộ trình r.
- a_{lor}: số lần lộ trình r ghé location l.

Mô hình mở rộng được viết như sau:

$$min \sum_{o \in O} \sum_{r \in R_o} c_{or} p_{or} \qquad (13)$$

$$\sum_{r \in R_o} \rho_{or} = 1 \qquad \forall o \in O \qquad (14)$$

$$\sum_{r \in R_o} a_{lor}\, \rho_{or} \geq \sum_{s \in S(o)} \xi_{ls} \qquad \forall l \in L, \forall o \in O \qquad (15)$$

Các ràng buộc gán vị trí (2)–(4) vẫn được giữ nguyên.

Miền biến:

$$\rho_{or} \in \{0,1\}, \qquad \xi_{ls} \in \{0,1\} \qquad (16)$$

❖ Điểm mới trong mô hình hóa

Điểm mới quan trọng của mô hình là:

- Sử dụng mô hình toán học mở rộng thay vì compact formulation.
- Đóng gói toàn bộ quyết định định tuyến vào các biến lộ trình.
- Tăng cường liên kết giữa gán vị trí và định tuyến thông qua các ràng buộc liên kết và các bất đẳng thức tăng cường (SL, SL-1), giúp cải thiện đáng kể chất lượng nghiệm LP.

3.2 Phương pháp giải bài toán

Bài toán SLAPRP là một bài toán tối ưu tổ hợp tích hợp giữa gán vị trí lưu trữ (SLAP) và định tuyến người lấy hàng (PRP). Do số lượng cấu hình gán SKU–location và số lượng route khả thi đều tăng rất nhanh theo quy mô bài toán, việc giải trực tiếp mô hình compact bằng MIP thường gặp hai vấn đề: (i) kích thước mô hình lớn và (ii) LP relaxation yếu, dẫn đến thời gian tìm nghiệm tối ưu tăng mạnh. Phương pháp Branch-Cut-and-Price (BCP) là một khung giải phù hợp kết hợp Dantzig–Wolfe decomposition + Column Generation + Cutting Planes + Branch-and-Bound.

3.2.1 Tổng quan khung Branch-Cut-and-Price

Phương pháp BCP vận hành theo nguyên tắc: thay vì mô hình hóa định tuyến bằng biến cạnh x_{ij}^{o}(compact), phần định tuyến được biểu diễn bằng biến route ρ_{or} trong mô hình Dantzig–Wolfe (DW). Mỗi route r là một tour hợp lệ cho đơn o, có chi phí c_{or} và "mức ghé" location l là a_{lor}. Vì số lượng route là cực lớn (thường cấp số mũ), mô hình DW không thể liệt kê toàn bộ biến ρ_{or} ngay từ đầu. Do đó thuật toán sử dụng Column Generation để chỉ sinh các route "cần thiết" trong quá trình giải.

BCP triển khai theo vòng lặp:

1. Giải Restricted Master Problem (RMP) (một phiên bản DW chỉ chứa một tập con các route)
2. Dùng nghiệm đối ngẫu của RMP để giải Pricing Problem (tìm route mới có reduced cost âm)
3. Thêm route mới vào RMP và lặp cho đến khi không còn route cải thiện.
4. Nếu nghiệm còn phân số, thực hiện cắt để siết LP relaxation và hoặc branching để đảm bảo nghiệm nguyên.

Cấu trúc này vừa đảm bảo tính đúng đắn vừa giảm đáng kể kích thước mô hình so với MIP compact.

3.2.2 Bài toán chính: Restricted Master Problem (RMP)

Từ mô hình DW ở mục 3.1, RMP tại một thời điểm chỉ giữ lại tập route đang có $R'_o \subset R_o$. RMP quyết định:

- Các biến gán vị trí ξ_{ls}.
- Các biến lựa chọn route ρ_{or}với $r \in R'_o$.

RMP gồm các ràng buộc:

- Mỗi SKU gán đúng một location và thỏa dung lượng location

- Mỗi đơn chọn đúng một route (trên tập route hiện có)
- Ràng buộc liên kết: tổng "lượt ghé" location do route tạo ra phải đủ để phục vụ số SKU của đơn đang được gán vào location đó.

Trong giai đoạn column generation, RMP thường được giải ở dạng LP relaxation (cho phép ρ, ξ liên tục trong [0, 1]) để lấy thông tin đối ngẫu phục vụ pricing. Khi cần nghiệm nguyên tối ưu, thuật toán chuyển sang nhánh–cận (branch-and-bound) trên cấu trúc BCP.

3.2.3 Bài toán con: Pricing Problem (sinh route)

Sau khi giải RMP, ta thu được các biến đối ngẫu (dual variables) tương ứng với các ràng buộc, đặc biệt:

- Đối ngẫu của ràng buộc "mỗi đơn chọn 1 route".
- Đối ngẫu của ràng buộc liên kết location–assignment.

Với mỗi đơn o, pricing problem tìm một route r sao cho reduced cost âm:

$$\bar{c}_{or} = c_{or} - \pi_o - \sum_{l \in L} \mu_{lo} \; a_{lor} \; < \; 0 \quad (17)$$

trong đó π_o là dual của ràng buộc chọn route cho đơn o, và μ_{lo} là dual của ràng buộc liên kết tại location l cho đơn o. (Công thức reduced cost có thể thay đổi chi tiết theo cách paper triển khai, nhưng nguyên lý là: chi phí route trừ đi "lợi ích đối ngẫu" do route thỏa ràng buộc.)

Pricing problem được mô hình hóa trên đồ thị kho $G = (V_0, E)$ dưới dạng Elementary Shortest Path Problem with Resource Constraints (ESPPRC):

- "Shortest path" vì cần tối thiểu reduced cost.
- "Elementary" vì tránh lặp nút (hoặc tương đương tránh chu trình không cần thiết)
- "Resource constraints" vì route phải thỏa các điều kiện như ghé đủ location/điểm cần thiết của đơn, cấu trúc di chuyển hợp lệ trong kho, giới hạn số lần ghé, v.v.

ESPPRC được giải bằng thuật toán nhãn (labeling / dynamic programming) với cơ chế dominance để loại bỏ nhãn kém (không thể dẫn đến route tối ưu), giúp giảm đáng kể số trạng thái phải duyệt. Route có reduced cost âm sẽ được đưa về RMP như một cột mới.

Khởi tạo tập route ban đầu thường dùng các route đơn giản (ví dụ: route đi qua các vị trí cần ghé theo một thứ tự hợp lệ, hoặc route ngẫu nhiên/hợp lệ tối thiểu) để RMP khả thi ngay từ đầu.

3.2.4 Cutting planes: tăng cường mô hình (cuts)

Nếu chỉ dùng DW + column generation, LP relaxation vẫn có thể "lỏng", khiến nghiệm phân số nhiều và cây branch-and-bound lớn. Bài báo đưa vào các bất đẳng thức tăng cường (valid inequalities), đặc biệt nhóm SL inequalities và trường hợp SL-1, nhằm siết chặt liên kết giữa:

- Việc một SKU được gán vào location nào (ξ_{ls})
- Việc route có thực sự dừng/ghé location đó hay không (thông tin nằm trong ρ_{or}).

Các cuts được phát hiện bằng thủ tục separation: từ nghiệm hiện tại của RMP (thường là nghiệm phân số), thuật toán kiểm tra xem có bất đẳng thức nào bị vi phạm không. Nếu có, bất đẳng thức được thêm vào RMP và giải lại. Việc bổ sung cuts thường làm:

- Giảm gap giữa nghiệm LP và nghiệm nguyên.
- Giảm số node khi branch.
- Tăng tốc hội tụ tổng thể của BCP.

3.2.5 Branching: đảm bảo nghiệm nguyên

Sau khi column generation hội tụ ở cấp LP (không còn route có reduced cost âm) và không còn cuts bị vi phạm, nếu nghiệm vẫn phân số thì thuật toán thực hiện branching. Điểm quan trọng của BCP là chọn chiến lược branching sao cho:

- Không phá vỡ cấu trúc pricing problem.
- Hoặc nếu có, vẫn giữ được pricing giải được hiệu quả.

Theo hướng triển khai trong paper, branching thường ưu tiên trên biến assignment ξ_{ls} (ví dụ: ép $\xi_{ls} = 0$ hoặc $\xi_{ls} = 1$), vì:

- ξ là biến "cấu trúc", quyết định location của SKU,
- Tác động trực tiếp tới ràng buộc liên kết,
- Giữ pricing problem vẫn là ESPPRC (chỉ thay đổi tập location bắt buộc/khả dĩ).

Khi phân nhánh, mỗi node của cây B&B sẽ có một RMP riêng (với tập route hiện có) và tiếp tục chạy lại column generation + cuts ở node đó. Đây chính là "Price" nằm trong Branch-Cut-and-Price.

3.2.6 Tóm tắt quy trình thuật toán

Quy trình tổng quát mà dự án triển khai có thể mô tả như sau:

1. Khởi tạo: đọc instance (kho, SKU, orders), tạo tập route ban đầu R'_o để RMP khả thi.
2. Giải RMP (LP): tối ưu RMP ở dạng relaxation để lấy nghiệm primal và dual.

3. Pricing: với từng đơn o, giải ESPPRC để tìm route mới có reduced cost âm.
4. Thêm cột: nếu tìm được route mới, thêm vào R'_ovà quay lại bước 2.
5. Cut separation: nếu không còn route cải thiện, kiểm tra cuts (SL/SL-1, …); nếu có vi phạm, thêm cut và quay lại bước 2.
6. Branching: nếu nghiệm còn phân số, chọn biến branching (ưu tiên ξ_{ls}), tạo node con và lặp lại bước 2–5 cho từng node.
7. Dừng: khi tìm được nghiệm nguyên tối ưu và chứng minh tối ưu (node list rỗng / bound hội tụ).

❖ Phương pháp Branch-Cut-and-Price phù hợp với SLAPRP vì ba lý do chính:

- Quy mô route cực lớn: column generation là cơ chế chuẩn để xử lý số biến cấp số mũ bằng cách chỉ sinh biến cần thiết.
- Liên kết lưu trữ–định tuyến phức tạp: mô hình DW + cuts (SL/SL-1) giúp siết chặt ràng buộc liên kết, cải thiện đáng kể LP relaxation.
- Yêu cầu nghiệm tối ưu (exact): BCP vẫn đảm bảo tối ưu toàn cục nhờ branch-and-bound, đồng thời kiểm soát được độ phức tạp tốt hơn MIP compact.

3.3 Đánh giá độ phức tạp và tốc độ hội tụ của phương pháp

Bài toán SLAPRP là một bài toán tối ưu tổ hợp có độ phức tạp rất cao do tích hợp đồng thời hai lớp quyết định: gán vị trí lưu trữ và định tuyến người lấy hàng. Việc đánh giá độ phức tạp và tốc độ hội tụ của các phương pháp giải là cần thiết để làm rõ vì sao khung Branch-Cut-and-Price (BCP) được lựa chọn thay cho các mô hình MIP truyền thống.

3.3.1 Độ phức tạp tính toán của bài toán SLAPRP

Về bản chất, SLAPRP thuộc lớp NP-hard mạnh, do nó bao hàm ít nhất hai bài toán NP-hard đã biết:

Thứ nhất, bài toán gán vị trí lưu trữ (SLAP) tương đương với một dạng bài toán gán có ràng buộc dung lượng. Với $|S|$ SKU và $|L|$ location, số cấu hình gán khả thi tăng theo cấp số mũ khi xét đầy đủ các hoán vị và tổ hợp thỏa mãn dung lượng.

Thứ hai, bài toán định tuyến người lấy hàng (PRP) là một dạng bài toán định tuyến/TSP với các ràng buộc bổ sung. Với mỗi đơn hàng $o$có $|S(o)|$ SKU, số route khả thi tăng rất nhanh theo $|S(o)|$, đặc biệt trong mô hình của bài báo khi cho phép ghé nhiều lần cùng một location.

Quan trọng hơn, trong SLAPRP, hai lớp quyết định này không độc lập: một phương án gán vị trí chỉ có thể được đánh giá đúng khi xét routing tối ưu tương ứng, và ngược lại. Điều

này khiến không gian nghiệm của bài toán là tích Descartes của hai không gian tổ hợp lớn, làm cho độ phức tạp tổng thể tăng rất nhanh khi quy mô kho và số đơn hàng tăng.

❖ Hạn chế về độ phức tạp và hội tụ của mô hình MIP compact

Trong mô hình compact, việc định tuyến được mô tả bằng các biến nhị phân x_{ij}^{o} cho từng cung và từng đơn hàng. Khi số node lưu trữ tăng, số biến x_{ij}^{o} tăng theo $|O| \cdot |E|$, dẫn đến kích thước mô hình rất lớn.

Ngoài vấn đề kích thước, LP relaxation của mô hình compact thường rất yếu, chủ yếu do:

- Ràng buộc loại subtour (MTZ hoặc flow-based) không đủ mạnh.
- Ràng buộc liên kết giữa assignment và routing chỉ được mô hình hóa ở mức tổng quát.

Hệ quả là optimality gap lớn, khiến thuật toán branch-and-bound phải duyệt qua rất nhiều node để chứng minh tối ưu. Trong thực nghiệm được báo cáo trong bài báo, mô hình compact chỉ giải được các instance nhỏ, và thời gian giải tăng nhanh khi kích thước kho hoặc số đơn tăng.

3.3.2 Cải thiện độ phức tạp bằng Dantzig–Wolfe decomposition và Column Generation

Dantzig–Wolfe decomposition giúp tái cấu trúc bài toán bằng cách:

- Tách phần định tuyến ra khỏi mô hình gán vị trí.
- Biểu diễn định tuyến bằng các biến route ρ_{or} thay vì biến cạnh.

Mặc dù về mặt lý thuyết số route là cấp số mũ, column generation cho phép chỉ sinh ra các route có khả năng cải thiện nghiệm hiện tại. Điều này làm giảm đáng kể số biến phải xử lý đồng thời trong bài toán chính.

Ngoài ra, mô hình DW thường cho LP relaxation mạnh hơn so với compact formulation, vì nó mô hình hóa trực tiếp convex hull của tập route hợp lệ (hoặc xấp xỉ rất gần). Kết quả là:

- Bound LP tốt hơn ngay từ gốc.
- Ít vòng branch-and-bound hơn nếu cần nghiệm nguyên.

Tuy nhiên, DW + CG thuần túy vẫn chưa đủ, vì nghiệm LP có thể còn phân số nhiều và việc chứng minh tối ưu vẫn cần thêm cơ chế.

3.3.3 Vai trò trung tâm của Pricing Problem và tác động tới độ phức tạp

Trong Column Generation và BCP, pricing problem là thành phần chi phối lớn nhất đến thời gian chạy. Pricing được mô hình hóa dưới dạng ESPPRC, vốn cũng là một bài toán NP-hard theo worst-case.

Độ phức tạp của pricing phụ thuộc vào:

- Kích thước đồ thị kho.
- Số location cần ghé của mỗi đơn.
- Cấu trúc ràng buộc tài nguyên (số lần ghé, tránh chu trình, thứ tự ghé).

Pricing được giải bằng thuật toán labeling với cơ chế dominance rules để loại bỏ sớm các nhãn không tiềm năng. Điều này giúp giảm mạnh số trạng thái cần xét trong thực tế, mặc dù độ phức tạp lý thuyết vẫn cao.

Thực nghiệm cho thấy phần lớn thời gian của BCP được tiêu tốn cho pricing, do đó hiệu quả của labeling và dominance là yếu tố quyết định tốc độ hội tụ.

3.3.4 Tăng tốc hội tụ bằng Cutting Planes

Để giảm số vòng CG và số node branch-and-bound, bài báo đề xuất các bất đẳng thức tăng cường (valid inequalities), đặc biệt là nhóm SL và SL-1 inequalities.

Các bất đẳng thức này giúp:

- Siết chặt liên kết giữa việc một SKU được gán vào location và việc route có thực sự dừng tại location đó.
- Loại bỏ nhiều nghiệm LP phân số "phi thực tế" mà DW relaxation cho phép.

Tác động của cutting planes tới hội tụ thể hiện ở ba khía cạnh:

1. Giảm optimality gap tại node gốc.
2. Giảm số route cần sinh, vì nhiều route yếu bị loại bỏ sớm.
3. Giảm số node branch-and-bound, vì LP relaxation chặt hơn.

Mặc dù mỗi lần thêm cut làm bài toán LP nặng hơn, tổng thời gian giải thường giảm do số vòng lặp và số node giảm đáng kể.

3.3.5 Ảnh hưởng của chiến lược Branching tới tốc độ hội tụ

Trong BCP, chiến lược branching ảnh hưởng trực tiếp tới kích thước cây tìm kiếm. Bài báo và code ưu tiên branching trên biến assignment ξ_{ls} vì:

- ξ là biến cấu trúc, quyết định layout lưu trữ.

- Branching trên ξ làm giảm mạnh tính đối xứng của bài toán.
- Quan trọng nhất, branching trên ξ không phá vỡ cấu trúc pricing, tức pricing problem vẫn là ESPPRC với cùng dạng, chỉ khác dữ liệu đầu vào.

Nếu branching trên các đặc trưng routing quá chi tiết, pricing có thể trở nên phức tạp hơn nhiều, dẫn đến thời gian chạy tăng mạnh. Do đó, lựa chọn branching trên assignment là yếu tố quan trọng giúp BCP hội tụ nhanh trong thực tế.

❖ Đánh giá tổng thể tốc độ hội tụ

Tổng hợp các phân tích trên, có thể rút ra các nhận xét sau:

- Mô hình MIP compact có độ phức tạp và thời gian hội tụ tăng rất nhanh, chỉ phù hợp cho instance nhỏ.
- Dantzig–Wolfe + Column Generation cải thiện đáng kể bound và khả năng xử lý routing, nhưng vẫn cần cơ chế đảm bảo nghiệm nguyên.
- Branch-Cut-and-Price là khung giải phù hợp nhất cho SLAPRP vì tận dụng đồng thời:
    - Bound mạnh của DW
    - Sinh biến thông minh của CG
    - Siết miền nghiệm bằng cutting planes
    - Và đảm bảo tối ưu toàn cục bằng branching có định hướng.

Nhờ sự kết hợp này, BCP đạt tốc độ hội tụ vượt trội so với các phương pháp khác khi giải các instance SLAPRP có kích thước vừa và lớn, như được minh chứng trong bài báo và quá trình triển khai bằng mã nguồn Julia.

Chương 4: Phân tích kết quả

4.1. Các thông số dùng để đánh giá hệ thống

Trong bài toán Storage Location Assignment and Picker Routing Problem (SLAPRP), hệ thống được đánh giá thông qua một tập hợp các thông số phản ánh đồng thời chất lượng nghiệm tối ưu, hiệu năng thuật toán và mức độ phù hợp với thực tiễn logistics. Các thông số này được sử dụng nhất quán trong quá trình thực nghiệm nhằm đảm bảo khả năng so sánh giữa các cấu hình mô hình và thuật toán khác nhau.

Các thông số này không chỉ đo lường hiệu suất thô mà còn cung cấp cái nhìn sâu sắc về hành vi, hiệu quả và chất lượng giải pháp của thuật toán Branch-Cut-and-Price (BCP).

Việc đánh giá hiệu quả của hệ thống giải bài toán Storage Location Assignment and Picker Routing Problem (SLAPRP) được thực hiện thông qua các nhóm chỉ số sau:

4.1.1 Thông số tối ưu hoá (Optimization Metrics)

Tổng quãng đường picking (Total Picking Distance)

Đây là chỉ số quan trọng nhất của hệ thống, tương ứng với giá trị hàm mục tiêu của mô hình toán học. Chỉ số này phản ánh tổng chiều dài các route mà picker phải di chuyển để hoàn thành toàn bộ tập đơn hàng. Việc tối thiểu hoá tổng quãng đường picking giúp giảm chi phí lao động, thời gian xử lý đơn hàng và nâng cao hiệu suất vận hành kho.

Hàm mục tiêu được mô hình hoá như sau:

Trong đó:

- R là tập các route khả thi
- là chi phí (độ dài) của route r
- $\in\{0,1\}$ là biến quyết định, bằng 1 nếu route r được chọn.

Chi phí route theo từng đơn hàng (Route Cost)

Mỗi đơn hàng (order) được gán một route riêng, với chi phí được tính dựa trên layout kho và chính sách routing được sử dụng. Metric này cho phép phân tích chi tiết đóng góp của từng đơn hàng vào tổng chi phí, từ đó đánh giá mức độ cân bằng của các route được sinh ra.

4.1.2. Thông số đánh giá chất lượng nghiệm (Solution Quality Metrics)

Primal Bound

Là giá trị hàm mục tiêu của nghiệm nguyên tốt nhất tìm được trong quá trình chạy thuật toán. Chỉ số này thể hiện chi phí thực tế mà hệ thống đạt được.

Trong đó là nghiệm nguyên tốt nhất hiện tại.

Dual Bound

Là cận dưới của bài toán, thu được từ nghiệm tối ưu của bài toán thư giãn tuyến tính (LP relaxation). Dual bound đóng vai trò quan trọng trong việc đánh giá mức độ gần tối ưu của nghiệm.

Trong đó:

- O là tập các đơn hàng
- là biến đối ngẫu tương ứng với ràng buộc mỗi đơn hàng được phục vụ đúng một lần.

Optimality Gap (%)

Khoảng cách tối ưu được tính bằng tỷ lệ phần trăm giữa primal bound và dual bound. Chỉ số này cho biết mức độ sai lệch của nghiệm so với nghiệm tối ưu toàn cục. Gap nhỏ cho thấy nghiệm đạt chất lượng cao và thuật toán hội tụ tốt.

4.1.3. Thông số hiệu năng thuật toán (Algorithmic Performance Metrics)

Thời gian tính toán (Computational Time)

Thời gian cần thiết để thuật toán tìm được nghiệm tối ưu hoặc đạt đến giới hạn thời gian cho phép. Metric này phản ánh khả năng áp dụng của thuật toán trong các hệ thống thực tế.

Số node trong cây Branch-and-Bound

Chỉ số này biểu thị số lượng node được duyệt trong quá trình tìm kiếm. Số node càng lớn thì không gian tìm kiếm càng phức tạp, đồng thời phản ánh độ khó của instance.

Số lượng cột sinh ra (Number of Generated Columns)

Do thuật toán sử dụng Column Generation, số lượng cột được sinh ra trong quá trình giải là một chỉ số quan trọng để đánh giá hiệu quả của bài toán pricing và mức độ hội tụ của mô hình Dantzig–Wolfe.

Số lượng bất đẳng thức tăng cường đang hoạt động (Active Cuts)

Chỉ số này thể hiện số lượng các bất đẳng thức hợp lệ (ví dụ: SL-1 và SL cuts) được sử dụng tại nghiệm hiện tại. Việc theo dõi metric này giúp đánh giá mức độ tăng cường của mô hình và ảnh hưởng của các cuts đến chất lượng nghiệm.

4.2. Mô tả tập dữ liệu đầu vào

4.2.1 Tập dữ liệu đầu vào

Tập dữ liệu Guo

Trong báo cáo này, tập dữ liệu đầu vào được sử dụng là Guo benchmark, một bộ dữ liệu chuẩn thường được dùng để đánh giá các thuật toán giải bài toán SLAPRP. Bộ dữ liệu này được xây dựng dựa trên các kịch bản kho hàng thực tế trong lĩnh vực logistics và thương mại điện tử, đảm bảo tính đại diện và khả năng so sánh với các nghiên cứu trước đó.

Tập dữ liệu Guo được lưu trữ dưới dạng file văn bản (.txt), trong đó dữ liệu được tổ chức theo cấu trúc tuần tự, bao gồm các thành phần chính sau:

- Thông tin tổng quan về instance: số lượng lối đi (aisle) trong kho hàng, số lượng vị trí lưu trữ, số lượng đơn vị lưu kho SKU (Stock Keeping Unit) và số lượng đơn hàng.
- Thông tin layout kho: mô tả cấu trúc không gian của kho, phục vụ cho việc tính toán chính xác khoảng cách di chuyển của picker.
- Danh sách các đơn hàng (orders): mỗi đơn hàng được biểu diễn bởi một tập các SKU cần được picking.
- Các tham số bổ sung: bao gồm các hệ số hoặc cấu hình dùng để sinh dữ liệu, phục vụ việc tái lập thí nghiệm.

Việc lưu trữ dữ liệu dưới dạng file .txt giúp dữ liệu có tính linh hoạt cao, dễ dàng chỉnh sửa, mở rộng và sử dụng trong các môi trường lập trình khác nhau.

4.2.2 Mô tả file dữ liệu Guo

Tập dữ liệu Guo được lưu trữ dưới dạng file văn bản (.txt), trong đó dữ liệu được tổ chức theo cấu trúc tuần tự, bao gồm các thành phần chính như đã nêu ở trên. Cụ thể, file dữ liệu tuân theo một quy ước lưu trữ cố định theo thứ tự dòng, trong đó ý nghĩa của từng dòng dữ liệu được xác định bởi vị trí của nó trong file.

File dữ liệu đầu vào của Guo được tổ chức theo cấu trúc tuần tự gồm ba phần chính: (i) tham số bố cục kho, (ii) danh sách các lô đơn hàng và (iii) các phân bổ vị trí cố định. Cách tổ chức này phản ánh trực tiếp các thành phần cốt lõi của bài toán SLAPRP.

Phần dữ liệu	**Vị trí trong file**	**Dạng dữ liệu**	**Nội dung lưu trữ**	**Ý nghĩa trong mô hình SLAPRP**
(i) Tham số bố cục kho	Các dòng đầu file(1-4)	Số nguyên và danh sách số nguyên	Số aisle, số bay, tham số hình học kho, tổng số SKU, tổng số vị trí lưu trữ và dung lượng của từng vị trí	Xác định cấu trúc hình học kho, đồ thị di chuyển và ràng buộc sức chứa cho bài toán SLAPRP

(ii) Danh sách các lô đơn hàng (Orders)	Các dòng tiếp theo	Danh sách số nguyên (độ dài thay đổi)	Mỗi dòng biểu diễn một lô đơn hàng, gồm các ID SKU cần được picking	Xác định các điểm phải ghé thăm trong bài toán định tuyến người nhặt hàng (picker routing)
(iii) Phân bổ vị trí cố định (Fixed assignments – F)	Các dòng cuối file	Cặp số nguyên (s, l)	SKU s đã được gán sẵn tại vị trí lưu trữ l	Ràng buộc replenishment: các SKU này không được tái gán, chỉ tối ưu các SKU còn trống

Ví dụ:

Xét file dữ liệu SLAPRP_Guo_small_O50_alpha0.2_v1.txt, dữ liệu được tổ chức tuần tự và có thể phân tích trực tiếp theo thứ tự xuất hiện trong file như sau.

```
4 5
1 2 5
40
51
4 3 8 7 8 7 3 1 2 7 5 8 8 5 5 6 4 4 1 8 8 1 6 4 6 5 7 4 3 2 5 4 4 3 8 8 7 2 3 7 1 6 7 4 6 4 6 5 3 6
6
1 4 6 7
3 5 25
3 5 6 7 8 31 35 38
1 2 3 4 6 8 24
1 2 4 5 6 7 8 34
1 2 3 4 5 6 19
1 4 7
4
3 4
2 3 4 5 6 33 34
1 2 5 11 37
2 3 4 5 7 8 15 26
1 3 5 6 7 8 30 35
2 3 4 8 19
1 5 7 8 9
2 3 4 8 31 36
5 8 33 34
1 3 5 39
8
1 2 4 5 6 8 17 31
```

Tham số bố cục kho

Các dòng đầu tiên của file:

- Dòng 4 5 cho biết kho hàng có 4 aisle và 5 bay trên mỗi aisle, từ đó xác định cấu trúc hình học của kho.
- Dòng 1 2 5 chứa các tham số hình học liên quan đến khoảng cách giữa các block và cross-aisle, được dùng để tính chi phí di chuyển.

- Dòng 40 biểu thị tổng số SKU trong kho.
- Dòng 51 cho biết tổng số vị trí lưu trữ (storage nodes), lớn hơn số SKU, cho phép tồn tại các vị trí trống.

Ngay sau đó là một dòng gồm 51 số nguyên:

4 3 8 7 8 7 3 1 2 7 5 8 ...

Mỗi số biểu thị dung lượng của một vị trí lưu trữ, tức là số node vật lý thuộc cùng một location. Thông tin này tạo nên ràng buộc sức chứa trong mô hình SLAPRP.

Danh sách các lô đơn hàng (Orders)

Sau phần cấu hình kho, file chứa 50 dòng liên tiếp, mỗi dòng tương ứng với một đơn hàng.

- Dòng 1 4 6 7 biểu diễn một đơn hàng yêu cầu nhặt 4 SKU: 1, 4, 6 và 7.
- Dòng 3 5 25 là một đơn hàng khác với 3 SKU.
- Dòng 3 5 6 7 8 31 35 38 là đơn hàng có 8 SKU, thể hiện quy mô đơn hàng lớn hơn.

Trong mô hình SLAPRP, mỗi dòng tương ứng với một tuyến đường của người nhặt hàng, và các SKU trong dòng xác định các vị trí mà tuyến đường đó phải ghé thăm.

Phân bổ vị trí cố định (Fixed assignments – Replenishment)

Phần cuối của file gồm các cặp số nguyên, ví dụ:

- Cặp 16 4 cho biết SKU 16 đã được gán cố định tại vị trí lưu trữ 4.
- Cặp 35 18 cho biết SKU 35 nằm sẵn tại vị trí 18.

Các cặp này tạo thành tập F (fixed assignments). Trong bài toán SLAPRP – biến thể replenishment, các phân bổ này không được thay đổi. Thuật toán chỉ cần quyết định vị trí lưu trữ cho các SKU chưa xuất hiện trong tập F.

4.3 Phân tích kết quả

File result_guo_return.csv lưu trữ kết quả thực nghiệm thu được sau khi giải các instance SLAPRP của Guo bằng mô hình return routing. Mỗi dòng trong file tương ứng với một instance dữ liệu đầu vào và một lần chạy thuật toán.

result_guo_return.csv

Instance	Aisles	Rows	Orders	OrderLines	Slots	Picks	Skewness	Version	UB	LB	Gap	Time	Optimality	Nodes	Root_boun	Cuts	Policy	Solution
SLAPRP_Guo_small_O100_alpha0.2_v1.txt	4	5	104	4	40	416	nothing	1	3794	3794	0	4.5	TRUE	1	3794	2136	guo_return	5/23/20/25/16/37/9/19/18/40/4/17/24/33/2/
SLAPRP_Guo_small_O100_alpha0.2_v10.txt	4	5	102	5	40	510	nothing	0	3751	3752	0	4.8	TRUE	1	3751	2188	guo_return	5/13/3/24/4/18/21/36/7/32/20/29/9/22/11/2
SLAPRP_Guo_small_O100_alpha0.2_v2.txt	4	5	101	7	40	707	nothing	2	3995	3996	0	4.2	TRUE	1	3995	2196	guo_return	10/31/24/37/14/29/25/32/11/39/9/15/13/34
SLAPRP_Guo_small_O100_alpha0.2_v3.txt	4	5	104	7	40	728	nothing	3	4214	4214	0	4.8	TRUE	1	4214	2282	guo_return	1/39/13/28/10/34/26/32/15/22/6/18/3/23/16
SLAPRP_Guo_small_O100_alpha0.2_v4.txt	4	5	104	4	40	416	nothing	4	4009	4016	0	8.8	TRUE	7	3982	2132	guo_return	7/19/18/34/14/24/10/13/3/21/12/22/4/25/6/
SLAPRP_Guo_small_O100_alpha0.2_v5.txt	4	5	100	6	40	600	nothing	5	3814	3814	0	3.9	TRUE	1	3813	2236	guo_return	3/19/8/15/7/34/9/14/4/17/18/37/10/22/29/3
SLAPRP_Guo_small_O100_alpha0.2_v6.txt	4	5	100	8	40	800	nothing	6	4183	4184	0	4	TRUE	1	4183	2160	guo_return	5/15/13/17/10/12/24/39/6/18/1/14/11/23/8/
SLAPRP_Guo_small_O100_alpha0.2_v7.txt	4	5	104	8	40	832	nothing	7	4225	4226	0	3.1	TRUE	1	4225	2196	guo_return	17/33/15/23/36/40/21/27/9/39/5/26/25/31/1
SLAPRP_Guo_small_O100_alpha0.2_v8.txt	4	5	101	8	40	808	nothing	8	3911	3912	0	3.3	TRUE	1	3911	2056	guo_return	8/38/1/39/20/35/5/10/33/36/12/13/6/9/3/34
SLAPRP_Guo_small_O100_alpha0.2_v9.txt	4	5	101	5	40	505	nothing	9	4323	4324	0	4.9	TRUE	1	4323	2200	guo_return	14/38/9/26/4/36/11/37/2/19/5/31/13/16/7/3
SLAPRP_Guo_small_O100_alpha0.3_v1.txt	4	5	105	5	40	525	nothing	1	3780	3750	0	11.8	TRUE	3	3748.5	2054	guo_return	4/8/20/27/10/36/9/21/22/38/14/30/23/33/3/
SLAPRP_Guo_small_O100_alpha0.3_v10.txt	4	5	106	8	40	848	nothing	0	3970	3964	0	18.9	TRUE	3	3962.19	2209	guo_return	1/22/12/16/8/15/7/37/20/30/31/34/29/33/19
SLAPRP_Guo_small_O100_alpha0.3_v2.txt	4	5	106	1	40	106	nothing	2	3460	3456	0	28.3	TRUE	5	3443.36	2202	guo_return	4/28/15/30/2/7/6/8/17/27/19/33/12/16/14/3
SLAPRP_Guo_small_O100_alpha0.3_v3.txt	4	5	101	7	40	707	nothing	3	3552	3550	0	11.3	TRUE	5	3539.67	2001	guo_return	5/35/15/30/7/33/9/26/24/34/10/38/3/27/4/2
SLAPRP_Guo_small_O100_alpha0.3_v4.txt	4	5	100	6	40	600	nothing	4	4072	4072	0	9.7	TRUE	10	4065.02	2240	guo_return	18/37/4/16/10/30/25/28/19/26/32/35/15/29
SLAPRP_Guo_small_O100_alpha0.3_v5.txt	4	5	105	6	40	630	nothing	5	3932	3932	0	5.4	TRUE	1	3932	2132	guo_return	10/24/18/38/27/39/25/40/16/28/12/17/31/3

Cấu trúc tổng quát của file CSV

- Số dòng: 90
- Số cột: 19
- Mỗi dòng: 1 instance + 1 nghiệm tối ưu / gần tối ưu

4.3.1 Nhóm cột mô tả instance

Cột	Ý nghĩa
Instance	Tên file dữ liệu đầu vào .txt
Aisles	Số aisle của kho
Rows	Số bay / hàng trong mỗi aisle
Orders	Số lượng đơn hàng
OrderLines	Số SKU trung bình trên mỗi đơn
Slots	Tổng số SKU
Picks	Tổng số lần picking

4.3.2 Nhóm cột đặc trưng dữ liệu

Cột	Ý nghĩa
Skewness	Đặc điểm phân bố SKU (ví dụ: none / skewed)
Version	Phiên bản instance (v1, v2, …)
Policy	Chính sách routing được sử dụng (guo_return)

4.3.3 Nhóm cột kết quả tối ưu hóa

Cột	Ý nghĩa
UB	Giá trị hàm mục tiêu tốt nhất (Upper Bound)

LB	Cận dưới (Lower Bound)
Gap	Khoảng cách tối ưu (%)
Optimality	Có chứng minh tối ưu hay không
Time	Thời gian giải (giây)

4.3.4 Nhóm cột thông tin thuật toán

Cột	Ý nghĩa
Nodes	Số node trong cây Branch-and-Bound
Root_bound	Giá trị cận tại node gốc
Cuts	Số lượng cutting planes được sinh
Solution	Thứ tự gán SKU vào các vị trí lưu trữ

Ví dụ:

Kết quả thu được của file đầu vào SLAPRP_Guo_small_O50_alpha0.2_v1.txt

Phân tích:

Thuộc tính	**Giá trị**	**Phân tích & Đánh giá**
Instance	SLAPRP_Guo_small_O50_alpha0.2_v1.txt	Instance thuộc bộ dữ liệu Guo (small), dùng để kiểm chứng mô hình SLAPRP với biến thể bổ sung hàng tồn kho
Aisles	4	Quy mô kho nhỏ, phù hợp để kiểm tra tính đúng đắn và độ ổn định của mô hình
Rows	5	Bố cục kho đơn giản, không gây độ phức tạp cao trong tính toán khoảng cách
Orders	51	Số lượng đơn hàng ở mức trung bình, phù hợp với bài toán thử nghiệm
OrderLines	4	Quy mô mỗi đơn hàng vừa phải, không tạo tải đột biến cho bài toán routing
Slots	40	Tổng số SKU phù hợp với nhóm instance small

Picks	204	Khối lượng nhặt hàng hợp lý so với số đơn và SKU
Skewness	nothing	Phân bố SKU đồng đều → mô hình không bị thiên lệch, kết quả ổn định
Version	1	Phiên bản dữ liệu chuẩn, không có điều chỉnh đặc biệt
Policy	guo_return	Chính sách routing đơn giản, phù hợp để đánh giá hiệu quả gán vị trí
UB	1832	Giá trị nghiệm hợp lý cho kho nhỏ, phản ánh đúng tổng quãng đường
LB	1832	Trùng với UB → mô hình chặt, nghiệm đạt tối ưu
Gap	0	Không có sai lệch → đáp ứng yêu cầu tối ưu
Optimality	TRUE	Thuật toán chứng minh nghiệm tối ưu → độ tin cậy cao
Nodes	1	Giải tại node gốc → bài toán ổn định, không bùng nổ nhánh
Root_bound	1832	Cận gốc trùng nghiệm → mô hình tốt
Cuts	0	Không cần ràng buộc bổ sung → dữ liệu không gây khó solver
Time (ms)	1020	Thời gian ~1 giây → đáp ứng yêu cầu thực nghiệm
Solution	Chuỗi SKU	Nghiệm rõ ràng, tái lập được, phù hợp với ràng buộc replenishment
Solution		
19 – 31 – 4 – 36 – 1 – 39 – 9 – 16 – 18 – 37 – 8 – 14 – 2 – 33 – 3 – 23 – 28 – 38 – 26 – 27 – 11 – 13 – 15 – 25 – 5 – 6 – 17 – 20 – 22 – 40 – 21 – 24 – 7 – 10 – 30 – 35 – 32 – 34 – 12 – 29		

Đánh giá tổng hợp theo các tiêu chí

- Xét tổng thể các chỉ số thu được cho instance SLAPRP_Guo_small_O50_alpha0.2_v1.txt, có thể nhận thấy rằng kết quả đạt tính hợp lý cao khi giá trị nghiệm tốt nhất (UB) trùng với cận dưới (LB), dẫn đến khoảng cách tối ưu bằng 0. Điều này cho thấy mô hình được xây dựng đúng và thuật toán đã tìm được nghiệm tối ưu toàn cục, không tồn tại sai lệch về mặt toán học.
- Về tính ổn định, thuật toán thể hiện hành vi rất tốt khi bài toán được giải ngay tại node gốc của quá trình Branch-and-Bound, không cần sinh thêm ràng buộc cắt và không phát sinh số lượng lớn node tìm kiếm.
- Thời gian giải chỉ khoảng một giây cho thấy quá trình tối ưu diễn ra trơn tru, không có dấu hiệu dao động hay bất ổn trong tính toán.
- Xét về mức độ đáp ứng yêu cầu, kết quả hoàn toàn thỏa mãn các tiêu chí đặt ra cho bài toán SLAPRP trong bối cảnh bổ sung hàng tồn kho.
- Nghiệm đạt tối ưu toàn cục, thời gian tính toán nằm trong ngưỡng chấp nhận được, và cấu hình kho thu được tuân thủ đầy đủ các ràng buộc về vị trí lưu trữ cố định. Điều này chứng tỏ phương pháp đề xuất không chỉ hiệu quả về mặt chất lượng nghiệm mà còn phù hợp để áp dụng cho các instance có quy mô tương tự.

Tóm lại, các chỉ số đánh giá cho instance này cho thấy phương pháp giải là hợp lý, ổn định và đáp ứng tốt các yêu cầu của bài toán, đồng thời minh họa rõ lợi ích của biến thể replenishment trong việc giảm độ phức tạp mà vẫn đảm bảo hiệu quả tối ưu.

4.4 So sánh các phương pháp giải, thuật toán, môi trường và ngôn ngữ cho bài toán SLAPRP

Bài toán gán vị trí lưu trữ và định tuyến người lấy hàng (SLAPRP) là một bài toán tối ưu tổ hợp tích hợp, có độ phức tạp NP-hard mạnh. Do đó, trong thực tiễn và nghiên cứu, nhiều phương pháp giải khác nhau đã được đề xuất, từ heuristic đơn giản đến các thuật toán tối ưu chính xác. Mục này trình bày so sánh các phương pháp giải quyết bài toán SLAPRP theo bốn khía cạnh chính: cách tiếp cận thuật toán, mức độ tối ưu, môi trường triển khai, và ngôn ngữ lập trình, nhằm làm rõ tính phù hợp của phương pháp được lựa chọn trong dự án.

4.4.1 So sánh theo cách tiếp cận thuật toán

(1) Phương pháp heuristic và rule-based

Các phương pháp heuristic truyền thống thường giải bài toán gán vị trí lưu trữ (SLAP) và định tuyến người lấy hàng (PRP) một cách tách rời hoặc tuần tự, sử dụng các quy tắc đơn giản như:

- Dedicated / class-based storage cho SLAP
- Các routing policy như *S-shape*, *Return*, *Midpoint*, *Largest Gap* cho PRP

Ưu điểm của nhóm phương pháp này là:

- Dễ cài đặt
- Thời gian tính toán nhanh
- Phù hợp với hệ thống kho quy mô lớn trong vận hành thời gian thực

Tuy nhiên, nhược điểm lớn là:

- Không đảm bảo tối ưu toàn cục
- Không xét đầy đủ mối quan hệ tương tác giữa lưu trữ và định tuyến
- Chất lượng nghiệm phụ thuộc mạnh vào cấu trúc kho và dữ liệu đầu vào

Do đó, các phương pháp heuristic chỉ phù hợp cho bài toán thực tế yêu cầu phản hồi nhanh, nhưng không thích hợp cho mục tiêu nghiên cứu tối ưu chính xác.

(2) Phương pháp MIP với mô hình compact

Một số nghiên cứu tiếp cận SLAPRP bằng cách xây dựng mô hình quy hoạch nguyên hỗn hợp (MIP) dạng compact, trong đó:

- Biến gán SKU–location và biến định tuyến cùng xuất hiện trong một mô hình
- Định tuyến thường được mô tả bằng các biến cung và ràng buộc loại subtour (MTZ)

Ưu điểm:

- Mô hình rõ ràng, dễ hiểu về mặt toán học
- Có thể sử dụng trực tiếp các solver thương mại như CPLEX, Gurobi

Nhược điểm:

- Số biến và ràng buộc tăng rất nhanh theo quy mô kho và số đơn hàng
- LP relaxation yếu → thời gian giải tăng mạnh
- Chỉ giải được các instance rất nhỏ trong thực tế

Mô hình compact không phù hợp để giải các instance SLAPRP có kích thước trung bình trở lên khi yêu cầu định tuyến tối ưu .

(3) Phương pháp tối ưu chính xác với Branch-Cut-and-Price

Phương pháp Branch-Cut-and-Price (BCP) là một hướng tiếp cận nâng cao, kết hợp:

- Dantzig–Wolfe decomposition để tách phần định tuyến
- Column Generation để sinh động các lộ trình cần thiết
- Cutting Planes để tăng cường liên kết giữa lưu trữ và định tuyến
- Branch-and-Bound để đảm bảo nghiệm nguyên

Ưu điểm nổi bật:

- Giải quyết bài toán SLAPRP một cách tích hợp hoàn toàn

- Xử lý hiệu quả không gian nghiệm rất lớn
- Định tuyến được giải tối ưu chính xác, không phụ thuộc heuristic
- Phù hợp với các kho e-commerce có cấu trúc phức tạp và dữ liệu động

Nhược điểm:

- Cài đặt phức tạp
- Yêu cầu kiến thức sâu về tối ưu tổ hợp và quy hoạch tuyến tính
- Thời gian tính toán vẫn lớn với các instance rất lớn

Tuy nhiên, xét trên mục tiêu nghiên cứu và đánh giá tối ưu, BCP là phương pháp phù hợp nhất cho bài toán SLAPRP hiện nay.

4.4.2 So sánh theo mức độ tối ưu và chất lượng nghiệm

Phương pháp	**Mức độ tối ưu**	**Chất lượng nghiệm**
Heuristic / rule-based	Gần đúng	Phụ thuộc dữ liệu
MIP compact	Tối ưu (instance nhỏ)	Không ổn định
Branch-Cut-and-Price	Tối ưu toàn cục	Ổn định, nhất quán

Phương pháp được sử dụng trong dự án đảm bảo nghiệm tối ưu toàn cục, đồng thời cho phép đánh giá chính xác hiệu quả của các quyết định gán vị trí và định tuyến.

4.4.3 So sánh môi trường và công cụ triển khai

Việc lựa chọn môi trường và công cụ triển khai có ảnh hưởng trực tiếp đến khả năng mô hình hóa, hiệu quả tính toán và mức độ mở rộng của thuật toán giải bài toán SLAPRP. Do bài toán có cấu trúc lớn, tích hợp nhiều quyết định và sử dụng kỹ thuật phân rã nâng cao, các công cụ truyền thống giải MIP không phải lúc nào cũng đáp ứng tốt yêu cầu.

- Solver MIP thương mại (CPLEX, Gurobi)

Các solver thương mại như IBM ILOG CPLEX và Gurobi là những công cụ mạnh trong việc giải các mô hình quy hoạch nguyên hỗn hợp dạng compact.

Ưu điểm:

- Khả năng giải MIP rất tốt cho các mô hình vừa và nhỏ
- Tích hợp sẵn Branch-and-Bound, Cutting Planes, Presolve
- Dễ sử dụng thông qua các API (Python, C++, Java)

Hạn chế trong bài toán SLAPRP:

- Mô hình compact của SLAPRP có số lượng biến và ràng buộc tăng theo cấp số mũ
- LP relaxation yếu → số node trong cây Branch-and-Bound lớn
- Không hỗ trợ trực tiếp Column Generation và Branch-and-Price

- Việc cài đặt pricing problem (ESPPRC) trong solver thương mại là phức tạp và kém linh hoạt

Thực nghiệm trong nhiều nghiên cứu cho thấy các solver này chỉ giải được các instance rất nhỏ của SLAPRP khi yêu cầu định tuyến tối ưu, do đó không phù hợp cho mục tiêu nghiên cứu tổng quát.

- Môi trường triển khai Branch-Cut-and-Price

Đối với các bài toán có cấu trúc phân rã rõ ràng như SLAPRP, môi trường triển khai chuyên biệt cho Branch-Cut-and-Price (BCP) mang lại hiệu quả vượt trội.

Đặc điểm của môi trường BCP:

- Cho phép xây dựng mô hình master problem và pricing problem tách biệt
- Hỗ trợ Column Generation một cách tự nhiên
- Linh hoạt trong việc:
    - Thiết kế bài toán con (ESPPRC)
    - Thêm các bất đẳng thức tăng cường (SL, SL-1)
    - Điều khiển chiến lược sinh cột, sinh cắt và phân nhánh

Ưu điểm trong SLAPRP:

- Chỉ sinh ra các lộ trình (route) thực sự cần thiết
- Tránh hiện tượng bùng nổ số biến
- Tận dụng cấu trúc đồ thị của bài toán định tuyến
- Dễ dàng mở rộng sang các biến thể khác (routing policy, layout kho, replenishment)

Nhược điểm:

- Thời gian phát triển hệ thống dài hơn
- Yêu cầu kiến thức sâu về tối ưu tổ hợp và quy hoạch tuyến tính
- Khó triển khai trong môi trường sản xuất thời gian thực

Tuy nhiên, đối với bài toán nghiên cứu và đánh giá thuật toán, môi trường BCP là lựa chọn phù hợp và hiệu quả nhất.

So sánh tổng hợp môi trường triển khai

Tiêu chí	**Solver MIP truyền thống**	**Branch-Cut-and-Price**
Hỗ trợ Column Generation	Không trực tiếp	Có
Xử lý không gian nghiệm lớn	Kém	Tốt
Linh hoạt mở rộng mô hình	Thấp	Cao

Phù hợp SLAPRP tích hợp	Không	Rất phù hợp
Mục tiêu nghiên cứu học thuật	Hạn chế	Rất phù hợp

Từ bảng so sánh trên, có thể thấy rằng việc lựa chọn môi trường triển khai Branch-Cut-and-Price là phù hợp với đặc thù bài toán SLAPRP và mục tiêu của dự án.

4.4.4 Ngôn ngữ lập trình Julia trong triển khai thuật toán

Ngôn ngữ lập trình đóng vai trò quan trọng trong việc hiện thực hóa thuật toán Branch-Cut-and-Price, đặc biệt khi cần cân bằng giữa hiệu năng tính toán, khả năng mô hình hóa, và tính linh hoạt trong phát triển. Trong dự án này, nhóm lựa chọn Julia làm ngôn ngữ triển khai chính.

Julia được thiết kế hướng đến các bài toán tính toán khoa học và tối ưu, với các đặc điểm nổi bật:

- Hiệu năng cao, gần với C/C++ nhờ cơ chế JIT compilation
- Cú pháp gần gũi với toán học, dễ biểu diễn mô hình tối ưu
- Hỗ trợ tốt cho lập trình song song và xử lý dữ liệu lớn
- Thư viện tối ưu mạnh mẽ, đặc biệt là JuMP

JuMP là một DSL (Domain-Specific Language) cho tối ưu hóa trong Julia, cho phép:

- Khai báo biến, ràng buộc, hàm mục tiêu một cách trực quan
- Dễ dàng kết nối với các solver LP/MIP (GLPK, CPLEX, Gurobi)
- Linh hoạt trong việc cập nhật mô hình trong quá trình column generation

Trong triển khai Branch-Cut-and-Price:

- Master problem được mô hình hóa bằng JuMP
- Các cột (route) mới được thêm động vào mô hình
- Dual variables được trích xuất trực tiếp để giải pricing problem

Điều này giúp quá trình cài đặt column generation trở nên rõ ràng, dễ kiểm soát và ít lỗi hơn so với các ngôn ngữ truyền thống.

- So sánh Julia với các ngôn ngữ khác

So với Python:

- Julia nhanh hơn đáng kể trong các vòng lặp tính toán nặng
- Tránh overhead của interpreter
- Python phù hợp prototype, nhưng kém hiệu quả cho BCP quy mô lớn

So với C/C++:

- Julia dễ viết và bảo trì hơn

- Không cần quản lý bộ nhớ thủ công
- Thời gian phát triển nhanh hơn
- C/C++ vẫn vượt trội về hiệu năng thuần, nhưng chi phí phát triển cao

So với MATLAB:

- Julia là mã nguồn mở
- Hiệu năng tương đương hoặc tốt hơn
- Mạnh hơn trong lập trình cấu trúc và thuật toán phức tạp

- Đánh giá tổng hợp vai trò của Julia trong dự án

Việc lựa chọn Julia giúp:

- Hiện thực hóa thuật toán Branch-Cut-and-Price một cách hiệu quả
- Dễ dàng thử nghiệm nhiều chiến lược sinh cột và cắt
- Tăng khả năng mở rộng và tái sử dụng mã nguồn
- Phù hợp với định hướng nghiên cứu và học thuật

Do đó, Julia là một lựa chọn phù hợp và có cơ sở kỹ thuật rõ ràng cho việc triển khai bài toán SLAPRP trong dự án này.

KẾT LUẬN

Báo cáo đã trình bày một cách hệ thống quá trình nghiên cứu và triển khai bài toán gán vị trí lưu trữ và định tuyến người lấy hàng (Storage Location Assignment and Picker Routing Problem – SLAPRP) trong bối cảnh kho hàng hiện đại, đặc biệt là kho phục vụ thương mại điện tử. Trọng tâm của báo cáo là tiếp cận bài toán theo hướng tối ưu tích hợp, thay vì giải quyết tuần tự hoặc dựa trên các chính sách heuristic như trong nhiều phương pháp truyền thống.

Trước hết, báo cáo đã phân tích bối cảnh thực tế và tính cấp thiết của bài toán SLAPRP, làm rõ mối liên hệ chặt chẽ giữa quyết định gán vị trí lưu trữ và quyết định định tuyến người lấy hàng. Từ đó, mô hình hệ thống kho hàng được xây dựng dưới dạng đồ thị có hướng, cho phép mô tả linh hoạt nhiều cấu trúc kho và phản ánh sát hoạt động lấy hàng trong thực tế. Dựa trên mô hình này, bài toán SLAPRP được mô hình hóa dưới dạng bài toán tối ưu tổ hợp với các biến quyết định, hàm mục tiêu và hệ ràng buộc rõ ràng.

Tiếp theo, báo cáo đã trình bày chi tiết hai dạng mô hình toán học: mô hình compact và mô hình mở rộng dựa trên phân rã Dantzig–Wolfe. Trong đó, mô hình mở rộng khắc phục được hạn chế về chất lượng nghiệm LP của mô hình compact bằng cách biểu diễn các quyết định định tuyến thông qua các biến lộ trình. Trên cơ sở đó, khung thuật toán Branch-Cut-and-Price được lựa chọn và phân tích như một phương pháp giải phù hợp cho bài toán SLAPRP có quy mô lớn và cấu trúc phức tạp. Các thành phần chính của thuật toán như Column Generation, Cutting Planes và Branch-and-Bound cũng được làm rõ vai trò và cách phối hợp trong quá trình giải.

Cuối cùng, báo cáo đã triển khai thực nghiệm thuật toán bằng ngôn ngữ Julia trên các bộ dữ liệu mô phỏng, qua đó đánh giá hiệu quả của mô hình và phương pháp giải. Kết quả cho thấy cách tiếp cận tích hợp và tối ưu chính xác giúp cải thiện đáng kể chất lượng nghiệm so với các phương pháp giải rời rạc. Nhìn chung, báo cáo đã hoàn thành các mục tiêu đề ra, đồng thời tạo nền tảng cho các nghiên cứu mở rộng trong tương lai, như mở rộng quy mô bài toán, bổ sung ràng buộc thực tế hoặc so sánh với các phương pháp heuristic và metaheuristic khác.

Tài liệu tham khảo

[1] T. Prunet, N. Absi, and D. Cattaruzza, *"The Storage Location Assignment and Picker Routing Problem: A Generic Branch-Cut-and-Price Algorithm,"* arXiv preprint arXiv:2407.13570, Jul. 2024.

[2] G. Desaulniers, J. Desrosiers, and M. M. Solomon, *Column Generation*, New York, NY, USA: Springer, 2005.

[3] M. W. P. Savelsbergh, *"Branch-and-price: Theory and applications,"* European Journal of Operational Research, vol. 153, no. 3, pp. 451–466, 2004.

[4] J. Desrosiers, Y. Dumas, M. M. Solomon, and F. Soumis, *"Time constrained routing and scheduling,"* in *Network Routing*, Handbooks in Operations Research and Management Science, vol. 8, M. O. Ball et al., Eds., Amsterdam, The Netherlands: Elsevier, 1995, pp. 35–139.

[5] R. de Koster, T. Le-Duc, and K. J. Roodbergen, *"Design and control of warehouse order picking: A literature review,"* European Journal of Operational Research, vol. 182, no. 2, pp. 481–501, 2007.