

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG



ĐỒ ÁN TỐT NGHIỆP
NGÀNH CÔNG NGHỆ THÔNG TIN

Phát triển thuật toán tìm kiếm cục bộ cho bài toán
lập lô trình giao hàng kết hợp xe tải và thiết bị bay
drone

Sinh viên thực hiện :

Nguyễn Tuấn Đạt 20130856 CNTT2.02-K58

Giảng viên hướng dẫn :

TS. Phạm Quang Dũng

Hà Nội 12-2017

Phiếu giao nhiệm vụ đồ án tốt nghiệp

Thông tin về sinh viên

- Họ và tên : Nguyễn Tuấn Đạt
- Email : tuandat95cbn@gmail.com
- Điện thoại liên lạc : 01643995246
- Lớp : CNTT2.02 K58
- Hệ đào tạo : Đại học chính quy
- Đồ án tốt nghiệp được thực hiện tại : Bộ môn khoa học máy tính - Viện công nghệ thông tin và truyền thông.
- Thời gian làm đồ án tốt nghiệp : Từ ngày 21/8/2017 đến 22/12/2017.

Mục đích nội dung của đồ án tốt nghiệp

Phát triển thuật toán tìm kiếm cục bộ cho bài toán lập lộ trình giao hàng kết hợp xe tải và thiết bị bay drone.

Các nhiệm vụ cụ thể của đồ án tốt nghiệp

1. Nghiên cứu bài toán lập lộ trình giao hàng kết hợp xe tải và một thiết bị bay drone.
2. Phát triển thuật toán giải quyết bài toán lập lộ trình giao hàng kết hợp xe tải và nhiều thiết bị bay drone.
3. Thực nghiệm và đánh giá hai bài toán.
4. Xây dựng chương trình ứng dụng hỗ trợ lập lộ trình giao hàng kết hợp xe tải và thiết bị bay drone.

Lời cam đoan của sinh viên

Tôi - Nguyễn Tuấn Đạt - cam kết đồ án tốt nghiệp là sản phẩm của bản thân tôi dưới sự hướng dẫn của TS. Phạm Quang Dũng.

Các kết quả trong đồ án là trung thực, không phải sao chép toàn văn của bất kì công trình nào khác.

Hà Nội, ngày 22 tháng 12 năm 2017
Sinh viên

Nguyễn Tuấn Đạt

Xác nhận của giáo viên hướng dẫn về mức độ hoàn thành và cho phép bảo vệ:

Hà Nội, ngày 22 tháng 12 năm 2017
Giảng viên hướng dẫn

TS. Phạm Quang Dũng

Lời cảm ơn

Trước khi trình bày nội dung của đồ án này, tôi xin được gửi lời cảm ơn sâu sắc và chân thành nhất đến TS. Phạm Quang Dũng, người đã tận tình hướng dẫn tôi trong suốt quá trình thực hiện đồ án này cũng như những năm tháng học tại trường Đại học Bách Khoa Hà Nội. Đồng thời tôi cũng xin bày tỏ lòng biết ơn đến các thầy cô trưởng Đại học Bách Khoa Hà Nội, Viện công nghệ thông tin và truyền thông, đặc biệt là các thầy cô bộ môn Khoa học máy tính đã tận tình chỉ dạy cho tôi trong những năm tháng học tập ở trường.

Tôi cũng xin bày tỏ lòng biết ơn đến các anh chị ở công ty cổ phần OLBIUS đã nhiệt tình giúp đỡ tôi phần công nghệ cho đồ án này. Cùng với đó, tôi cũng xin cảm ơn các anh chị ở lab MSO đã giúp đỡ tôi trong những năm tháng tôi sinh hoạt ở lab.

Cuối cùng tôi xin gửi lời cảm ơn đến gia đình, bạn bè đã luôn ở bên tôi, động viên và giúp đỡ tôi trong suốt quá trình học tập và thực hiện đồ án tốt nghiệp.

Mục lục

Phiếu giao nhiệm vụ đồ án tốt nghiệp	1
Lời cảm ơn	3
Bảng chữ viết tắt	7
Danh sách bảng	8
Danh sách hình vẽ	9
Mở đầu	9
1 Cơ sở lý thuyết	11
1.1 Tối ưu hóa tổ hợp và ứng dụng	11
1.1.1 Tối ưu hóa tổ hợp	11
1.1.2 Ứng dụng	14
1.2 Các hướng tiếp cận giải bài toán tối ưu hóa tổ hợp	16
1.2.1 Thuật toán giải đúng	16
1.2.2 Thuật toán giải gần đúng	18
1.3 Công nghệ ofbiz	19
2 Bài toán lập lộ trình giao hàng kết hợp một xe tải với một thiết bị bay cùng các nghiên cứu liên quan	20
2.1 Bài toán lập lộ trình vận tải giao hàng kết hợp xe tải và một thiết bị bay .	20
2.1.1 Mô tả bài toán	20
2.1.2 Các nghiên cứu liên quan	21
2.1.3 Mô hình bài toán min-cost TSPD	22
2.1.4 Ràng buộc	25
2.1.5 Hàm mục tiêu	25
2.1.6 Mô hình Mixed Integer Programming	25

2.2	Thuật toán GRASP	27
2.2.1	Thuật toán phân tách	28
2.2.2	Các toán tử tìm kiếm cục bộ	32
2.3	Thuật toán tìm kiếm cục bộ	34
3	Đề xuất thuật toán tìm kiếm cục bộ giải bài toán lập lô trình vận tải giao hàng kết hợp một xe tải và nhiều thiết bị bay	40
3.1	Bài toán lập lô trình giao hàng kết hợp xe tải và nhiều thiết bị bay	40
3.2	Phát biểu bài toán	40
3.3	Thuật toán tìm kiếm cục bộ giải quyết bài toán lập lô trình giao hàng kết hợp một xe tải và nhiều thiết bị bay	41
3.3.1	Toán tử move	41
3.3.2	Thuật toán tìm kiếm cục bộ	43
4	Thử nghiệm và đánh giá	47
4.1	Dữ liệu	47
4.2	Kết quả thử nghiệm	48
5	Thiết kế và xây dựng chương trình ứng dụng	54
5.1	Công nghệ sử dụng	54
5.1.1	Công nghệ ofbiz	54
5.1.2	GoogleMap API	55
5.1.3	Bootstrap	55
5.2	Sơ đồ use-case	56
5.3	Cơ sở dữ liệu	57
5.4	Các màn hình	57
Kết luận và hướng phát triển		57
Tài liệu tham khảo		63

Bảng chữ viết tắt

Chữ viết tắt	Tên đầy đủ
ERP	Hệ thống quản trị doanh nghiệp - Enterprise Resource Planning
JSP	Java Server Pages
FTL	FreeMarker Template Language
SOAP	Simple object access protocol (một dạng web service)
GRASP	A Greedy Randomized Adaptive Search Procedure
TSP	Traveling salesman problem
TSPD	Traveling salesman problem with Drone
TSPkD	Traveling salesman problem with k Drones
CSP	Bài toán thỏa mãn ràng buộc - Constraint Satisfaction Problem
DD	Một chuyến giao hàng bởi drone - Drone delivery
TD	Một hành trình giao hàng bởi xe tải - Truck delivery
MIP	Mixed Integer Programming
FSTSP	Flying Sidekick Traveling Salesman Problem
GTSP	Generalized Traveling Salesman Problem
HDP	Heterogeneous Delivery Problem

Danh sách bảng

4.1	Bảng kết quả bộ dữ liệu 2 ứng với 1, 2, 3, 4 drones	48
4.2	Bảng kết quả bộ dữ liệu 2 ứng với 1, 2, 3, 4 drones	49
4.3	Bảng kết quả bộ dữ liệu 2 ứng <i>move 1, 2, 3, 4 point</i> cho bài toán min-cost TSPkD 2 drones	51

Danh sách hình vẽ

2.1	Lộ trình bài toán TSP và lộ trình bài toán TSPD tương ứng [9]	24
2.2	Auxiliary graph cho lộ trình TSP được trình bày trong hình 2.1 [9]	32
2.3	Toán tử <i>truck_relocation</i>	33
2.4	Toán tử <i>drone_relocation</i>	33
2.5	Toán tử <i>drone_removal</i>	34
2.6	Toán tử <i>two_exchange</i>	34
2.7	Mô phỏng thao tác <i>relocateAsTruck</i>	38
2.8	Mô phỏng thao tác <i>relocateAsDrone</i>	39
3.1	Ví dụ cho thao tác <i>relocate_D</i> sang bài toán min-cost Traveling salesman problem with k Drones (TSPkD)	42
3.2	Chọn 3 điểm liên tiếp – thao tác move-3-point	43
3.3	Xóa bỏ 1 điểm vừa chọn ra khỏi lộ trình – thao tác move-3-point	43
3.4	Chuyển các nút chọn thành DD – thao tác move-3-point	43
4.1	Biểu đồ thể hiện kết quả thuật toán với 1, 2, 3, 4 drone- Bộ dữ liệu 1	50
4.2	Biểu đồ thể hiện kết quả thuật toán với 1,2,3,4 drone - Bộ dữ liệu 2	51
4.3	Biểu đồ thể hiện kết quả thuật toán với <i>move – 1, 2, 3, 4 – point</i> - Bộ dữ liệu 2	53
5.1	Sơ đồ use-case của ứng dụng	56
5.2	Thiết kế cơ sở dữ liệu của ứng dụng	58
5.3	Màn hình thay đổi ngôn ngữ sang tiếng Anh	59
5.4	Màn hình tạo dữ liệu lập lịch	59
5.5	Màn hình chỉnh sửa tập dữ liệu	60
5.6	Màn hình lập lịch	60
5.7	Màn hình kết quả với một drone	61
5.8	Màn hình kết quả với bốn drone	61

Giới thiệu chung

Bài toán lập lịch vận chuyển hàng hóa từ lâu đã là lớp bài toán tối ưu hóa quan trọng và có tính thực tiễn cao. Công nghệ ngày càng phát triển, đặc biệt gần đây, phương tiện bay không người lái trở phổ biến với nhiều mẫu mã đặc trưng cho các mục đích sử dụng khác nhau. Thiết bị bay không người lái (hay còn gọi là drone) đã và đang được nghiên cứu sử dụng trong ngành công nghiệp vận chuyển hàng hóa. Với công nghệ hiện nay, drone có thể nâng những vật có khối lượng hàng chục kilogram và quãng đường bay có thể đạt đến hàng chục kilometers. Những đặc tính trên khiến thiết bị bay không người lái có thể là xu hướng trong nhiều năm tới đây.

Trong thời buổi công nghệ thông tin phát triển rộng khắp như hiện nay, bài toán lập lịch vận chuyển hàng hóa gần như là một phần không thể thiếu trong hệ thống Enterprise Resource Planning - Quản trị doanh nghiệp (ERP) của các công ty vận tải hàng đầu thế giới. Chính vì vậy trong đồ án này, chúng tôi đã phát triển và cài đặt mô hình thuật toán tìm kiếm cục bộ cho bài toán lập lộ trình giao hàng kết hợp xe tải và thiết bị bay không người lái ứng dụng vào công nghệ Ofbiz dưới dạng web service. Ofbiz là một ứng dụng mã nguồn mở về ERP được Apache phát triển từ năm 2001. Ofbiz cho phép lập trình viên có thể can thiệp, thay đổi hệ thống linh hoạt cũng như có khả năng phát triển một ứng dụng độc lập hoàn chỉnh.

Nội dung chính của đồ án bao gồm 6 chương:

- **Chương 1:** Cơ sở lý thuyết.
- **Chương 2:** Bài toán lập lộ trình giao hàng kết hợp một xe tải với một thiết bị bay cùng các nghiên cứu liên quan.
- **Chương 3:** Đề xuất thuật toán tìm kiếm cục bộ giải bài toán lập lộ trình vận tải giao hàng kết hợp một xe tải và nhiều thiết bị bay.
- **Chương 4:** Thủ nghiệm và đánh giá.
- **Chương 5:** Thiết kế chương trình ứng dụng.
- **Chương 6:** Kết luận và hướng phát triển.

Chương 1

Cơ sở lý thuyết

Trong chương này chúng tôi sẽ trình bày cơ sở lý thuyết chung cho bài toán tối ưu hóa tổ hợp, tối ưu hóa thỏa mãn ràng buộc. Các định nghĩa bài toán tối ưu hóa tổ hợp, bài toán thỏa mãn ràng buộc và ứng dụng được trình bày trong phần 1.1. Ở phần 1.2 chúng tôi trình bày hai hướng tiếp cận giải bài toán thỏa mãn ràng buộc là thuật toán giải đúng và thuật toán giải gần đúng. Trong phần 1.3, chúng tôi giới thiệu tổng quan về công nghệ chính xây dựng module ứng dụng cho đồ án này là công nghệ Ofbiz. Các tài liệu tham khảo chúng tôi sử dụng trong phần này là [1], [2], [4], [6], [7], [23].

1.1 Tối ưu hóa tổ hợp và ứng dụng

1.1.1 Tối ưu hóa tổ hợp

Bài toán tối ưu hóa tổ hợp (Combinatorial Optimization Problem) [6], [7] dưới dạng tổng quát có thể được phát biểu như sau:

Cho một tập hữu hạn $N = \{1, \dots, n\}$, trọng số c_j , $j \in N$, và một tập F là một tập con của N . Bài toán yêu cầu tìm một tập con với tổng trọng số nhỏ nhất :

$$\min_{S \subseteq F} \left\{ \sum_{j \in S} c_j : S \in F \right\}$$

Bài toán thỏa mãn ràng buộc

Một ràng buộc [2] là một quan hệ trên trên một tập các biến. Mỗi biến có một tập hữu hạn các giá trị có thể nhận gọi là miền giá trị. Một ràng buộc có thể được biểu diễn bằng một biểu thức toán học hoặc một bảng liệt kê các giá trị phù hợp cho các biến. Một bài toán thỏa mãn ràng buộc - Constraint Satisfaction Problem (CSP) bao gồm :

1. Một tập các **biến** X.
2. Miền giá trị (một tập hữu hạn các giá trị) cho mỗi biến D.
3. Một tập hữu hạn các ràng buộc C.

Một trạng thái của bài toán được định nghĩa bởi một phép gán các giá trị cho tất cả các biến. Một trạng thái được gọi là lời giải khi nó thỏa mãn tất cả các ràng buộc trong C . Một vài bài toán CSP còn yêu cầu tối ưu một hoặc nhiều hàm mục tiêu.

Chúng tôi trình bày hai ví dụ điển hình cho bài toán thỏa mãn ràng buộc là bài toán N con hậu và bài toán phân bổ chương trình học.

N con hậu (N-QUEEN)

Bài toán yêu cầu xếp n quân hậu trên bàn cờ vua $n \times n$ sao cho không có hai quân hậu bất kì nào ăn được nhau.

Mô hình toán học

Biến Gọi $X = \{x_1, \dots, x_n\}$ với x_i có giá trị là cột của con hậu hàng thứ i .

Miền giá trị Miền giá trị của x_i là $D(x_i) = \{1, \dots, n\}$, $\forall i = 1, 2, \dots, n$

	1	2	3	4
1		X		
2				X
3	X			
4			X	

VD: Với 4 con hậu được xếp như hình bên, trạng thái của các biến sẽ là:

$$x_1 = 2, x_2 = 4, x_3 = 1, x_4 = 3$$

Các ràng buộc

- Các con hậu không được ở trên cùng một cột.

$$x_i \neq x_j, \quad \forall i \neq j, \quad i, j \in \{1, \dots, n\}$$

- Các con hậu không cùng trên một hàng chéo.

$$x_i - x_j \neq i - j, \quad \forall i \neq j, \quad i, j \in \{1, \dots, n\}$$

$$x_j - x_i \neq i - j, \quad \forall i \neq j, \quad i, j \in \{1, \dots, n\}$$

Bài toán phân bổ môn học (Balanced Academic Curriculum Problem)

Bài toán phân bổ môn học [1] yêu cầu thiết kế một chương trình học, được định nghĩa bằng việc sắp xếp các môn vào các kỳ học sao cho thỏa mãn các quy định và cân bằng về số tín chỉ cũng như số môn giữa các học kì. Một chương trình học phải thỏa mãn các quy định sau:

- Một chương trình học định nghĩa bởi một tập các môn học và tập các mối quan hệ giữa chúng.
- Số lượng kì: Một chương trình học chỉ được phép kéo dài trong một số kì nhất định.
- Định lượng: Mỗi môn học có một chỉ số tín chỉ khi mà hoàn thành lớp mỗi sinh viên nhận được tích lũy.
- Lớp tiên quyết: Có một số môn cần có những lớp khác học trước.
- Giới hạn tín chỉ: Trong một kì sinh viên không được học quá ít, hoặc quá nhiều tín chỉ.
- Số lớp tối thiểu: Số lượng lớp tối thiểu trong một kì của sinh viên.
- Số lớp tối đa: Số lượng lớp tối đa trong một kì của sinh viên.

Mục đích của vấn đề là xếp các môn vào các kì sao cho thỏa mãn tất cả các quy định trên và chênh lệch số lượng tín chỉ trong các kì là nhỏ nhất.

Mô hình toán học

Đầu vào

- $nPeriods$ là số lượng kì cần xếp lịch.
- $nCourses$ là số lượng môn cần xếp lịch.
- $prereq$ là một danh sách các phần tử có dạng $\langle i, j \rangle$ thể hiện môn i cần phải được học trước môn j .
- $credit(j)$ là số lượng tín chỉ của môn j .
- a là số lượng tín chỉ ít nhất trong một kì.
- b là số lượng tín chỉ nhiều nhất trong một kì.
- c là số lượng môn ít nhất trong một kì.
- d là số lượng môn nhiều nhất trong một kì.

Biến Biến quyết định mô hình hóa bài toán là

$$X = \{X_{ij} \mid i \in \{1, \dots, nPeriods\}, j \in \{1, \dots, nCourses\}\}$$

Miền giá trị Miền giá trị của X_{ij} là $D(X_{ij}) = \{0, 1\}$, trong đó :

- $X_{i,j} = 1$ nếu lớp j được đặt trong kì i .
- $X_{i,j} = 0$ nếu ngược lại.

Các ràng buộc Ta định nghĩa :

$$load[i] = \sum_{j \in \{1, \dots, nCourses\}} credit(j) * X_{i,j}, \forall i \in \{1, \dots, nPeriods\}$$

- Ràng buộc cho biết mỗi môn chỉ được xếp trong đúng một học kì

$$\sum_{i \in \{1, \dots, nPeriods\}} X_{ij} = 1, \forall j \in \{1, \dots, nCourses\}$$

- Ràng buộc thể hiện điều kiện học phần học trước

$$\forall \langle i, j \rangle \in prereq, \forall k \in \{1, \dots, nPeriods\}$$

$$X_{kj} \leq \sum_{r=1}^{k-1} X_{ri}$$

- Ràng buộc về giới hạn số tín chỉ trong một kì

$$a \leq load[i] \leq b, \forall i \in \{1, \dots, nPeriods\}$$

- Ràng buộc về giới hạn số lớp học trong một kì

$$c \leq \sum_{j \in \{1, \dots, nCourses\}} X_{ij} \leq d, \forall i \in \{1, \dots, nPeriods\}$$

1.1.2 Ứng dụng

Trong phần này chúng tôi trình bày một vài bài toán tối ưu hóa tổ hợp cụ thể. Những bài toán này đều là những bài toán thực tiễn được phát biểu dưới dạng bài toán tối ưu hóa tổ hợp, qua đó thể hiện được tính ứng dụng cao của bài toán tối ưu hóa tổ hợp trong môi trường thực tế.

Lập thời khóa biểu

Cho một tập môn học: $C = \{c_1, \dots, c_n\}$ cần phải xếp vào các tiết học trong các ngày trong tuần. Mỗi môn học c_i có $d(c_i)$ là số lượng tiết và $p(c_i)$ là giảng viên được yêu cầu dạy môn học. Mỗi ngày được chia thành 12 tiết bao gồm 6 tiết buổi sáng và 6 tiết buổi chiều. Bài toán yêu cầu xếp lịch cho tập các môn học C sao cho thỏa mãn các ràng buộc:

- Hai môn học bất kì thuộc cùng một giảng viên dạy thì không có thời gian dạy chồng lắp nhau.
- Mỗi môn học chỉ được học trong một buổi sáng hoặc chiều.

Phân công y tá

Đầu vào Bài toán yêu cầu xếp lịch ca trực cho n y tá trong m ngày. Mỗi ngày có k kíp trực khác nhau.

Ràng buộc

Ràng buộc cứng

- Mỗi y tá chỉ trực một ca trong ngày.
- Mỗi y tá chỉ trực một số lượng thời gian nhất định.
- Với một kíp số lượng lần trực của y tá không được vượt quá một đại lượng nhất định.
- Mỗi y tá không có đợt nghỉ quá ngắn.
- Số lượng ca trực liên tục của một y tá không được quá nhiều hoặc quá ít.
- Các y tá không được trực quá trong các đợt cuối tuần.
- Các y tá được đăng ký một số ngày họ được nghỉ.

Ràng buộc mềm

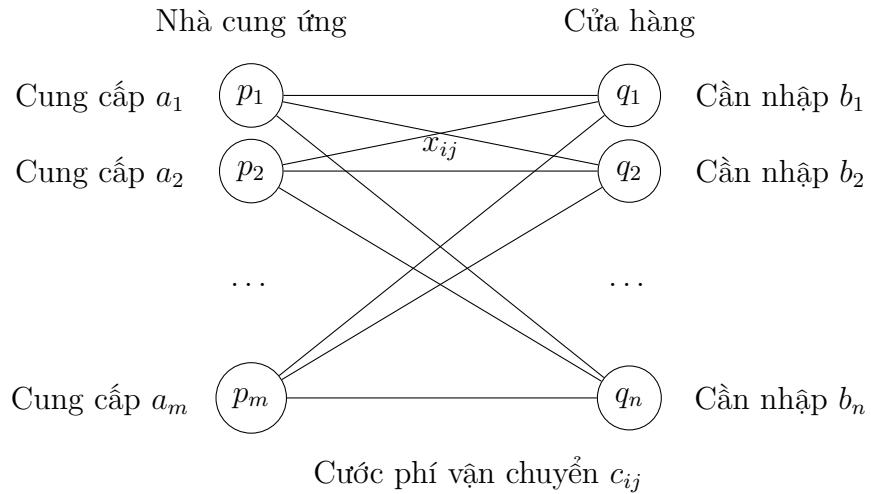
- Các y tá được phép yêu cầu nghỉ hoặc làm một số ca nào đó.
- Bệnh viện yêu cầu số lượng y tá trong mỗi ca trực.

Lập lô trình vận tải

Lớp bài toán lập lô trình vận tải [8] là một trong những lớp bài toán tối ưu quan trọng trong ngành công nghiệp giao thông vận tải. Xét bài toán tổng quát được định nghĩa như sau:

Xét một mô hình vận tải gồm m nhà cung ứng (hay còn gọi là điểm phát) và n cửa hàng (điểm thu). Bài toán yêu cầu vận chuyển hàng hóa từ các nhà cung ứng đến các cửa hàng sao cho chi phí di chuyển là nhỏ nhất.

- m nhà cung ứng.
- n cửa hàng.
- a_i lượng hàng nhà cung ứng i cung cấp.
- b_j lượng hàng cần nhập của cửa hàng j .
- c_{ij} cước phí vận chuyển một đơn vị hàng từ nhà cung cấp i đến cửa hàng j .



1.2 Các hướng tiếp cận giải bài toán tối ưu hóa tổ hợp

1.2.1 Thuật toán giải đúng

Trong phần này chúng tôi trình bày một vài phương pháp phổ biến được sử dụng giải đúng bài toán tối ưu hóa tổ hợp. Ngoài các phương pháp được nêu dưới đây còn rất nhiều các phương pháp khác được sử dụng để giải bài toán tối ưu hóa tổ hợp.

Quy hoạch động (Dynamic Programming)

Quy hoạch động [3] là một kỹ thuật được thiết kế khá vạn năng để phát triển cho nhiều lớp bài toán khác nhau. Kỹ thuật này là một trong những công cụ mạnh để thiết kế thuật toán giải bài toán tối ưu hóa.

Giống như chia để trị¹, quy hoạch động giải bài toán bằng việc tổng hợp lời giải từ các bài toán con. Với quy hoạch động các bài toán con được phép chồng lên nhau, tức là có thể có hai bài toán cha có chung một bài toán con. Với mỗi bài toán con được giải, chúng được lưu lại vào một bảng để tránh việc lặp lại tính toán.

Các bước để phát triển một thuật toán quy hoạch động là:

- Biểu diễn cấu trúc của một lời giải tối ưu cho bài toán đầu vào.
- Biểu diễn giá trị của một lời giải tối ưu cho bài toán đầu vào.
- Tính toán các lời giải tối ưu của các bài toán con, theo thứ tự từ dưới lên.
- Xây dựng lời giải tối ưu từ những thông tin đã tính toán.

¹Chia để trị là một thuật toán cơ bản trong ngành công nghệ thông tin, thuật toán thực hiện giải quyết một bài toán bằng việc giải các bài toán con của nó, sau đó tổng hợp các lời giải bài toán con thành lời giải bài toán cha. Việc tổng hợp này thường được thực hiện một cách đệ quy

Quy hoạch ràng buộc (Constraint Programming)

Quy hoạch ràng buộc [4] cũng là một phương pháp giải đúng bài toán tối ưu hóa tổ hợp. Chúng ta có thể xây dựng chiến lược tìm kiếm trong quy hoạch ràng buộc, đôi khi ta có thể thêm các nhánh cận để thuật toán hiệu quả hơn. Hai thành phần chính trong quy hoạch ràng buộc là:

- Loại bỏ giá trị thừa.
- Tìm kiếm quay lui.

Giải thuật tìm kiếm quay lui là giải thuật tìm kiếm phổ biến nhất được sử dụng trong bài toán CSP.

- Dựa trên giải thuật tìm kiếm theo chiều sâu.
- Mỗi lần gán, chỉ gán giá trị cho một biến.
- Sau mỗi phép gán giá trị cho một biến nào đó, kiểm tra các ràng buộc có được thỏa mãn bởi các biến đã được gán giá trị cho đến thời điểm hiện tại. Quay lui nếu có vi phạm ràng buộc.

Có hai yếu tố chính ảnh hưởng đến lời giải của phương pháp quay lui là:

- Thứ tự được xét của các biến.
- Với mỗi biến thứ tự được xét của các giá trị.

Giải thuật kiểm tra tiến là một giải thuật điển hình về loại bỏ giá trị thừa. Mục đích của kiểm tra tiến là tránh các thất bại trong tìm kiếm bằng việc kiểm tra trước các ràng buộc. Ý tưởng của thuật toán như sau:

- Ở mỗi bước gán giá trị, thực hiện theo dõi các giá trị hợp lệ (có thể được gán) đối với các biến chưa được gán giá trị.
- Loại bỏ hướng tìm kiếm khi bắt kì một biến chưa được gán giá trị mà không còn giá trị hợp lệ.

Quy hoạch nguyên tuyến tính (Linear Integer Programming)

Giả sử ta có một hàm

$$\max\{cx : Ax \leq b, x \geq 0\}$$

với A là một ma trận m hàng n cột, c là một vector hàng gồm n phần tử, b là một vector cột gồm m phần tử và x là một vector cột gồm n biến số chưa biết.

Với tất cả các biến phải là giá trị nguyên ta có mô hình bài toán quy hoạch nguyên tuyến tính[6] :

$$\max cx$$

$$Ax \leq b$$

$$x \geq 0 \text{ và nguyên}$$

Các thuật toán thường được sử dụng giải quyết bài toán quy hoạch nguyên tuyến tính là:

- Phương pháp nhánh cận.
- Cutting plane.
- Branch and cut.

1.2.2 Thuật toán giải gần đúng

Thuật toán giải đúng luôn cho lời giải chính xác nhưng bù lại có độ phức tạp tính toán tương đối cao thường là hàm mũ, vì vậy với những bài toán có kích thước đầu vào lớn thì việc sử dụng thuật toán giải đúng thường như là không thể. Trong khi đó các thuật toán giải gần đúng mặc dù không cho lời giải chính xác tuyệt đối nhưng thời gian thực thi lại hoàn toàn có thể chấp nhận được. Chính vì vậy, hiện nay với những bài toán tối ưu hóa tổ hợp có kích thước đầu vào lớn các thuật giải gần đúng thường được sử dụng. Trong phần này chúng tôi trình bày một vài các thuật toán gần đúng điển hình để giải bài toán tối ưu hóa tổ hợp đó là: thuật toán tham lam, thuật toán tìm kiếm cục bộ và thuật toán di truyền. Ngoài ra còn rất nhiều thuật toán gần đúng khác được sử dụng để giải bài toán tối ưu hóa tổ hợp.

Thuật toán tham lam (Greedy Algorithms)

Giải thuật cho các bài toán tối ưu hóa thường đi từng bước một, với mỗi bước có các tập lựa chọn. Cho nhiều vấn đề, quy hoạch động thường được sử dụng để xác định lựa chọn tốt nhất. Thuật toán tham lam [3] luôn luôn chọn một lựa chọn tốt nhất ở bước hiện tại. Việc lựa chọn này như đi theo một tối ưu cục bộ và ta luôn hi vọng nó sẽ dẫn đến tối ưu toàn cục. Chính vì vậy, thuật toán tham lam không phải lúc nào cũng đưa ra được lời giải tối ưu.

Thuật toán tham lam là một phương pháp khá mạnh và làm việc tốt cho các bài toán với kích thước lớn.

Thuật toán tìm kiếm cục bộ (Local Search Algorithms)

Tìm kiếm cục bộ [5] là một trong các mô hình được sử dụng để giải quyết các bài toán CSP. Nó cung cấp cơ sở cho một vài phương pháp thành công và linh hoạt nhất trong việc giải các bài toán lớn và khó bắt nguồn từ nhiều ứng dụng thực tế.

Ý tưởng của tìm kiếm cục bộ là: bắt đầu từ một lời giải được sinh ngẫu nhiên hoặc heuristic. Những lời giải này có thể là tối ưu cục bộ hoặc chưa hoàn chỉnh Thuật toán cải thiện các lời giải này từng bước bằng cách tạo ra các thay đổi nhỏ trong nó ở mỗi bước. Các phương thức tìm kiếm cục bộ khác nhau khác nhau trong cách cải tiến lời giải đạt được. Thuật toán sẽ dừng lại khi không thể cải tiến thêm được lời giải.

Một số phương pháp tìm kiếm cục bộ sử dụng yếu tố ngẫu nhiên để đảm bảo việc tìm kiếm không bị dừng lại ở tối ưu cục bộ. Các phương pháp đáng chú ý như stochastic

hill-climbing, simulated annealing, tabu search, dynamic local search Lớp các phương pháp như vậy được gọi là *metaheuristics*.

Thuật toán di truyền (Genetic Algorithms)

Thuật toán di truyền [5] được phát triển bởi J. Holland vào năm 1970s. Ý tưởng của thuật toán di truyền dựa trên thuyết tiến hóa của Darwin về cơ chế tiến hóa của quần thể trong tự nhiên. Các toán tử thường được sử dụng trong thuật toán di truyền là:

- Toán tử sinh sản: chọn lọc, tái sinh.
- Toán tử lai ghép.
- Toán tử đột biến.

Các bước thực hiện cơ bản của thuật toán di truyền như sau:

1. Khởi tạo quần thể ban đầu gồm các nhiễm sắc thể.
2. Lặp:
 - (a) Xác định giá trị hàm mục tiêu của các nhiễm sắc thể.
 - (b) Tạo nhiễm sắc thể mới bằng toán tử di truyền.
 - (c) Chọn lọc tự nhiên.

1.3 Công nghệ ofbiz

Apache Ofbiz [23] là một công nghệ trên nền tảng ERP cung cấp các tính năng phong phú cho doanh nghiệp như thương mại điện tử, quản lý sản xuất, quản lý dự án bán lẻ và thương mại. Ofbiz có thể sử dụng trong các tổ chức trong tất cả các ngành và mọi quy mô ở bất kỳ nước nào. Ofbiz là framework mã nguồn mở nên các nhà sản xuất hoàn toàn có thể mở rộng và phát triển để thực hiện các tính năng riêng của từng sản phẩm.

Ofbiz sử dụng ngôn ngữ lập trình Java xây dựng ứng dụng Web. Kiến trúc ofbiz gồm 3 tầng:

- **Tầng dữ liệu** là tầng chịu trách nhiệm truy cập dữ liệu từ cơ sở dữ liệu, lưu trữ và cung cấp các giao diện truy cập cho tầng nghiệp vụ.
- **Tầng nghiệp vụ** cung cấp các service cho người sử dụng. Có nhiều loại service trong tầng này: Java methods, simple object access protocol là một dạng web service (SOAP), simple services, workflow, etc. Service engine sẽ chịu trách nhiệm quản lý transaction và bảo mật.
- **Tầng trình diễn** là tầng trên cùng của ứng dụng cung cấp các giao diện người dùng. Ofbiz sử dụng các khái niệm "screen" để trình bày các trang web. Một giao diện người dùng có thể được ghép lại bởi nhiều screen. Mỗi screen tương ứng với một component. Component có thể được viết bằng Java Server Pages (JSP) hoặc FreeMarker Template Language (FTL).

Chương 2

Bài toán lập lộ trình giao hàng kết hợp một xe tải với một thiết bị bay cùng các nghiên cứu liên quan

Trong chương này chúng tôi trình bày về bài toán lập lộ trình giao hàng kết hợp một xe tải với một thiết bị bay không người lái cùng hai thuật toán A Greedy Randomized Adaptive Search Procedure (GRASP) và TSP-LS. Cả hai thuật toán đều xây dựng lời giải cho bài toán yêu cầu từ kết quả của bài toán người du lịch tương ứng. Trong phần 2.1 chúng tôi mô tả bài toán lập lộ trình giao hàng kết hợp một xe tải với một thiết bị bay, các yêu cầu, ràng buộc và mô hình quy hoạch nguyên bộ phận. Trong phần 2.2 và 2.3, chúng tôi trình bày cụ thể về thuật toán GRASP và TSP-LS bao gồm ý tưởng và mã giải. Các tài liệu tham khảo sử dụng trong phần này bao gồm [9], [10] [11], [12], [13], [14], [15], [16], [17], [18], [19].

2.1 Bài toán lập lộ trình vận tải giao hàng kết hợp xe tải và một thiết bị bay

2.1.1 Mô tả bài toán

Trong bài toán giao hàng truyền thống, xe tải nhận hàng từ một điểm sau đó đi giao hàng ở tất cả các cửa hàng, mỗi cửa hàng đúng một lần. Mô hình này được gọi là bài toán người du lịch. Nhưng gần đây, dưới sự phát triển mạnh mẽ của phương tiện bay không người lái, một phương thức giao hàng mới đã được ra đời là Traveling salesman problem with Drone (TSPD), trong đó drone tham gia giao hàng cùng với xe tải. Dưới đây là bốn lợi ích nổi bật của drone [9] :

- Drone không cần người lái.
- Drone là một phương tiện bay.
- Drone bay giao hàng nhanh hơn xe tải.
- Drone tốn ít chi phí hơn xe tải.

Ngoài những ưu điểm drone cũng có những điểm hạn chế:

- Drone có giới hạn năng lượng, chỉ bay được một khoảng nhất định sau đó nó phải sạc điện.
- Drone chỉ mang được giới hạn khối lượng.

Ngược lại với drone, xe tải là phương tiện có thể chở nhiều hàng hóa và có nhiều năng lượng, có thể chạy quãng đường dài. Dựa trên những ưu nhược điểm của hai phương tiện này một phương thức giao hàng được đề xuất có tên gọi là "last mile delivery with drone" [10]. Trong phương thức giao hàng này, xe tải mang toàn bộ kiện hàng mà nó phải giao đi. Sau đó xe tải vận chuyển drone đến gần địa điểm khách hàng và cho phép drone phục vụ khách hàng trong phạm vi nó hoạt động được. Trong lúc đó xe tải vẫn tiếp tục thực hiện tiếp lô trình của mình. Drone phục vụ khách hàng xong sẽ quay lại xe tải ở một điểm mới khác với điểm nó bay lên.

Cho một danh sách khách hàng cần phục vụ, để giao hàng drone được thả ở một điểm và quay lại ở một điểm khác. Mỗi khách hàng được phục vụ chỉ bởi drone hoặc xe tải. Hai phương tiện xuất phát cùng nhau từ kho hàng và lại quay về kho hàng khi giao hàng xong. Chúng ta gọi một chuyến giao hàng bởi drone - drone delivery (DD) gồm 3 thành phần $\langle i, j, k \rangle$ trong đó i là điểm bay của drone, j là điểm khách hàng mà drone giao hàng tới và k là điểm mà drone quay trở lại xe tải. Các quy định về điểm bay, điểm giao hàng và điểm đón như sau:

- Điểm bay của drone (launch node) là điểm mà drone rời khỏi xe tải đi giao hàng. Điểm bay của drone bắt buộc phải là một điểm giao hàng hoặc điểm kho hàng.
- Điểm giao hàng (drone node) là một vị trí cần giao hàng. Không phải tất cả các điểm đều có thể trở thành điểm giao hàng của drone vì có những điểm ngoài phạm vi phục vụ được của drone.
- Điểm hạ cánh (rendezvous node) hay còn gọi là điểm đón là điểm mà drone quay trở lại xe tải. Tất nhiên hai phương tiện có thể phải chờ nhau tại điểm này.

Gọi một hành trình giao hàng bởi xe tải - truck delivery (TD) là một chuỗi lộ trình qua các điểm $\langle e_0, e_1, \dots, e_k \rangle$ với e_0 và e_k là điểm kho. Drone có một mức năng lượng nhất định vì vậy nó chỉ bay được liên tục trong một khoảng cách nhất định mà drone có thể bay (endurance), trong đồ án này chúng tôi gọi là mức chịu đựng. Một DD gọi là chấp nhận được nếu quãng đường từ điểm bay qua điểm giao hàng về điểm đón nhỏ hơn mức chịu đựng của drone. Cả xe tải và drone đều có một đại lượng là chi phí vận chuyển. Trong thực tế, chi phí vận chuyển của drone thường thấp hơn nhiều so với xe tải bởi vì trọng lượng của nó nhỏ hơn và cũng tốn ít năng lượng hơn.

2.1.2 Các nghiên cứu liên quan

Trong phần này chúng tôi trình bày các nghiên cứu liên quan đến bài toán vận tải với sự tham gia của xe tải và drone.

Năm 2015, Murray và Chu [11] giới thiệu bài toán Flying Sidekick Traveling Salesman Problem (FSTSP). Một mô hình Mixed Integer Programming (MIP) và một heuristic cũng được đề xuất. Heuristic dựa trên ý tưởng "xe tải trước, drone sau" trong đó đầu

tiên họ xây dựng lộ trình cho xe tải bằng việc giải bài toán Traveling salesman problem (TSP) và sau đó lặp đi lặp lại việc sử dụng thủ tục *relocation* để giảm hàm mục tiêu. Thực chất thủ tục *relocation* kiểm tra mỗi điểm thuộc hành trình TSP và thử chuyển chúng thành các điểm được giao hàng bởi drone. Khi một điểm đã được chuyển thành drone nó sẽ không bao giờ được kiểm tra lại nữa. Một khía cạnh khác các điểm giao hàng có thể thay đổi vị trí trong lộ trình nếu việc đặt vào một vị trí mới có thể làm giảm chi phí của lời giải.

Agatz [12] trong một án phẩm không chính thức nghiên cứu một vấn đề hơi khác gọi là Traveling Salesman Problem with Drone (TSP-D), trong đó drone phải đi theo mạng lưới đường bộ như chiếc xe tải. Hơn nữa, trong bài toán này drone phải bay lên và hạ cánh ở cùng vị trí, trong khi với FSTSP điều này bị cấm. Bài toán cũng được giải bởi heuristic "Xe tải trước, Drone sau" trong đó lộ trình của drone được xây dựng dựa trên tìm kiếm cục bộ hoặc quy hoạch động. Gần đây hơn, Ponza [13] kế thừa từ nghiên cứu của Murray và Chu [11], đề xuất một nâng cấp của mô hình MIP cho bài toán FSTSP và giải quyết bằng một phương pháp heuristic dựa trên Simulated Annealing ¹.

Tất cả các nghiên cứu được nhắc đến trên đều có hàm mục tiêu là thời gian xe tải hoàn thành lộ trình và quay trở lại điểm kho, tối ưu hàm mục tiêu này hướng đến tăng chất lượng phục vụ [15]. Gần đây một nghiên cứu bởi Mathew [18] cho một vấn đề được gọi là Heterogeneous Delivery Problem (HDP). Tuy nhiên, không giống những bài toán được nghiên cứu trước đây bài toán được mô hình trên một đường phố thực tế mà ở đó xe tải không được phép giao hàng đến khách hàng. Thay vào đó, tại mỗi điểm cuối của cạnh, xe tải có thể thả một drone di phục vụ khách hàng. Bằng cách này, bài toán có thể được chuyển đổi thuận lợi sang bài toán Generalized Traveling Salesman Problem (GTSP) [19].

Kế thừa nghiên cứu của Murray và Chu [11] và nhận thấy trong các mô hình vận tải chi phí vận chuyển mới thường là quan trọng (xem [16] và [17]), tác giả Hà Quang Minh và cộng sự đề xuất mô hình min-cost TSPD [9] với các đặc điểm:

- Xe tải và drone có thể đi cùng nhau, cả hai phương tiện đều có thể phục vụ các điểm khách hàng.
- Drone phải bay lên từ một điểm khách hàng và quay lại xe tải ở một điểm khác với điểm nó bay lên.
- Xe tải không thể quay lại điểm mà nó đã đi qua để đón drone.

Và quan trọng hơn cả hàm mục tiêu của bài toán là tối thiểu chi phí vận chuyển của cả hai phương tiện. Đây cũng chính là bài toán được chúng tôi trình bày và là nội dung chính của chương này.

Bởi vì bài toán sẽ trở thành bài toán TSP khi sức chịu đựng drone giảm dần về không, cho nên bài toán là NP-hard.

2.1.3 Mô hình bài toán min-cost TSPD

Mô hình bài toán được xây dựng trùu tượng dựa trên mô hình đồ thị $G = (V, A)$, $V = \{0, 1, \dots, n, n + 1\}$ với đỉnh 0 và $n + 1$ tương ứng là kho. N khách hàng được giao tương

¹Simulated Annealing là một phương pháp metaheuristic

ứng với tập N điểm tương ứng là đỉnh 1 đến đỉnh n trong đồ thị G . Gọi d_{ij} và d'_{ij} là khoảng cách từ đỉnh i đến đỉnh j theo xe tải và drone. Tiếp đó C_1 và C_2 là chi phí phải trả cho xe tải và drone khi đi một đơn vị độ dài. Gọi s là một tập các đỉnh theo thứ tự $s = \langle s_1, s_2, \dots, s_t \rangle$, $s_i \in V, i = 1, \dots, t$ hay gọi là chuỗi lộ trình. Chúng ta định nghĩa các khái niệm sau:

- $V(s) \subseteq V$ là danh sách các đỉnh có trong s .
- $pos(i, s)$ vị trí của đỉnh i trong s .
- $next_s(i)$ đỉnh tiếp theo của i trong s .
- $prev_s(i)$ đỉnh trước của i trong s .
- $first(s)$ đỉnh bắt đầu trong s .
- $last(s)$ đỉnh kết thúc trong s .
- $s[i]$ đỉnh thứ i trong s .
- $size(s)$ số lượng đỉnh trong s .
- $sub(i, j, s)$ với $i, j \in s$ là một chuỗi lộ trình con của s .
- $A(s) = \{(i, next_s(i)) | i \in V(s) \setminus last(s)\}$ là tập cạnh thuộc đường đi của s .

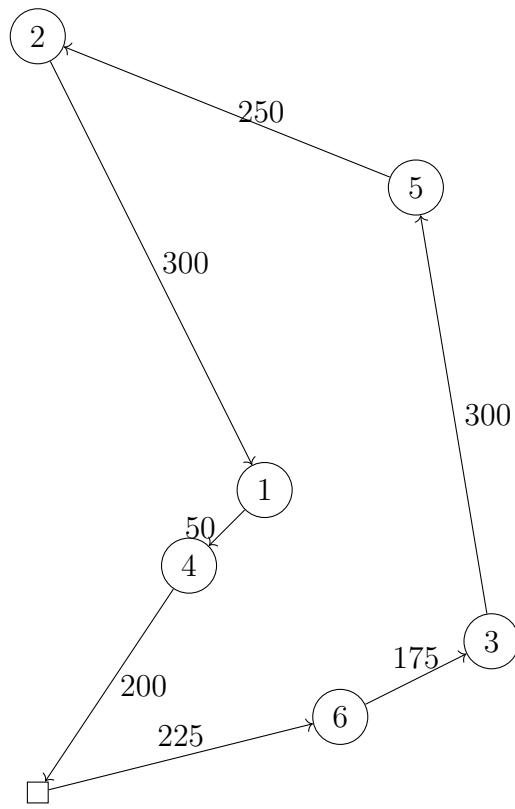
Một DD là một tập $\langle i, j, k \rangle \quad i, j, k \in V, i \neq j, j \neq k, k \neq i, d'_{ij} + d'_{jk} \leq \epsilon$, với ϵ là một hằng số thể hiện quãng đường lớn nhất mà drone có thể bay được (sức chịu đựng của drone), Δ là thời gian chờ đợi lớn nhất của cả hai phương tiện. Tất cả các DD thỏa mãn trên đồ thị G là tập

$$P = \{\langle i, j, k \rangle : i, j, k \in V, i \neq j, j \neq k, k \neq i, d'_{ij} + d'_{jk} \leq \epsilon, |d_{i \rightarrow k} - (d'_{ij} + d'_{jk})| \leq \Delta\}$$

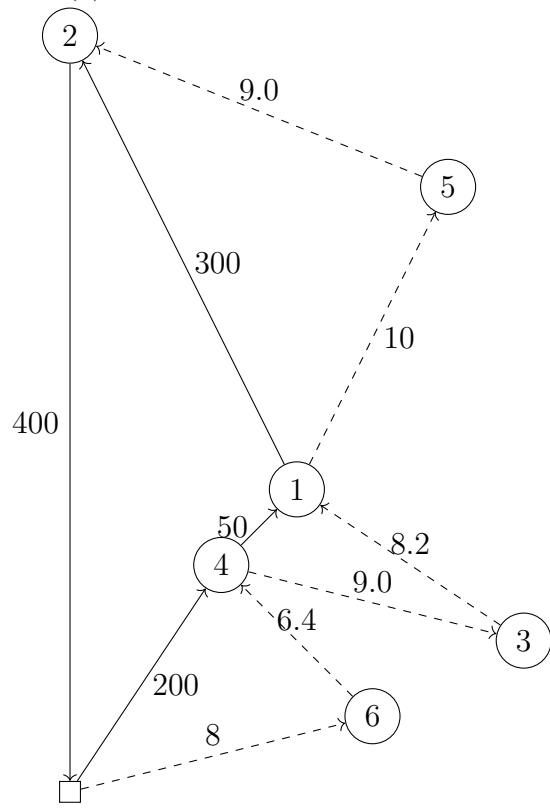
Một lời giải của bài toán min-cost TSPD bao gồm 2 thành phần:

- Một TD là một chuỗi lộ trình qua các điểm $\langle e_0, e_1, \dots, e_k \rangle$ với $e_0 = 0$ và $e_k = n + 1$ 0 và $n + 1$ là điểm kho, $e_i \neq e_j, \forall i \neq j$
- Một tập $DDs = \{DD | DD \subseteq P\}$

Hình 2.1 mô tả một ví dụ về lời giải cho bài toán min-cost TSPD.



(a) Một hành trình người du lịch



(b) Một hành trình TSPD

Hình 2.1: Lộ trình bài toán TSP và lộ trình bài toán TSPD tương ứng [9]

2.1.4 Ràng buộc

Một lời giải của bài toán min-cost TSPD phải đảm bảo những ràng buộc sau đây.

(A) Mỗi khách hàng luôn phải được phục vụ bởi xe tải hoặc drone.

$$\forall e \in N : e \in TD \text{ or } \exists \langle i, e, k \rangle \in DDS. \quad (2.1)$$

(B) Một khách hàng chỉ được phục vụ duy nhất bởi một drone.

$$\forall \langle i, j, k \rangle, \langle i', j', k' \rangle \in DD : j \neq j' \quad (2.2)$$

(C) Mỗi một DD phải thỏa mãn với TD.

$$\forall \langle i, j, k \rangle \in DDS : j \notin TD, i \in TD, k \in TD, pos(i, TD) < pos(k, TD) \quad (2.3)$$

(D) Luôn luôn chỉ tồn tại một drone tải một thời điểm.

$$\forall \langle i, ., k \rangle \in DD, \forall e \in sub(i, k, TD), \forall \langle i', j', k' \rangle \in DD : e \neq i' \quad (2.4)$$

2.1.5 Hàm mục tiêu

Một vài khái niệm

- $cost(i, j, k) = C_2(d'_{ij} + d'_{jk})$ $\langle i, j, k \rangle \in P$: chi phí của một DD.
- $cost(TD) = \sum (i, j) \in A(TD) : C_1 d_{ij}$: chi phí của một TD.
- $cost(DDs) = \sum \langle i, j, k \rangle \in DD : cost(i, j, k)$: chi phí của tất cả các DD trong lời giải.
- $cost(TD, DD) = cost(TD) + cost(DD)$: chi phí của lời giải.
- $cost(sub(i, k, s))$ có giá trị là chi phí của cả xe tải và drone trong chuỗi lộ trình s .

Hàm mục tiêu của bài toán là tối thiểu tổng chi phí của lời giải.

2.1.6 Mô hình Mixed Integer Programming

Bài toán sẽ được trình bày cụ thể trong phần này bởi mô hình Mixed Integer Programming.

Biến

Gọi $x_{ij} \in \{0, 1\}$ với $i \in V_L = \{0, 1, \dots, n\}, j \in V_R = \{1, 2, \dots, n+1\}$ biểu diễn TD, với $x_{ij} = 1$ nghĩa là xe tải đi từ điểm i đến điểm j .

Gọi $y_{ijk} \in \{0, 1\}$ biểu diễn DD với $y_{ijk} = 1$ nếu $\langle i, j, k \rangle$ là một DD.

Ta định nghĩa $p_{ij} \in \{0, 1\}$ có giá trị bằng 1 nếu điểm i được xe tải đến trước điểm j ($i \neq j$). Vì vậy ta khởi tạo $p_{0j} = 1$, $\forall j \in N$ để yêu cầu xe tải bắt buộc phải xuất phát từ điểm kho.

Gọi $u_i, 0 \leq u_i \leq n+1$ là vị trí của điểm i ($i \in V$) trong lộ trình của xe tải.

Mô hình MIP

$$\text{Min } C_1 \sum_{i \in V_L} \sum_{\substack{j \in V_R, \\ i \neq j}} d_{ij} x_{ij} + C_2 \sum_{i \in V_L} \sum_{j \in N, \substack{k \in V_R, \\ i \neq j}} \sum_{\langle i, j, k \rangle \in P} (d'_{ij} + d'_{jk}) y_{ijk} \quad (2.5)$$

$$\sum_{\substack{i \in V_L, \\ i \neq j}} x_{ij} + \sum_{\substack{i \in V_L, \\ i \neq j}} \sum_{\substack{k \in V_R, \\ \langle i, j, k \rangle \in P}} y_{ijk} = 1 \quad \forall j \in N \quad (2.6)$$

$$\sum_{j \in V_R} x_{0j} = 1 \quad (2.7)$$

$$\sum_{i \in V_L} x_{i,n+1} = 1 \quad (2.8)$$

$$u_i - u_j + 1 \leq (n+2)(1 - x_{ij}) \quad \forall i \in V_L, j \in \{V_R : i \neq j\} \quad (2.9)$$

$$\sum_{\substack{i \in V_L, \\ i \neq j}} x_{ij} = \sum_{\substack{k \in V_R, \\ k \neq j}} x_{jk} \quad \forall j \in N \quad (2.10)$$

$$2y_{ijk} \leq \sum_{\substack{h \in V_L, \\ h \neq i}} x_{hi} + \sum_{\substack{l \in N, \\ l \neq k}} x_{lk}, \quad \forall i \in N, j \in \{N : i \neq j\}, k \in \{V_R : \langle i, j, k \rangle \in P\} \quad (2.11)$$

$$y_{0jk} \leq \sum_{\substack{h \in V_L, \\ h \neq k \\ h \neq j}} x_{hk} \quad j \in N, k \in \{V_R : \langle 0, j, k \rangle \in P\} \quad (2.12)$$

$$u_k - u_i \geq 1 - (n+2)(1 - \sum_{\substack{j \in N, \\ \langle i, j, k \rangle \in P}} y_{ijk}) \quad \forall i \in V_L, k \in V_R : k \neq i \quad (2.13)$$

$$\sum_{j \in N, \substack{k \in V_R, \\ j \neq i}} y_{ijk} \leq 1 \quad \forall i \in V_L \quad (2.14)$$

$$\sum_{i \in V_L, \substack{j \in N, \\ i \neq k}} y_{ijk} \leq 1 \quad \forall k \in V_R \quad (2.15)$$

$$u_i - u_j \geq 1 - (n-2)p_{ij} - M(2 - \sum_{\substack{h \in V_L, \\ h \neq i}} x_{hi} - \sum_{\substack{k \in N, \\ k \neq j}} x_{kj}) \quad \forall i \in N, j \in \{V_R : j \neq i\} \quad (2.16)$$

$$u_i - u_j \leq -1 + (n+2)(1 - p_{ij}) + M(2 - \sum_{\substack{h \in V_L, \\ h \neq i}} x_{hi} - \sum_{\substack{k \in N, \\ k \neq j}} x_{kj}) \quad \forall i \in N, j \in \{V_R : j \neq i\} \quad (2.17)$$

$$u_0 - u_j \geq 1 - (n-2)p_{0j} - M(1 - \sum_{\substack{k \in N, \\ k \neq j}} x_{kj}) \quad \forall j \in V_R \quad (2.18)$$

$$u_0 - u_j \leq -1 + (n+2)(1 - p_{0j}) + M(1 - \sum_{\substack{k \in N, \\ k \neq j}} x_{kj}) \quad \forall j \in V_R \quad (2.19)$$

$$u_l \geq u_k - M(3 - \sum_{\substack{j \in N, \\ j \neq i \\ \langle i, j, k \rangle \in P}} y_{ijk} - \sum_{\substack{m \in N, \\ m \neq i \\ m \neq l}} \sum_{\substack{n \in V_R, \\ n \neq i \\ n \neq k \\ \langle l, m, n \rangle \in P}} y_{lmn} - p_{il}) \quad (2.20)$$

$\forall i \in V_L, k \in \{V_R : k \neq i\}, l \in \{N : l \neq i, l \neq k\}$

Hàm mục tiêu là tối ưu tổng chi phí (2.5). Các ràng buộc trong mô hình quy hoạch nguyên được giải thích như sau:

- Ràng buộc (2.6) đảm bảo mỗi một khách hàng chỉ được giao một lần bằng drone hoặc xe tải (2.1).
- Ràng buộc (2.7), (2.8) đảm bảo xe tải luôn xuất phát từ điểm kho.
- Ràng buộc (2.9) đảm bảo cho một chuỗi lộ trình thỏa mãn yêu cầu.
- Ràng buộc (2.10) đảm bảo xe tải đến và đi tại mọi điểm khách hàng mà nó giao.
- Ràng buộc (2.11) đảm bảo điểm bay và điểm đón drone phải được xe tải đến thăm.
- Ràng buộc (2.12) đảm bảo nếu drone bay từ điểm kho xe tải phải thăm điểm đón nó.
- Ràng buộc (2.13) đảm bảo rằng nếu DD $\langle i, j, k \rangle$ thì xe tải luôn phải giao hàng ở i trước ở k .
- Ràng buộc (2.14), (2.15) đảm bảo không có 2 DD nào có cùng điểm khách hàng (2.2).
- Ràng buộc (2.16), (2.17), (2.18), (2.19) đảm bảo nếu điểm i được giao hàng trước điểm j trong TD thì thứ tự của chúng phải được đảm bảo.
- Ràng buộc (2.20) đảm bảo luôn tồn tại một DD tại một thời điểm trên lộ trình (2.4).

2.2 Thuật toán GRASP

Trong phần này chúng tôi trình bày thuật toán A Greedy Randomized Adaptive Search Procedure (GRASP) cho bài toán min-cost TSPD. Thuật toán gồm hai bước: (1) thuật toán phân tách, (2) thuật toán tìm kiếm cục bộ. Ở bước 1, chúng tôi xây dựng thuật toán phân tách nhằm sinh lời giải cho bài toán min-cost TSPD từ lời giải của bài toán TSP, lời giải TSP được sinh ra bởi thư viện CBLSVR. Trong bước hai, thuật toán tìm kiếm cục bộ được sử dụng để cải thiện lời giải được sinh ra từ thuật toán phân tách trong bước trước. Trong bước này, chúng tôi trình bày các toán tử được làm mới để phù hợp cho bài toán min-cost TSPD. Thuật toán được trình bày cụ thể trong thuật toán 1.

Algorithm 1 A Greedy Randomized Adaptive Search Procedure (GRASP)

Input: Tập các điểm cần giao hàng, kho, các tham số về chi phí và khoảng cách, thời gian chờ đợi, mức chịu đựng của drone.

Output: Chi phí nhỏ nhất thu được và một hành trình cho xe tải và drone.

```
1: bestSolution = null;  
2: bestObjectValue =  $\top$ ;  
3: iteration = 0;  
4: while iteration <  $n_{TSP}$  do  
5:   iteration = iteration + 1;  
6:   tour = Sinh lời giải từ thư viện CBLSVR  
7:   tspdSolution = SplitAlgorithm(tour);  
8:   tspdSolution = LocalSearch(tspdSolution);  
9:   if  $f(\textit{tspdSolution}) < \textit{bestObjectValue}$  then  
10:    bestSolution = tspdSolution;  
11:    bestObjectiveValue =  $f(\textit{tspdSolution})$ ;  
12:    iteration = 0;  
13:   end if  
14: end while  
15: return bestSolution
```

2.2.1 Thuật toán phân tách

Trong phần này chúng tôi sẽ trình bày thuật toán phân tách (bước 1 trong thuật toán GRASP). Thuật toán gồm 2 bước nhỏ: (1) xây dựng auxiliary graph, (2) sinh lời giải.

Ý tưởng của thuật toán phân tách là xây dựng lời giải min-cost TSPD từ TSP bằng cách tách các điểm giao hàng bằng drone. Bắt đầu từ một hành trình người du lịch (TSP) thuật toán chọn ra các điểm cần chuyển thành drone, với một điểm khách hàng được chọn giao bằng drone, điểm đó sẽ được xóa khỏi hành trình xe tải và hình thành một DD. Trong thuật toán này, để sinh lời giải TSP chúng tôi sử dụng thư viện CBLSVR [20].

Algorithm 2 Thuật toán phân tách

(Step 1: Xây dựng auxiliary graph và tìm đường đi ngắn nhất)

Input: Hành trình TSP s.**Output:** P lưu đường đi ngắn nhất từ auxiliary graph, V là giá trị của các đường đi ngắn nhất, và T là danh sách các DD có thể và giá trị của nó.

```
1:  $arcs = \emptyset;$ 
2:  $T = \emptyset;$ 
3: for each  $i \in s \setminus \text{last}(s)$  do
4:    $k = pos(i, s) + 1;$ 
5:    $arcs = arcs \cup \{(i, k, cost(i, k, s))\};$ 
6: end for
7: for each  $i \in s \setminus \{last(s), s[pos(last(s), s) - 1]\}$  do
8:   for each  $k \in s : pos(k, s) \geq pos(i, s) + 2$  do
9:      $minValue = \top;$ 
10:     $minIndex = \top;$ 
11:    for each  $j \in s : pos(i, s) < pos(j, s) < pos(k, s)$  do
12:      if  $\langle i, j, k \rangle \in P$  và  $|d_{i \rightarrow k} - (d'_{ij} + d'_{jk})| < \Delta$  then
13:         $cost =$ 
14:         $cost(sub(i, k, s)) + C_1(d_{prev(j, s)next(j, s)} - d_{prev(j, s), j} - d_{j, next(j, s)}) + cost(i, j, k);$ 
15:        if  $cost < minValue$  then
16:           $minValue = cost;$ 
17:           $minIndex = pos(j, s);$ 
18:        end if
19:      end if
20:    end for
21:     $arcs = arcs \cup \{(i, k, minValue)\};$ 
22:    if  $minIndex \neq \top$  then
23:       $T = T \cup \{(i, s[minIndex], k, minValue)\};$ 
24:    end if
25:  end for
26:   $V[0] = 0.0;$ 
27:   $P[0] = 0;$ 
28: for each  $k \in s \setminus \{0\}$  do
29:   for each  $(i, k, cost) \in arcs$  do
30:     if  $V[k] > V[i] + cost$  then
31:        $V[k] = V[i] + cost;$ 
32:        $P[k] = i;$ 
33:     end if
34:   end for
35: end for
36: return  $(P, V, T)$ 
```

Xây dựng auxiliary graph và tìm lô trình ngắn nhất

Trong thuật toán 2 chúng tôi xây dựng một đồ thị $H = (V', A')$ dựa trên hành trình TSP s của đồ thị $G = (V, A)$ với $V = V'$, mọi cạnh $arc(i, j) \in A'$ biểu diễn lô trình con từ i đến j với điều kiện $pos(i, s) < pos(j, s)$.

Với i, j là hai đỉnh liên tiếp nhau chi phí $c_{i,j}$ của cạnh $arc(i, j)$ được tính bằng công thức:

$$c_{ij} = C_1 d_{ij}$$

Tuy nhiên khi đỉnh i, k không liên tiếp nhau mà đỉnh j nằm giữa i, k sao cho $\langle i, j, k \rangle \in P$

$$c_{ik} = \min_{\langle i, j, k \rangle \in P} cost(sub(i, k, s)) + C_1(d_{prev_{sj}, next_{sj}} - d_{prev_{sj}, j} - d_{j, next_{sj}}) + cost(i, j, k)$$

Nếu đỉnh i, k không liền nhau và không tồn tại đỉnh j ở giữa i, k sao cho $\langle i, j, k \rangle$ có thể là một DD thì:

$$c_{ik} = +\infty$$

Thuật toán tính chi phí cạnh trong đồ thị H được trình bày từ dòng 1 đến dòng 25 trong thuật toán 2. Danh sách T chứa các DD được lưu lại để sử dụng cho thuật toán xây dựng lời giải (dòng 21 đến 22).

Đồ thị H được sử dụng để tính toán chi phí v_k là đường đi ngắn nhất từ kho đến đỉnh k . Bởi vì đồ thị xây dựng là có hướng và không có chu trình nên giá trị v_k có thể tính dễ dàng bằng thuật toán quy hoạch động. Hơn nữa, một cạnh trong đường đi ngắn nhất (một cung trong đồ thị H) $arc(i, k)$ hoàn toàn có thể tạo ra một DD với điểm bay là i , điểm đón là k . Chính lợi điểm này luôn luôn đảm bảo không có hai DD nào overlap, tức một thời điểm không thể nào có hai drone đang bay.

Cụ thể, đặt $v_0 = 0$ giá trị v_k của mỗi đỉnh $k \in V' \setminus \{0\}$ được tính bằng:

$$v_k = \min\{v_i + c_{ik} : (i, k) \in A'\} \quad \forall k = 2, \dots, n+1$$

Chúng tôi lưu một mảng lưu vết P , $P(j), j = 1, \dots, n+1$ là đỉnh trước đó của của j . Các bước xây dựng v, P được trình bày trong thuật toán 2 từ dòng 27 đến dòng 34.

Thuật toán sinh lời giải

Dựa vào P, T tìm được phần trước, trong phần này chúng tôi trình bày thuật toán 3 sinh lời giải cho bài toán min-cost TSPD. Bước đầu tiên chúng tôi xây dựng một tập đỉnh $S_a = 0, 1, \dots, n+1$ trình bày đường đi từ 0 đến $n+1$ trên auxiliary graph (dòng 1 đến dòng 8 trong thuật toán 3). Hai điểm gần nhau trong S_a trình bày một lộ trình con trong lời giải. Tuy nhiên chúng ta vẫn phải xây dựng các DD tương ứng với mỗi lộ trình con đó. Các DD này đã được tính toán sẵn và lưu lại trong T ở bước trước.

Lời giải cho bài toán min-cost TSPD gồm 2 thành phần TD và DDs, ở đây chúng tôi định nghĩa S_t (TD) và S_d (DDs). Để xây dựng DDs, với một cặp điểm $i, i+1$ trong S_a xác định số điểm trong tourTSP nằm giữa hai điểm này, nếu có ít nhất một điểm nằm giữa hai điểm trên thì ta lấy DD với điểm đầu cuối tương ứng trong T bỏ vào DDs (S_d). Thuật toán được trình bày trong thuật toán 3 dòng 10 đến 15.

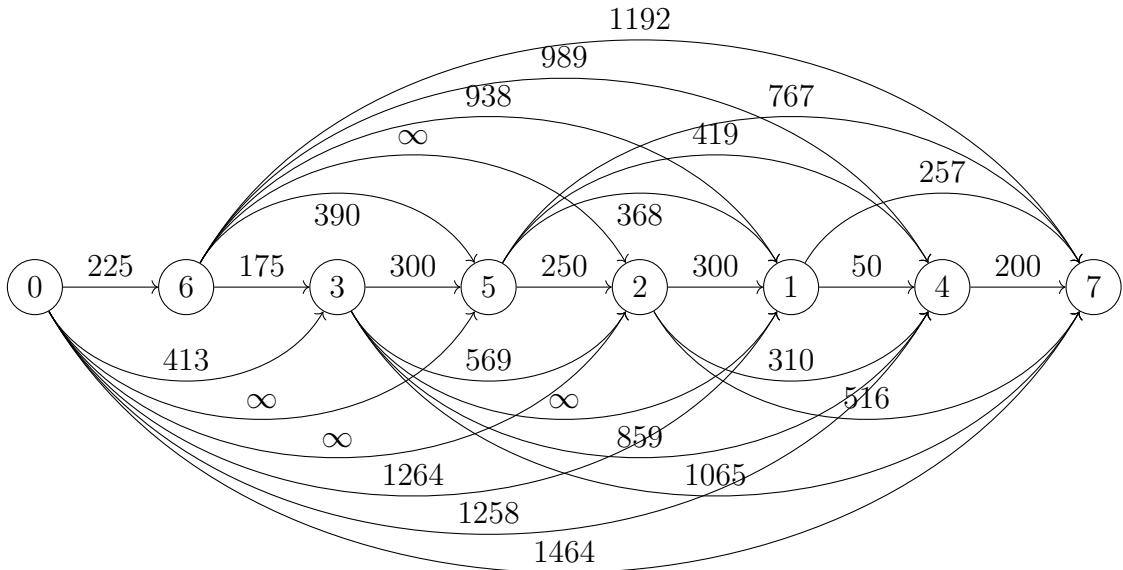
Để xây dựng TD, ta bắt đầu từ điểm 0 trong S_a . Mỗi cặp điểm $i, i+1$ trong S_a ta tạo ra một lộ trình con trong lời giải TSPD bằng việc lấy các điểm khách hàng từ hành trình TSP. Tuy nhiên trong trường hợp $\langle i, j, i+1 \rangle$ là một DD thì ta phải loại bỏ j ra khỏi TD.

Algorithm 3 Thuật toán phân tách

(Step 2: Sinh lời giải từ step 1)

Input: P lưu đường đi ngắn nhất từ auxiliary graph, V là giá trị của các đường đi ngắn nhất, và T là danh sách các DD có thể và giá trị của nó và tspTour là lộ trình của xe tải
Output: Chi phí và lời giải cho bài toán min-cost TSPD.

```
1:  $j = n + 1;$ 
2:  $i = \top;$ 
3:  $S_a = \{j\}$ 
4: while  $i \neq 0$  do
5:    $i = P[j];$ 
6:    $S_a = S_a \cup \{i\};$ 
7:    $j = i;$ 
8: end while
9:  $S_a = S_a.reverse();$ 
10:  $S_t = \emptyset;$ 
11:  $S_d = \emptyset;$ 
12: for  $i = 0; i < S_a.size - 1; i++$  do
13:   if Giữa  $S_a[i]$  và  $S_a[i + 1]$  trong  $tspTour$ , có ít nhất một điểm then
14:      $n_{drone} =$  điểm giao hàng tương ứng được lưu trong T
15:      $S_d = S_d \cup \{\langle S_a[i], n_{drone}, S_a[i + 1] \rangle\};$ 
16:   end if
17: end for
18:  $currentPosition = 0$ 
19: while  $currentPosition \neq n + 1$  do
20:   if  $currentPosition$  là một điểm bay của một DD t trong  $S_d$  then
21:      $S_t = S_t \cup \{$  tất cả các điểm thuộc  $tspTour$ 
22:     nằm từ  $currentPosition$  đến  $t_{rendezvous\_node}$  ngoại trừ  $t_{drone\_node}\}$ 
23:      $currentPosition = t_{rendezvous\_node}$ 
24:   else
25:      $S_t = S_t \cup \{currentPosition\};$ 
26:      $currentPosition = tspTour[indexOf(currentPosition) + 1];$ 
27:   end if
28: end while
29:  $tspdSolution = (S_t, S_d);$ 
29: return  $tspdSolution$ 
```



Hình 2.2: Auxiliary graph cho lô trình TSP được trình bày trong hình 2.1 [9]

2.2.2 Các toán tử tìm kiếm cục bộ

Trong phần này chúng tôi trình bày các toán tử cho bài toán min-cost TSPD, bao gồm: (1) *relocate_T*, (2) *relocate_D*, (3) *remove_D*, (4) *two_exchange*.

Hai toán tử tìm kiếm cục bộ được lấy cảm hứng từ truyền thống là *relocate_truck* và *two_exchange*. Ngoài ra, dựa trên những đặc thù riêng biệt của bài toán min-cost TSPD chúng tôi trình bày thêm hai toán tử mới đó là "*drone_relocate*" dùng để tạo ra một DD và "*drone_removal*" dùng để xóa bỏ một DD. Để bắt đầu chúng ta định nghĩa một vài từ khóa sau:

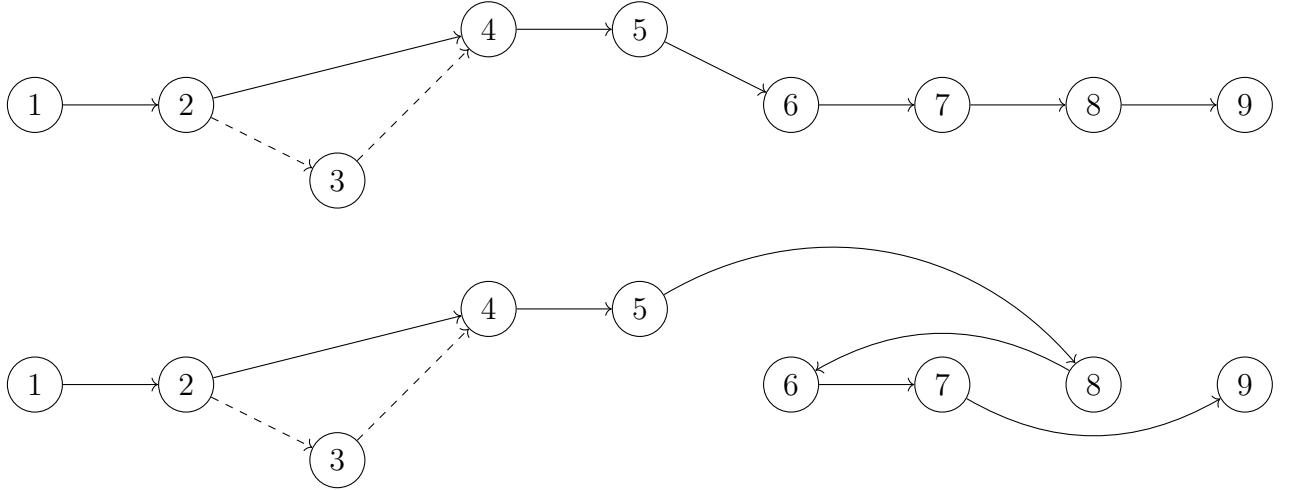
- $N_T(TD, DDs) = \{e : e \in TD, \langle e, ., . \rangle \notin DDs, \langle ., ., e \rangle \notin DDs\}$ là tập các điểm chỉ thuộc TD, không nằm trong bất kì DD nào.
- $N_D(TD, DDs) = \{e : \langle ., e, . \rangle \in DD\}$ là tập các điểm giao hàng drone trong lời giải TSPD.

Tiếp theo chúng tôi định nghĩa chi tiết từng toán tử như sau :

1. *relocation_truck* : Đây là toán tử truyền thống được làm mới để phù hợp với bài toán min-cost TSPD với hai điểm khác biệt sau: (1) chúng tôi chỉ cân nhắc các điểm thuộc N_T , (2) chúng tôi chỉ di chuyển nút vào các vị trí trong TD. Cụ thể ta định nghĩa

$$\text{relocate}_T((TD, DDs), a, b), a \in N_T((TD, DDs)), b \in TD, b \neq a, b \neq 0$$

là toán tử di chuyển điểm a đến trước điểm b trong TD.

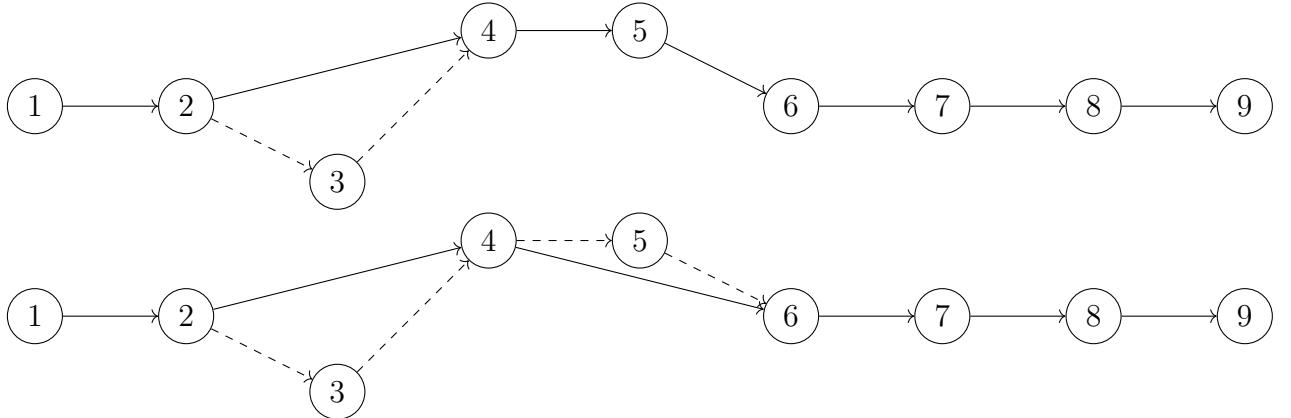


Hình 2.3: Toán tử *truck_relocation*

2. *Drone_relocation*: ý tưởng của toán tử là chuyển một điểm giao hàng bằng xe tải sang drone hoặc di chuyển điểm giao hàng drone qua các điểm bay và điểm đón mới. Cụ thể chúng tôi định nghĩa:

$$\begin{aligned} & \text{relocate}_D((TD, DDs), a, i, k) \\ & a \in N_T(TD, DD) \cup N_D(TD, DDs), i, k \in TD \setminus \{a\}, \\ & i \neq k, pos(i, TD) < pos(k, TD), \langle i, a, k \rangle \in P \end{aligned}$$

là một toán tử thực hiện xây dựng DD $\langle i, a, k \rangle$. Có hai trường hợp xảy ra: (1) nếu a là một điểm giao hàng bởi xe tải, toán tử sẽ thực hiện tạo $\langle i, a, k \rangle$ sau đó xóa bỏ a khỏi TD, (2) nếu a là DD toán tử thực hiện thay đổi điểm bay và điểm đón của DD đó thành $\langle i, a, k \rangle$.

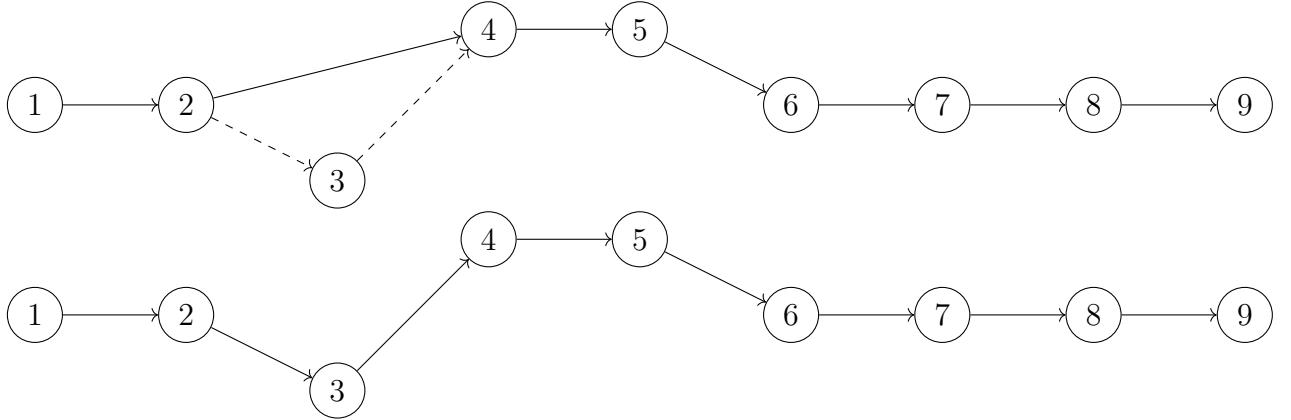


Hình 2.4: Toán tử *drone_relocation*

3. *Drone_Removal* : Trong toán tử này chúng ta chọn một điểm giao hàng bằng drone bất kì và di chuyển nó thành điểm giao hàng bởi xe tải. Chi tiết, chúng tôi định nghĩa toán tử:

$$remove_D((TD, DD), j, k), j \notin TD, \langle ., j, . \rangle \in DDs, k \in TD, k \neq \{0, n + 1\}$$

xóa bỏ DD $\langle ., j, . \rangle$ và điểm giao hàng bằng drone j được chuyển thành điểm giao hàng bằng xe tải và đặt trước k trong TD.

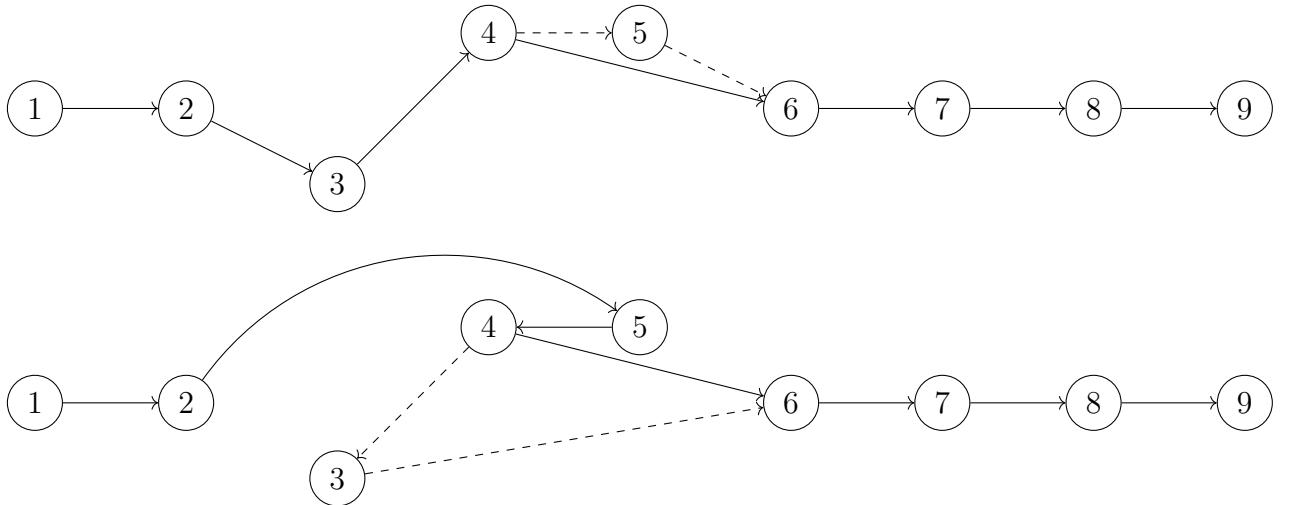


Hình 2.5: Toán tử *drone_removal*

4. *two_exchange* : Ở toán tử này, chúng tôi đổi vị trí hai điểm bất kì trong lời giải, có 3 khả năng có thể xảy ra: (1) đổi vị trí của hai điểm giao hàng bởi drone, (2) đổi vị trí giao hàng của 2 điểm giao hàng bằng xe tải,(3) đổi điểm giao hàng giữa hai phương tiện. Chi tiết, ta định nghĩa toán tử :

$$\text{two_exchange}((TD, DDS), a, b), a, b \in V \setminus \{0, n + 1\}, a \neq b$$

ta đổi vị trí hai điểm a, b tương ứng với 3 trường hợp.



Hình 2.6: Toán tử *two_exchange*

Tất nhiên khi sử dụng tất cả các toán tử trên phải thỏa mãn các ràng buộc của bài toán min-cost TSPD.

2.3 Thuật toán tìm kiếm cục bộ

Trong phần này chúng tôi sẽ trình bày một thuật toán tìm kiếm cục bộ để giải quyết bài toán TSPD. Ý tưởng chủ đạo của thuật toán như sau: thuật toán bắt đầu bằng một hành trình người du lịch sau đó liên tục thay đổi vị trí khách hàng trong lộ trình đến bao giờ không có phép đổi nào có thể làm giảm được chi phí nữa, cụ thể được trình bày bởi thuật toán 4. Dòng 1-8 khởi tạo các biến toàn cục, cụ thể:

- *Customers* : n khách hàng đánh số từ $1 \rightarrow n$.
- *truckRoute* : TD
- *truckSubRoute* chứa tất cả các tập con của *truckRoute* tức là chứa tất cả các *subroute* có thể của *truckRoute* (*subroute* là một chuỗi lô trình con của *truckRoute*).
- i^*, j^*, k^* với j^* là khách hàng cần đổi chỗ nhất và giữa khách hàng i^*, k^* là vị trí tốt nhất để đặt khách hàng j^* trong lô trình.
- *maxSavings*: giá trị giảm được sau khi đặt lại vị trí của j^* .
- *isDroneNode*: xác định có phải là một thao tác tạo DD
- *Stop*: xác định kết thúc vòng lặp thuật toán hay tiếp tục.

Thuật toán sẽ dừng lại khi *maxSavings* không còn dương.

Trong thuật toán 4 ta định nghĩa

$$drone(s, (TD, DDS)) = \begin{cases} True & \text{if } \exists j \in V(s), j \neq first(s), \\ & j \neq last(s) : \langle first(s), j, last(s) \rangle \in DDS \\ False & \text{if } \forall j \in V(s), j \neq first(s), \\ & j \neq last(s) : \langle first(s), j, last(s) \rangle \notin DDS \end{cases}$$

Algorithm 4 TSP-LS heuristic

Input: Tập các điểm cần giao hàng, kho. Các tham số về chi phí và khoảng cách, thời gian chờ đợi, mức chịu đựng của drone.

Output: Chi phí nhỏ nhất thu được và một hành trình cho xe tải và drone.

```
1: Customers =  $N$ ;  
2: truckRoute = solveTSP( $N$ );  
3: truckSubRoutes = {truckRoute};  
4: sol = {truckRoute,  $\emptyset$ };  
5:  $i^* = -1$ ;  
6:  $j^* = -1$ ;  
7:  $k^* = -1$ ;  
8: maxSavings = 0;  
9: isDroneNode = null;  
10: Stop = false;  
11: repeat  
12:   for each  $j \in Customers$  do  
13:     savings=calcSaving( $j$ )  
14:     for each subroute in truckSubRoutes do  
15:       if drone(subroute,sol) then  
16:         (isDroneNode, maxSavings,  $i^*$ ,  $j^*$ ,  $k^*$ ) =  
17:           relocateAsTruck( $j$ , subroute, savings);  
18:       else  
19:         (isDroneNode, maxSavings,  $i^*$ ,  $j^*$ ,  $k^*$ ) =  
20:           relocateAsDrone( $j$ , subroute, savings);  
21:       end if  
22:     end for  
23:     if maxSavings>0 then  
24:       (sol, truckRoute, truckSubRoutes, Customers) = applyChange(isDroneNode,  
25:                      $i^*$ ,  $j^*$ ,  $k^*$ , sol, truckRoute, truckSubRoutes, Customers)  
26:       maxSavings = 0;  
27:     else  
28:       Stop = true;  
29:     end if  
30:   end for  
31: until Stop
```

Trong mỗi vòng lặp thực hiện hai bước: (1) cân nhắc xem có khách hàng nào khi di chuyển đến vị trí mới trong lộ trình tạo ra kết quả tốt nhất, (2) cân nhắc xem việc di chuyển khách hàng đó có làm lời giải hiện tại tốt lên không. Nếu có thì cập nhật lời giải hiện tại, *truckRoute* và *truckSubRoutes* và xóa khách hàng đó trong danh sách khách hàng, ngược lại việc di chuyển khách hàng không làm lời giải tốt lên thuật toán sẽ kết thúc. Cả hai bước trên được trình bày trong thuật toán 4, 5, 6, 7, 8. Bước 1 được thể hiện từ dòng 13 đến 20 trong thuật toán 4. Thuật toán cân nhắc mỗi khách hàng j và tính chi phí nếu xóa bỏ j khỏi vị trí hiện tại. Việc tính toán sẽ được trình bày trong thuật toán 5. Sau đó thuật toán sẽ thử đặt khách hàng j vào các *subroute*. Khi *subroute* đang xét có hai điểm đầu cuối lần lượt là điểm bay và điểm đón drone thì ta thử đặt j vào lộ trình xe tải trong *subroute*. Ngược lại ta sẽ thử đặt j vào một DD mới. Thuật toán tính toán chi phí khi đặt j vào TD và DD được trình bày bởi thuật toán 6, 7.

Algorithm 5 calcSavings(j)

Input: j : khách hàng trong TD

Output: Giá trị chi phí giảm đi.

```
1:  $i = prev_{truckRoute}(j);$ 
2:  $k = next_{truckRoute}(j);$ 
3:  $savings = (d_{i,j} + d_{j,k} - d_{i,k})C_1;$ 
4: return  $savings;$ 
```

Trong thuật toán 6 mục tiêu của chúng tôi là tìm ra vị trí tốt nhất trong *subroute s* để đặt khách hàng j . Việc này được thực hiện bởi việc đặt thử khách hàng j vào giữa các điểm liên tiếp nhau trong s . Đầu tiên chúng ta sẽ phải kiểm tra xem việc đặt j vào giữa cặp khách hàng i, k không vi phạm bất kì ràng buộc về thời gian chờ đợi nào trong các *subroute* khác. Sau đó ta kiểm tra việc đặt j vào phải mất chi phí nhỏ hơn *savings*. Vì tồn tại một nút DD nằm trên s nên ta phải kiểm tra xem việc đặt thêm j vào có làm vượt quá giới hạn năng lượng của drone không. Cuối cùng kiểm tra xem việc đặt j ở giữa i, k có phải là lời giải tốt nhất hiện tại không cập nhật giá trị mới cho $i^*, j^*, k^*, maxSavings$.

Algorithm 6 relocateAsTruck($j, s, savings$)

(Tính toán chi phí của việc chuyển khách hàng j đến vị trí mới trong lộ trình)

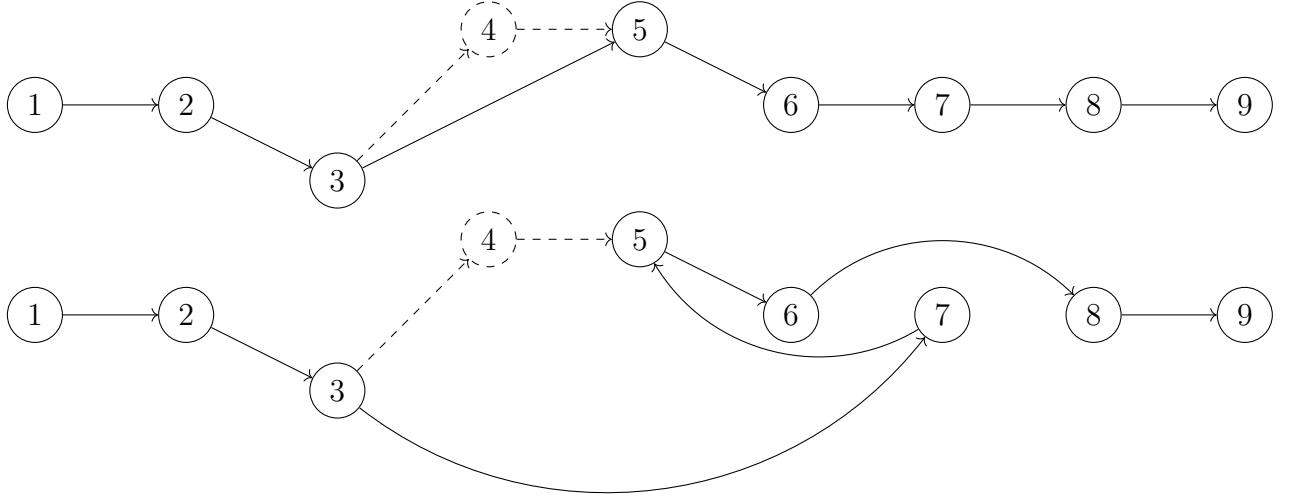
Input: j : khách hàng trong TD

s : *subroute* cần nhắc đặt khách hàng j

$savings$: lượng thay đổi nếu xóa bỏ j khỏi vị trí của nó

Output: Kết quả cập nhật vào các biến i^*, j^*, k^* và *isDroneNode*

```
1:  $a = first(s);$ 
2:  $b = last(s);$ 
3: for each  $(i, k) \in A(s)$  do
4:   if việc đặt  $j$  vào vị trí này không vi phạm bất kì ràng buộc về thời gian chờ đợi nào của các subroute khác then
5:      $\delta = (d_{i,j} + d_{j,k} - d_{i,k})C_1$ 
6:     if  $\delta < savings$  then
7:       if  $dist_T(s) + (d_{i,j} + d_{j,k} - d_{i,k}) < \epsilon$  then
8:         if  $savings - \delta > maxSavings$  then
9:            $isDroneNode = False;$ 
10:           $j^* = j; i^* = i; k^* = k;$ 
11:           $maxSaving = saving - \delta;$ 
12:        end if
13:      end if
14:    end if
15:  end if
16: end for
17: return (isDroneNode, maxSavings,  $i^*, j^*, k^*$ )
```



Hình 2.7: Mô phỏng thao tác relocateAsTruck

Trong thuật toán 7 chúng tôi tính toán chi phí nếu chuyển j thành khách hàng giao bởi một DD. Điều này được thực hiện bởi biến $\langle i, j, k \rangle$ thành một DD, với i, k là 2 điểm trong s . Tương tự với thuật toán 6 ta phải kiểm tra xem DD vừa tạo có thỏa mãn không. Nếu việc chuyển j vào một DD là lời giải tốt nhất hiện tại thì cập nhật giá trị mới cho $i^*, j^*, k^*, maxSavings$.

Algorithm 7 relocateAsDrone($j, s, savings$)

(Tính toán chi phí của việc chuyển khách hàng j sang giao hàng bằng drone)

Input: j : khách hàng trong TD.

s : subroute cân nhắc đặt khách hàng j .

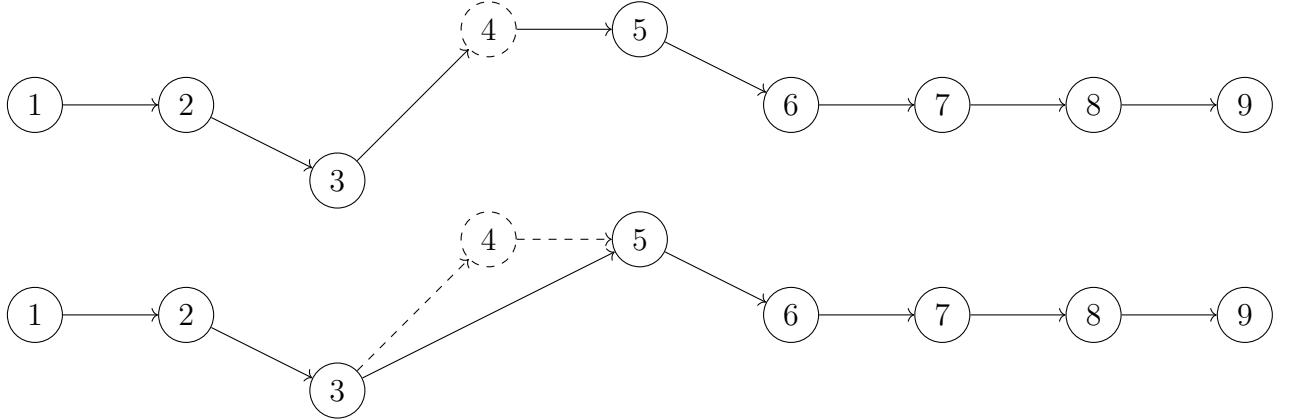
$savings$: lượng thay đổi nếu xóa bỏ j khỏi vị trí của nó.

Output: Kết quả cập nhật vào các biến i^* , j^* , k^* và $isDroneNode$

```

1: for  $i = 0, size(s) - 2$  do
2:   for  $k = i + 1, size(s) - 1$  do
3:     if  $\langle s[i], j, s[k] \rangle \in P$  then
4:        $\delta = (d'_{s[i],j} + d'_{j,s[k]})C_2;$ 
5:       if  $savings - \delta > maxSavings$  then
6:          $isDroneNode = True;$ 
7:          $j^* = j; i^* = s[i]; k^* = s[k];$ 
8:          $maxSavings = saving - \delta;$ 
9:       end if
10:      end if
11:    end for
12:  end for
13: return ( $isDroneNode, maxSavings, i^*, j^*, k^*$ )

```



Hình 2.8: Mô phỏng thao tác relocateAsDrone

Trong thuật toán 8 thực hiện công việc cập nhật lời giải nếu $maxSavings \neq 0$, vì vậy nếu $isDroneNode = true$ tức là thực hiện tạo một DD mới, ngược lại thực hiện thay đổi TD. Nếu là tạo một DD mới thì thuật toán thực hiện tạo DD $\langle i^*, j^*, k^* \rangle$ và sau đó cập nhật $truckRoute$, $subTruckRoutes$ sau đó xóa bỏ i^*, j^*, k^* khỏi tập $Customers$. Nếu là cập nhật TD thì ta chỉ cần thực hiện xóa bỏ j^* khỏi vị trí cũ và cập nhật j^* vào vị trí mới giữa i^* và k^* .

Quay lại thuật toán 4 thuật toán sẽ kiểm tra xem nếu $maxSaving \neq 0$ thì sẽ thực hiện $applyChanges$ còn không thì vòng lặp sẽ dừng lại.

Algorithm 8 applyChanges

Input: $isDroneNode, i^*, j^*, k^*, sol, truckRoute, truckSubRoutes, Customers$
Output: Cập nhật $truckRoute, truckSubRoutes, sol$

```

1: if  $isDroneNode == True$  then
2:     Thêm mới một DD  $i^* \rightarrow j^* \rightarrow k^*$ 
3:     Xóa bỏ  $j^*$  trong  $truckRoute$  và  $truckSubRoutes$ 
4:     Cập nhật  $truckSubRoutes$ 
5:     Xóa bỏ  $i^* j^* k^*$  trong  $Customers$ 
6: else
7:     Xóa bỏ  $j^*$  ở truck subroute hiện tại
8:     Cập nhật  $j^*$  vào vị trí mới giữa  $i^*$  và  $k^*$ 
9: end if
10: Cập nhật  $sol$ 
11: return ( $sol, truckRoute, truckSubRoutes, Customers$ )

```

Chương 3

Đề xuất thuật toán tìm kiếm cục bộ giải bài toán lập lộ trình vận tải giao hàng kết hợp một xe tải và nhiều thiết bị bay

Trong chương này chúng tôi đề xuất bài toán lập lộ trình vận tải giao hàng kết hợp một xe tải và nhiều thiết bị bay và một thuật toán tìm kiếm cục bộ giải quyết bài toán. Cụ thể, phần 3.1 sẽ giới thiệu và trình bày lý do xuất phát của bài toán lập lộ trình giao hàng kết hợp một xe tải và nhiều thiết bị bay không người lái, phần 3.2 sẽ trình bày phát biểu bài toán, biến và các mô hình, phần 3.3 chúng tôi sẽ trình bày một thuật toán tìm kiếm cục bộ giải quyết bài toán này.

3.1 Bài toán lập lộ trình giao hàng kết hợp xe tải và nhiều thiết bị bay

Trong phần trước chúng ta đã có cái nhìn tổng quan về bài toán lập lộ trình giao hàng kết hợp giữa xe tải và drone. Trong phần này chúng tôi trình bày một nâng cấp của bài toán min-cost TSP-D. Trong phần 2.1, chúng tôi cũng đã nêu lên các ưu điểm của drone so với xe tải và hiệu quả của phương pháp giao hàng kết hợp giữa xe tải và drone. Chính vì điều ấy, chúng tôi nhận thấy rằng nếu sử dụng nhiều hơn một thiết bị bay drone trong một lộ trình vận tải giao hàng cùng với xe tải có thể sẽ đem lại hiệu quả hơn nữa. Hơn nữa, trong thực tế một xe tải với khả năng của nó hoàn toàn có thể mang nhiều hơn một drone. Dựa trên những nhận xét ban đầu như vậy chúng tôi xây dựng một nâng cấp của bài toán min-cost TSPD là bài toán min-cost TSPkD.

3.2 Phát biểu bài toán

Một công ty phân phối hàng hóa có xe tải và đội drone. Trong đó một cặp xe tải và k drone ($k \in N$) thực hiện một lộ trình giao hàng cho n khách hàng sao cho thỏa mãn các yêu cầu sau.

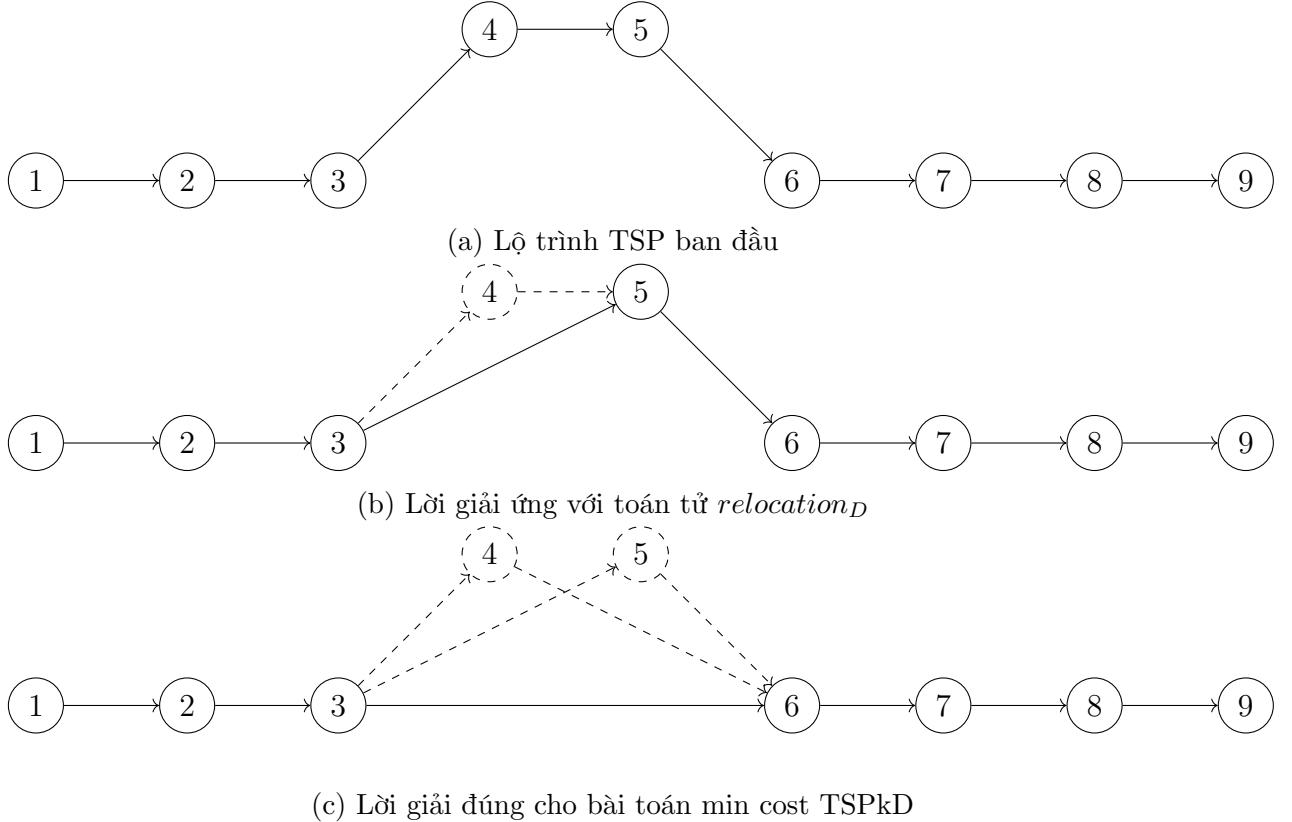
- Xe tải và các drone xuất phát từ điểm kho và quay lại kho khi hoàn thành lộ trình giao hàng.
- Drone luôn phải xuất phát và giao hàng từ xe tải, sau đó quay lại xe tải sau mỗi chuyến giao hàng.
- Điểm xuất phát và điểm kết thúc một chuyến giao hàng đều phải là một điểm khách hàng.
- Mỗi một chuyến giao hàng drone chỉ giao một khách hàng.
- Mỗi khách hàng chỉ được giao bởi drone hoặc xe tải và chỉ duy nhất một drone hoặc xe tải.
- Chuyến giao hàng của drone không được vượt quá mức năng lượng cho phép ε .
- Hai phương tiện phải chờ nhau nếu không đến cùng lúc tại điểm đón. Thời gian chờ nhau không được vượt quá Δ .
- Với một điểm khách hàng là điểm đón và điểm bay của nhiều chuyến drone. Xe tải luôn phải đợi tất cả drone về rồi mới thực hiện thả drone đi thực hiện các chuyến tiếp. Nếu điểm đó là điểm bay của nhiều drone thì các drone coi như được thả đi cùng một thời điểm.
- Tại một thời điểm bất kỳ không được có vượt quá k thiết bị bay cùng đang hoạt động.
- Chi phí trên một đơn vị độ dài của drone là C_2 , của xe tải là C_1 . Drone chỉ mất phí khi đang trong một chuyến giao hàng. Tổng chi phí của chuyến giao hàng bằng tổng chi phí của xe tải và drone cộng lại.
- Hàm mục tiêu cho bài toán là tối thiểu tổng chi phí của lộ trình giao hàng.

3.3 Thuật toán tìm kiếm cục bộ giải quyết bài toán lập lộ trình giao hàng kết hợp một xe tải và nhiều thiết bị bay

Nhận thấy bài toán min-cost TSPkD sẽ dần tới bài toán TSP khi ε tiến dần đến 0, vì vậy bài toán là NP-hard. Chính vì vậy chúng tôi lựa chọn hướng tiếp cận thuật giải gần đúng cụ thể là thuật toán tìm kiếm cục bộ để giải quyết bài toán. Trong phần 3.3.1 chúng tôi sẽ trình bày toán tử move mà chúng tôi sử dụng trong thuật toán, phần 3.3.2 là trình bày chi tiết thuật toán.

3.3.1 Toán tử move

Trong phần trước chúng tôi đã trình bày toán tử move $relocate_D$, trong đó thực hiện lấy một phần tử từ TD thiết lập một DD. Nếu ta đem y nguyên toán tử này sang áp dụng cho bài toán min-cost TSPkD thì gặp một vài bất cập. Cụ thể xem xét ví dụ trong hình 3.1.

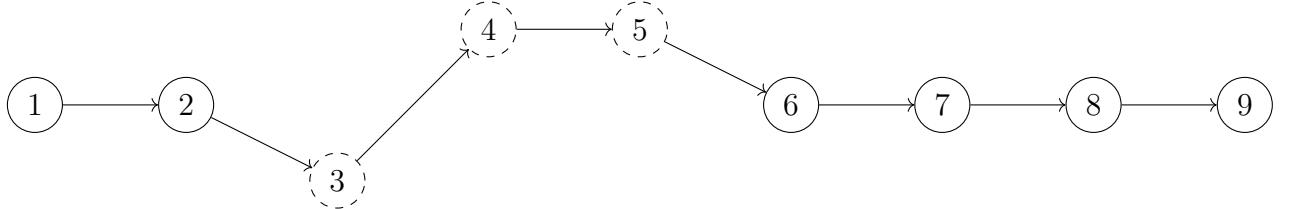


Hình 3.1: Ví dụ cho thao tác $relocate_D$ sang bài toán min-cost TSPkD

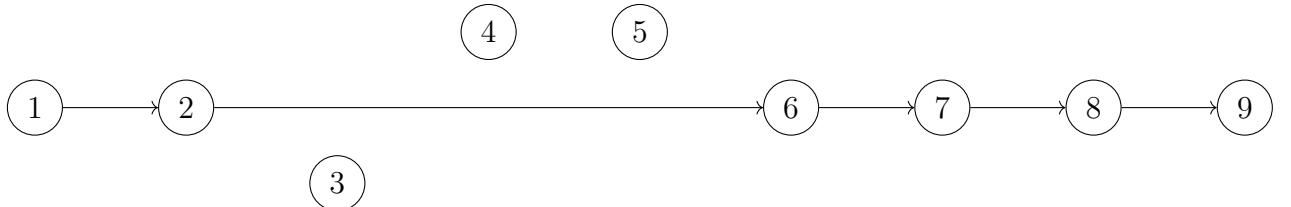
Với toán tử $relocate_D$ từ hành trình TSP ban đầu (hình 3.1a), ta lựa chọn điểm 4 là điểm khách hàng bằng xe tải cần chuyển sang giao hàng bằng drone. Trong bước chọn vị trí của điểm bay và điểm đón cho 4 thuật toán sẽ lựa chọn sao cho chi phí giảm được là lớn nhất, vì vậy 3 và 5 được chọn. Nhận thấy ngay, việc lựa chọn này sẽ khiến 5 trở thành một điểm đón drone và không thể chuyển thành điểm drone nữa. Để tránh trường hợp này chúng tôi đề xuất một toán tử mới trong đó cả điểm 4 và 5 đều được lấy ra khỏi lộ trình của xe tải.

Toán tử move- t -point thực hiện chuyển t điểm liên tiếp trong lộ trình thành t DD, các bước thực hiện như sau:

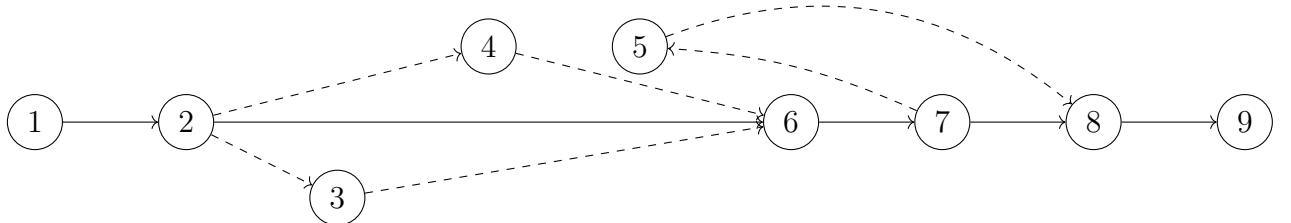
- Bước 1: Chọn t điểm giao hàng liên tiếp trong lộ trình $\langle v_1, v_2, \dots, v_t \rangle$
- Bước 2: Xóa bỏ t điểm giao hàng vừa chọn ra khỏi lộ trình.
- Bước 3: Với mỗi v_i ta thực hiện các bước:
 1. Tìm hai điểm v_{il} v_{ir} trong lộ trình sao cho lượng chi phí giảm được nhiều nhất.
 2. Tạo DD mới với v_{il} v_i v_{ir}



Hình 3.2: Chọn 3 điểm liên tiếp – thao tác move-3-point



Hình 3.3: Xóa bỏ 3 điểm vừa chọn ra khỏi lô trình – thao tác move-3-point



Hình 3.4: Chuyển các nút chọn thành DD – thao tác move-3-point

Mô phỏng một thao tác *move – 3 – point* được thể hiện qua các hình 3.2, 3.3, 3.4. Bước 1 ta chọn 3 điểm liên tiếp nhau, vd: điểm 3, 4, 5 (Hình 3.2). Bước 2 ta xóa bỏ 3 điểm vừa chọn ra khỏi lô trình (Hình 3.3). Bước 3 với điểm giao hàng 3:

1. Thực hiện chọn điểm bay và điểm đón cho DD có điểm giao hàng 3 là điểm 2 và điểm 6.
2. Thực hiện xây dựng DD tạo bởi 2, 3 và 6.

Lặp lại bước 3 với điểm 4, 5.

3.3.2 Thuật toán tìm kiếm cục bộ

Trong phần này chúng tôi trình bày thuật toán tìm kiếm cục bộ sử dụng các thao tác move trình bày trong các phần trước. Ý tưởng của thuật toán như sau: Bắt đầu từ một hành trình của bài toán người du lịch, thuật toán sử dụng thao tác move sao cho không còn thao tác move nào có thể làm giảm chi phí nữa, cụ thể được trình bày trong thuật toán 9.

Algorithm 9 TSPkD-LS heuristic

Input: Tập các điểm cần giao hàng, kho, các tham số về chi phí và khoảng cách, thời gian chờ đợi, mức chịu đựng của drone.

Output: Chi phí nhỏ nhất thu được và một hành trình cho xe tải và drone.

```
1: truckRoute = solveTSP(N);
2: tour = (truckRoute, droneDeliveryList);
3: maxSavings = 0;
4: Stop = false;
5: repeat
6:   if maxSaving < relocateT(tour) then
7:     maxSaving = relocateT(tour);
    Lưu vị trí relocateT
8:   else if maxSaving < relocateD(droneDeliveryList) then
9:     maxSaving = relocateD(droneDeliveryList);
    Lưu vị trí relocateD
10:  else if maxSaving < removeD(tour) then
11:    maxSaving = removeD(tour)
    Lưu vị trí removeD
12:  else if maxSaving < two_exchange(tour) then
13:    maxSaving = two_exchange(tour);
    Lưu vị trí two_exchange
14:  else if maxSaving < kpointmove(tour) then
15:    maxSaving = move_t_point(tour);
    Lưu vị trí move_t_point
16:  end if
17:  if maxSavings > 0 then
18:    Cập nhật lại trạng thái của tour theo toán tử move cho kết quả tốt nhất
19:    maxSavings = 0;
20:  else
21:    Stop = true;
22:  end if
23: until Stop
```

Trong thuật toán 9 chúng tôi thực hiện tính toán chi phí giảm đi với tất cả các toán tử, sau đó lựa chọn toán tử cho chi phí giảm nhiều nhất và chuyển bài toán sang trạng thái mới. Dòng 6-18 là lựa chọn toán tử cho chi phí giảm nhiều nhất. Các toán tử *relocate_D*, *relocate_T*, *remove_D*, *two_exchange* được trình bày từ các phần trước chúng tôi xin phép không trình bày lại trong phần này. Mã giả toán tử *move-t-point* được trình bày trong thuật toán 11.

Algorithm 10 caculateSavingPoints(i, t)

Input: i vị trí điểm đầu tiên được lấy ra trong $truckRoute$, t số điểm được lấy ra trong $truckRoute$

Output: Chi phí giảm đi sau khi lấy t điểm bắt đầu từ i ra khỏi $truckRoute$

```
1: savings = 0;  
2: for  $j = i, i + t - 1$  do  
3:    $p = prev_{truckRoute}(j);$   
4:    $q = next_{truckRoute}(j);$   
5:   savings = savings +  $d_{p,q}C_1$   
6: end for  
7:  $p = prev_{truckRoute}(i);$   
8:  $q = next_{truckRoute}(i + t - 1);$   
9: savings = savings -  $d_{p,q}C_1$ ;  
return savings.
```

Ý tưởng của thủ tục *move_t_point* 11 như sau. Chúng tôi xem xét tất cả các *move_t_point* với $t \leq maxRangeMove$, vd: với $maxRangeMove = 3$ chúng tôi sẽ xem xét *move - 1 - point*, *move - 2 - point*, *move - 3 - point*. Cụ thể với mỗi i ta thực hiện lựa chọn lần lượt các chuỗi lộ trình con liên tiếp trong $truckRoute$ gồm i điểm. Nếu có một điểm đã là điểm bay hay điểm hạ cánh của một DD thì không xét chuỗi đó và thực hiện chuỗi tiếp theo (dòng 7-9). Ngược lại, ta sẽ thực hiện thiết lập DD cho từng điểm trong chuỗi lộ trình (dòng 10- 24). Sau khi thiết lập hết các DD ta kiểm tra xem chi phí giảm thu được có lớn nhất hiện tại không. Nếu có, ta lưu lại chi phí và chuỗi điểm.

Hàm *relocateAsDrone* là thủ tục tính toán chi phí giảm đi khi chuyển một điểm thuộc TD lên thành một DD. Hàm này chúng tôi làm tương tự hàm *relocateAsDrone* trong thuật toán 4, chỉ khác để phù hợp với bài toán min-cost TSPkD chúng ta phải cài đặt lại các hàm kiểm tra ràng buộc trong thủ tục. Hàm *caculateSavingPoints(j, i)* là hàm tính chi phí giảm đi khi $truckRoute$ bị lấy i phần tử từ j được trình bày trong thuật toán 10.

Algorithm 11 move_t_point

Input: tour, maxRangeMove

Output: Chi phí và vị trí giảm nhiều nhất.

```
1: moveMaxSaving=0;
2: truckSubRoutes = {truckRoute};
3: for  $i = 1, maxRangeMove$  do
4:   for  $j = 1, size(truckRoute) - i + 1$  do
5:     savings = caculateSavingPoints( $j, i$ );
6:     Lấy toàn bộ  $i$  point từ điểm  $j$  ra khỏi truckRoute.
7:     if Có một điểm không hợp lệ then
8:       continue;
9:     end if
10:    for  $h = j, j + i$  do
11:      aPointMaxSaving = 0;
12:       $la^* = -1$ ;
13:       $dr^* = -1$ ;
14:       $re^* = -1$ ;
15:      for each subroute in truckSubRoutes do
16:        aPointSavings = 0;
17:        ( $pointmaxSavings, la, dr, re$ ) =
18:          relocateAsDrone( $h, subroute, aPointSavings$ );
19:        if  $aPointMaxSaving < pointmaxSavings$  then
20:          aPointMaxSaving =  $pointmaxSavings$ ;
21:           $la^* = la$ ;
22:           $dr^* = dr$ ;
23:           $re^* = re$ ;
24:        end if
25:      end for
26:      savings = savings + aPointMaxSaving;
27:      Thêm DD vào danh sách DD cho move.
28:    end for
29:    if  $moveMaxSaving < savings$  then
30:      moveMaxSaving = savings;
31:      Lưu trạng thái đạt moveMaxSavings.
32:    end if
33:    Đưa truckRoute về trạng thái ban đầu.
34:  end for
35: return moveMaxSavings và trạng thái đạt moveMaxSavings.
```

Chương 4

Thử nghiệm và đánh giá

Trong chương này chúng tôi thực hiện đánh giá các thuật toán. Cụ thể trong đồ án chúng tôi cài đặt 2 thuật toán là *TSP – LS Heuristic* cho bài toán min-cost TSPD 1 drone và thuật toán *TSPkD – LS* cho bài toán min-cost TSPkD 2, 3, 4 drones. Thực nghiệm được thực hiện trên hai bộ dữ liệu một bộ theo bài báo [9] và một bộ sinh bằng cách chọn ngẫu nhiên trên Googlemap.

4.1 Dữ liệu

Các tham số của mỗi bộ dữ liệu cụ thể như sau:

- Bộ 1 (theo bài báo [9]):

- Tốc độ của xe tải: 40 km/h
- Tốc độ của drone: 40 km/h
- Chi phí trên 1km của xe tải: 25
- Chi phí trên 1km của drone: 1
- Khoảng thời gian tối đa hai phương tiện đợi nhau: 99999999 phút
- Sức chịu đựng của drone: 13,33333 km
- Bao gồm 35 tập dữ liệu với số điểm là 10 và 50.

Chú ý: Bộ dữ liệu có định nghĩa sẵn tập điểm giao hàng mà được phép giao hàng bởi drone.

- Bộ 2:

- Tốc độ của xe tải: 40 km/h
- Tốc độ của drone: 40 km/h
- Chi phí trên 1km của xe tải: 25
- Chi phí trên 1km của drone: 1
- Khoảng thời gian tối đa hai phương tiện đợi nhau: 15 phút
- Sức chịu đựng của drone: 15 km
- Bao gồm 10 tập dữ liệu với số điểm lần lượt 10, 20, 30, 40, 50, tương ứng một số điểm có 2 tập dữ liệu.

4.2 Kết quả thử nghiệm

Thuật toán được cài đặt bằng ngôn ngữ lập trình Java, chạy trên máy sử dụng hệ điều hành Window, cấu hình máy :

- CPU: Intel Core i5-2520M CPU @ 2,25 GHz.
- Memory: 8Gb.

Trong thực nghiệm chúng tôi chọn bộ tham số như sau :

- Với thư viện CBLSVR:
 - MaxStable 50
 - MaxIter 300
 - TimeLimit 10
- Với TSPD :
 - maxRangeMove 7

Bảng 4.1 chúng tôi thể hiện kết quả của bài toán *min-cost TSPD* với 1, 2, 3, 4 drone. Kết quả của bài toán *tspd -1- drone* được chạy bởi thuật toán *TSP-LS Heuristic* được trình bày trong phần 2.3. Các kết quả ứng với bài toán 2, 3, 4 drone được chạy bởi thuật toán được trình bày trong chương 3. Trong biểu đồ 4.1 thể hiện % chênh lệch của bài toán min-cost TSPkD với 2,3,4 drone so với bài toán 1 drone. Công thức tính phần trăm chênh lệch như sau:

$$\% = \frac{\text{Chi phí của bài toán } k \text{ drones} - \text{Chi phí của bài toán 1 drone}}{\text{Chi phí của bài toán 1 drone}} \times 100$$

Bảng 4.1: Bảng kết quả bộ dữ liệu 2 ứng với 1, 2, 3, 4 drones

STT	Tập dữ liệu	tspd-1-drone		tspd-2-drone		tspd-3-drone		tspd-4-drone	
		Chi phí	Thời gian (ms)						
0	<i>mbA101</i>	990.53	1.00	990.53	0.00	990.53	1.00	990.53	0.00
1	<i>mbA102</i>	953.64	3.00	953.64	0.00	953.64	1.00	953.64	0.00
2	<i>mbA103</i>	985.68	1.00	985.68	0.00	985.68	1.00	985.68	0.00
3	<i>mbA104</i>	911.87	1.00	911.87	1.00	911.87	0.00	911.87	1.00
4	<i>mbA105</i>	911.67	2.00	910.55	0.00	910.55	1.00	910.55	0.00
5	<i>mbB101</i>	2086.51	228515.00	2099.51	488.00	2099.51	286.00	2099.51	382.00
6	<i>mbB102</i>	1943.58	641944.00	1958.00	264.00	1958.00	308.00	1958.00	276.00
7	<i>mbB103</i>	1925.97	471074.00	1872.30	350.00	1871.79	465.00	1871.10	480.00
8	<i>mbB104</i>	2160.91	160644.00	2122.00	57.00	2122.00	68.00	2122.00	81.00
9	<i>mbB105</i>	1983.99	378731.00	1969.16	148.00	1969.16	123.00	1969.16	126.00
10	<i>mbB106</i>	2081.65	322429.00	2081.65	83.00	2081.65	80.00	2081.65	90.00
11	<i>mbB107</i>	2066.85	253436.00	2064.72	248.00	2064.72	298.00	2064.72	293.00
12	<i>mbB108</i>	2013.60	475925.00	1969.65	220.00	1969.65	242.00	1969.65	243.00
13	<i>mbB109</i>	1967.11	1026864.00	1946.92	298.00	1946.92	293.00	1946.92	295.00
14	<i>mbB110</i>	2047.90	177981.00	1947.27	367.00	1947.57	350.00	1945.99	375.00

15	<i>mbC101</i>	3964.19	8536.00	3963.72	61.00	3963.72	82.00	3963.72	83.00
16	<i>mbC102</i>	3732.77	6601.00	3732.77	160.00	3732.77	182.00	3732.77	166.00
17	<i>mbC103</i>	3713.82	15427.00	3661.81	417.00	3661.81	490.00	3661.81	450.00
18	<i>mbC104</i>	4103.35	11749.00	4058.82	184.00	4058.82	233.00	4058.82	224.00
19	<i>mbC105</i>	3848.47	23550.00	3676.27	606.00	3664.47	662.00	3664.47	892.00
20	<i>mbC106</i>	4041.45	22180.00	3957.82	491.00	3957.82	354.00	3957.82	354.00
21	<i>mbC107</i>	4146.52	10660.00	4145.69	109.00	4145.69	140.00	4145.69	146.00
22	<i>mbC108</i>	3924.68	26514.00	3876.36	316.00	3874.41	359.00	3874.41	439.00
23	<i>mbC109</i>	4044.95	17850.00	3895.35	246.00	3894.37	262.00	3894.37	305.00
24	<i>mbC110</i>	3876.91	18155.00	3870.08	141.00	3870.08	166.00	3870.08	160.00
25	<i>mbD101</i>	5559.54	7470.00	5546.41	216.00	5546.41	338.00	5546.41	292.00
26	<i>mbD102</i>	6033.59	3932.00	6069.41	108.00	6069.41	142.00	6069.41	146.00
27	<i>mbD103</i>	5603.71	5648.00	5603.71	311.00	5603.71	241.00	5603.71	234.00
28	<i>mbD104</i>	5704.79	13405.00	5432.29	527.00	5432.29	752.00	5432.29	711.00
29	<i>mbD105</i>	5726.94	4721.00	5726.94	97.00	5726.94	108.00	5726.94	102.00
30	<i>mbD106</i>	5561.71	9631.00	5561.71	425.00	5561.71	356.00	5561.71	366.00
31	<i>mbD107</i>	5869.62	3879.00	5867.98	110.00	5867.98	113.00	5867.98	110.00
32	<i>mbD108</i>	5193.21	9347.00	5187.36	355.00	5179.85	415.00	5179.85	509.00
33	<i>mbD109</i>	5990.58	6391.00	5891.65	177.00	5851.43	326.00	5851.43	414.00
34	<i>mbD110</i>	5518.99	4487.00	5308.85	141.00	5308.85	136.00	5308.85	143.00

Bảng 4.2: Bảng kết quả bộ dữ liệu 2 ứng với 1, 2, 3, 4 drones

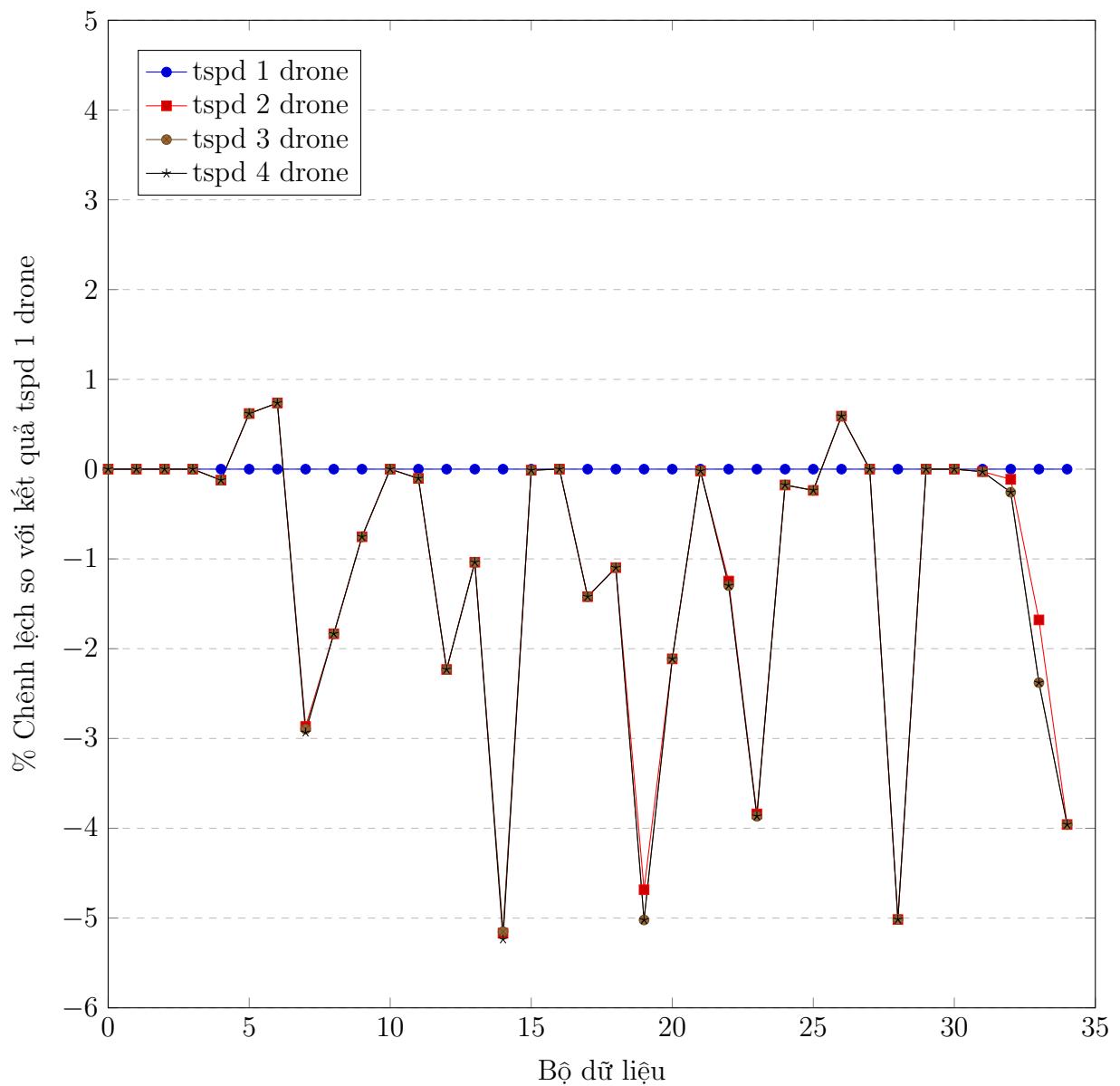
STT	Tập dữ liệu	tspd-1-drone		tspd-2-drone		tspd-3-drone		tspd-4-drone	
		Chi phí	Thời gian (ms)						
0	<i>data10₁</i>	471	3	346	5	345	3	345	2
1	<i>data10₂</i>	908	3	560	5	617	6	616	5
2	<i>data20₁</i>	1123	721	553	44	552	35	551	80
3	<i>data20₂</i>	1716	336	952	48	952	61	953	63
4	<i>data30₁</i>	1438	18864	1132	312	1048	304	1004	350
5	<i>data30₂</i>	2149	15166	1714	292	1713	322	1712	374
6	<i>data40₁</i>	1254	245945	970	1487	998	3269	965	3049
7	<i>data40₂</i>	1339	334655	1125	979	1131	1271	1119	1402
8	<i>data50₁</i>	2418	874590	1500	3454	1523	5536	1444	4055
9	<i>data50₂</i>	3779	294517	2864	6131	2859	4897	2857	5954

Kết quả trong bảng 4.1 và biểu đồ 4.1 cho thấy rằng việc sử dụng nhiều hơn một drone kết quả thu được là tốt hơn. Cụ thể hầu hết các bộ dữ liệu bài toán ứng với 2, 3, 4 drone cho kết quả bằng hoặc tốt hơn 1 drone (trừ mbB101, mbB102, mbD102). Với bộ dữ liệu 2 ứng với tập dữ liệu 10 đến 50 điểm bao gồm cả kho. Kết quả chi phí chênh lệch thu được được thể hiện trong đồ thị 4.2.

Trong bảng 4.2 chúng tôi trình bày kết quả của thuật toán với bộ dữ liệu 2. Tương tự như 4.1, kết quả của bài toán *tspd -1 drone* được chạy bởi thuật toán *TSP-LS Heuristic*, kết quả ứng với bài toán 2, 3, 4 drone được chạy bởi thuật toán được trình bày trong 3. Bộ dữ liệu 2 ứng với tập dữ liệu 10 đến 50 điểm bao gồm cả kho. Kết quả chi phí chênh lệch thu được được thể hiện trong đồ thị 4.2.

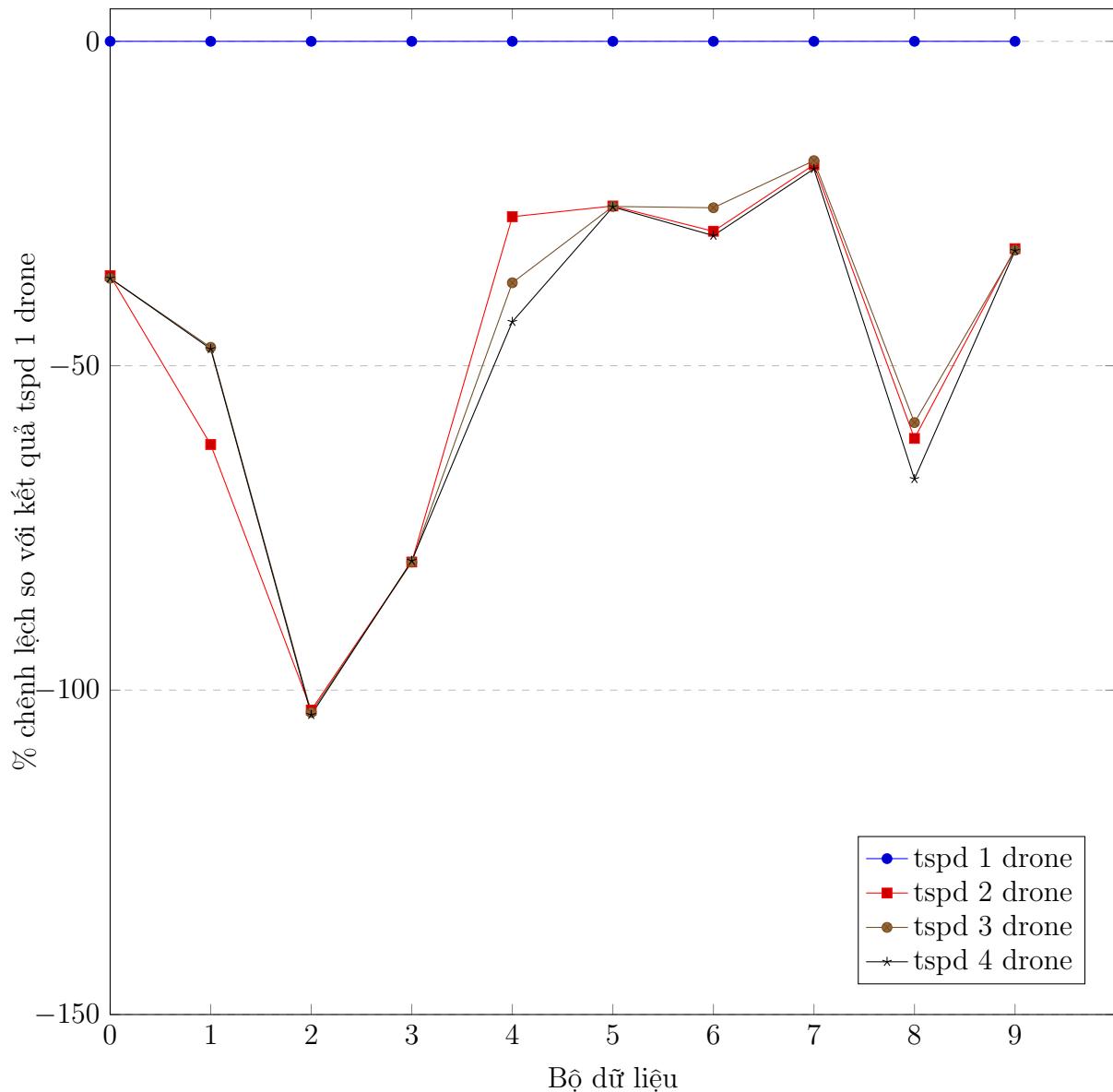
Nhìn vào kết quả thu được trong bảng 4.2 và biểu đồ 4.2 chúng tôi cũng thêm một lần nữa khẳng định việc sử dụng nhiều hơn 1 drone thu được kết quả tốt hơn sử dụng chỉ 1 drone. Cụ thể kết quả ứng với bài toán 2, 3, 4 drone tốt hơn hẳn so với 1 drone trong

Biểu đồ thể hiện kết quả của k drone so với một drone (Bộ dữ liệu 1)



Hình 4.1: Biểu đồ thể hiện kết quả thuật toán với 1, 2, 3, 4 drone- Bộ dữ liệu 1

Biểu đồ thể hiện kết quả của k drone so với một drone (Bộ dữ liệu 2)



Hình 4.2: Biểu đồ thể hiện kết quả thuật toán với 1,2,3,4 drone - Bộ dữ liệu 2

tất cả các bộ dữ liệu. Hơn nữa thời gian chạy cũng thấp hơn do sinh một lúc nhiều DD sẽ hạn chế được số vòng lặp phải thực hiện. Xét riêng với 2, 3, 4 drone kết quả chưa thực sự rõ rệt. Điều này cho thấy thuật toán có độ ổn định chưa thực sự tốt.

Bảng 4.3: Bảng kết quả bộ dữ liệu 2 ứng move 1, 2, 3, 4 point cho bài toán min-cost TSPkD 2 drones

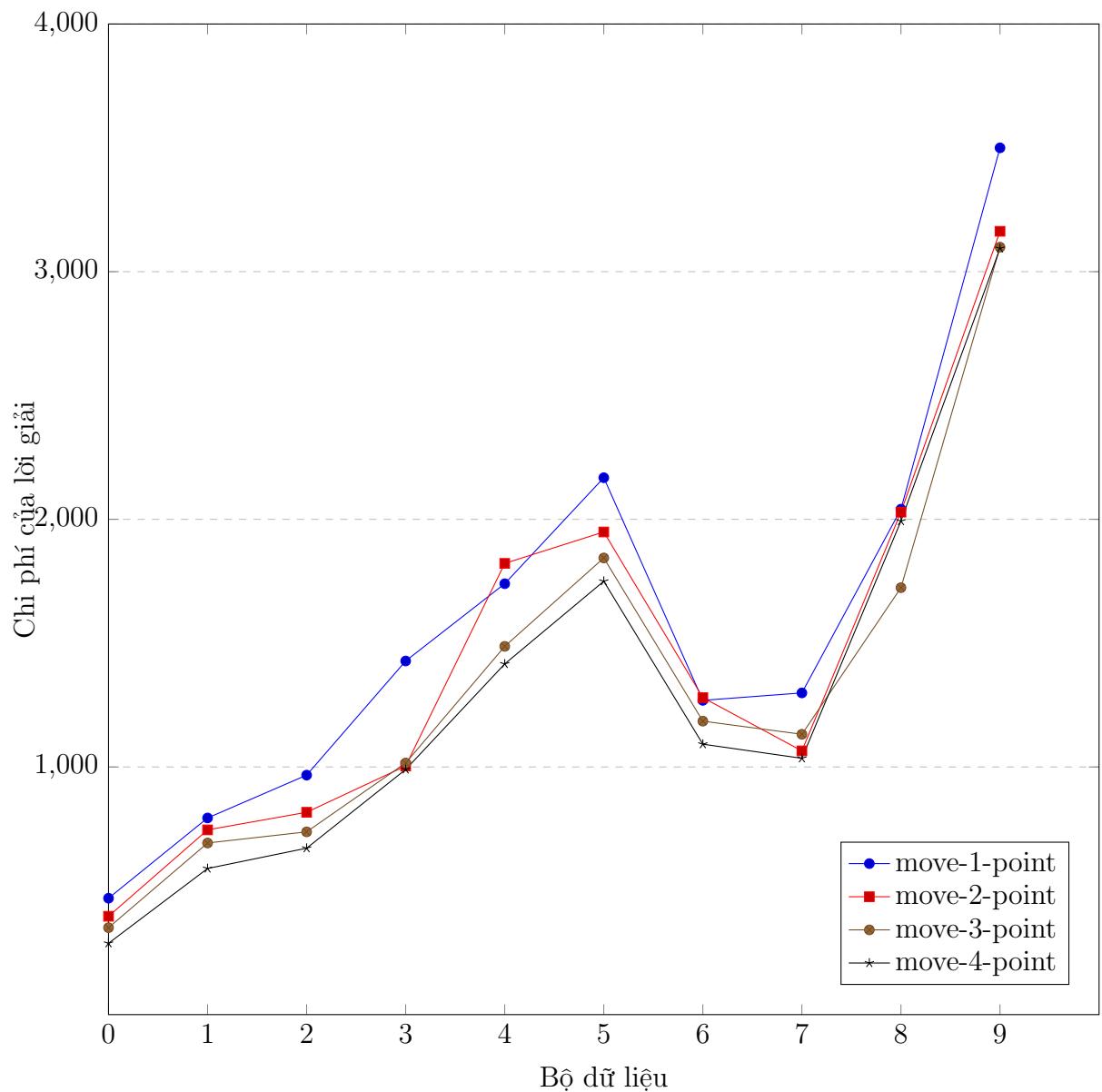
STT	Tập dữ liệu	move-1-point		move-2-point		move-3-point		move-4-point	
		Chi phí	Thời gian (ms)						
0	$data10_1$	470	1	397	1	351	3	288	1
1	$data10_2$	794	1	746	2	693	4	590	2
2	$data20_2$	1428	17	1004	19	1016	114	990	39
3	$data20_1$	967	19	817	17	738	37	672	38
4	$data30_1$	1740	100	1822	153	1487	376	1416	209

5	<i>data30₂</i>	2168	96	1949	146	1844	343	1750	235
6	<i>data40₁</i>	1269	294	1280	443	1185	1073	1092	985
7	<i>data40₂</i>	1299	482	1065	506	1132	1263	1035	959
8	<i>data50₁</i>	2041	1407	2029	2005	1724	4635	1993	2330
9	<i>data50₂</i>	3500	1334	3163	1966	3099	3868	3094	2394

Chúng tôi thực hiện khảo sát toán tử $move - t - point$ để khẳng định chắc chắn hơn rằng việc áp dụng nhiều hơn 1 điểm sẽ đem lại kết quả tốt hơn là sử dụng $relocate_D$ như bài toán 1 drone. Chúng tôi thực hiện khảo sát trên bài toán min-cost TSP2D (2 drone) với bộ dữ liệu 2 lần lượt các toán tử $move - 1 - drone$, $move - 2 - drone$, $move - 3 - drone$, $move - 4 - drone$. Tất nhiên trong phần này chúng tôi chỉ sử dụng đơn thuần toán tử $move - t - point$. Trong bảng 4.3 là chi phí và thời gian của bộ dữ liệu 2 khi sử dụng các toán tử $move - 1, 2, 3, 4 - point$. Biểu đồ 4.3 thể hiện chi phí của các toán tử move ứng với từng bộ dữ liệu.

Kết quả trong bảng 4.3 và biểu đồ 4.3 khẳng định thêm lại giả thiết đưa ra ở phần 3.3.1 về việc sử toán tử $move - t - point$ với $t > 1$ sẽ cho kết quả tốt hơn với bài toán min-cost TSPkD. Cụ thể với các bộ dữ liệu kết quả của $move - t - point$ với $t > 1$ cho kết quả tốt hơn hoặc bằng $t = 1$ (trừ trường hợp ở bộ dữ liệu *data30₁* chi phí $move - 2 - point$ lớn hơn $move - 1 - point$).

Biểu đồ thể hiện kết quả thuật toán với $move - 1, 2, 3, 4 - point$ - Bộ dữ liệu 2



Hình 4.3: Biểu đồ thể hiện kết quả thuật toán với $move - 1, 2, 3, 4 - point$ - Bộ dữ liệu 2

Chương 5

Thiết kế và xây dựng chương trình ứng dụng

Trong đồ án này chúng tôi xây dựng một chương trình ứng dụng web hỗ trợ người sử dụng lập kế hoạch vận chuyển hàng hóa. Kiến trúc của ứng dụng gồm 2 phần: (1) một web service chạy thuật toán, (2) một web-app cho nghiệp vụ và hiển thị sử dụng công nghệ ofbiz. Trong phần này chúng tôi xin phép chỉ trình bày module (2).

5.1 Công nghệ sử dụng

Công nghệ sử dụng chính trong xây dựng ứng dụng web là Apache Ofbiz. Ngoài ra chúng tôi còn sử dụng các công nghệ hiển thị để đem lại tính tiện dụng cho người dùng như GoogleMap hay bootstrap.

5.1.1 Công nghệ ofbiz

Ofbiz xây dựng ứng dụng dựa trên mô hình MVC vì vậy việc xây dựng các ứng dụng trong ofbiz cũng phải tuân thủ theo mô hình này.

Các thành phần trong Ofbiz

- *framework* là nơi chứa các thành phần hoạt động chính của ofbiz như: kết nối cơ sở dữ liệu, caching, render screens, quản lý giao dịch Đây là component được load đầu tiên khi hệ thống khởi động.
- *applications* đây là thành phần core của ofbiz nơi chứa các component nghiệp vụ như, quản lý nội dung, quản lý đơn hàng, . . .
- *specialpurpose* bao gồm thêm các component và ứng dụng của ofbiz.
- *themes* bao gồm các resource cần thiết giao diện trong ofbiz.
- *hotdeploy* sử dụng để tạo các component mới cho người sử dụng.

Để tạo một component mới trong hotdeploy ta đơn giản chỉ chạy task ant *create-component* và nhập các thông tin cần thiết hệ thống sẽ tạo một thư mục là tên component vừa nhập vào trong hotdeploy.

Các thành phần quan trọng trong một component

- *config* chứa các config cho một component vd: *SlpUiLabels.xml* chứa config về ngôn ngữ trong ofbiz, phục vụ cho chuyển đổi ngôn ngữ trong trang web.
- *entitydef* chứa các định nghĩa về entity map với các bảng trong cơ sở dữ liệu.
- *lib* chứa các thư viện java sử dụng component.
- *servicedef* chứa các định nghĩa service.
- *src* chứa code được sử dụng trong component.
- *webapp* chứa controller, các view, resource của component.
- *widget* chứa các định nghĩa về screen của component.

Các bước để tạo một luồng hoạt động trong ofbiz:

1. Bước 1: Tạo một định nghĩa mapping trong controller.xml.
2. Bước 2: Định nghĩa các service cần thiết cho mapping (nếu cần).
3. Bước 3: Định nghĩa screen cho mapping (nếu cần).
4. Bước 4: Tạo view .ftl cho mapping vừa tạo.

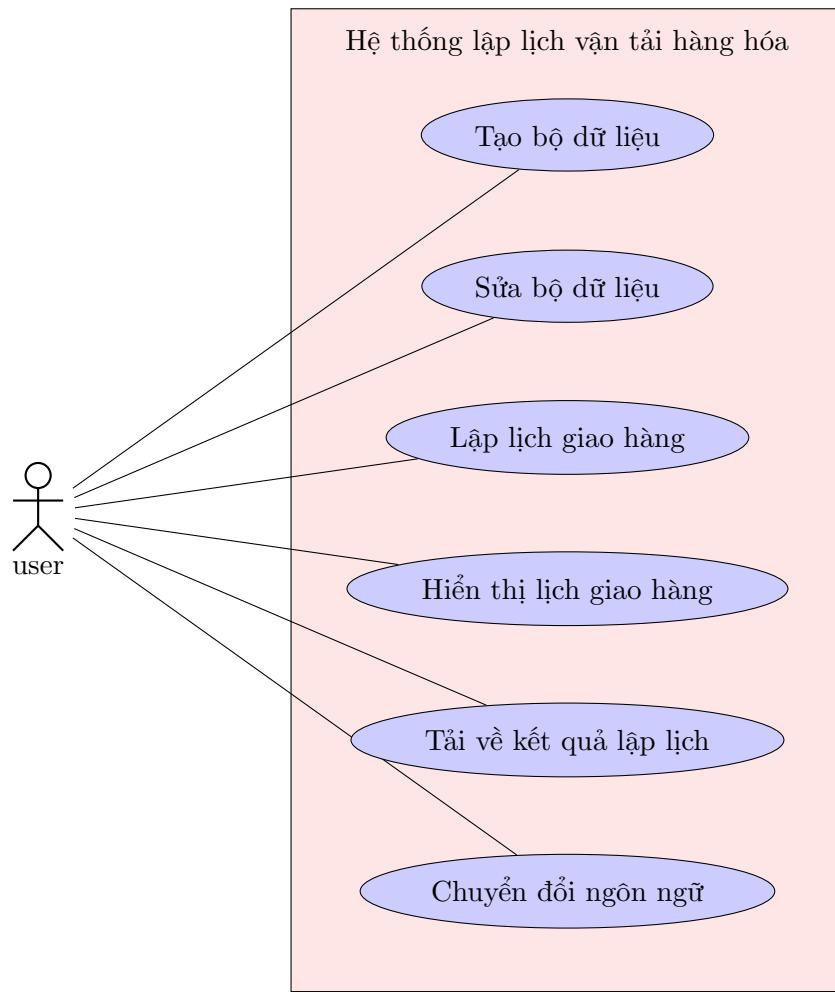
5.1.2 GoogleMap API

Trong đồ án này chúng tôi sử dụng API của Google Map để truy vấn thông tin thời gian quãng đường cho các điểm khách hàng và hiển thị giao diện web cho người sử dụng dễ thao tác. Google map java client được sử dụng trong server để truy vấn khoảng cách điểm, JavaScript api được sử dụng ở client để hiển thị bản đồ cũng như những thao tác với bản đồ.

5.1.3 Bootstrap

Bootstrap là một framework kết hợp html, css và javascript cho phép người sử dụng xây dựng view một cách đơn giản hơn mà vẫn đảm bảo responsive. Bootstrap có một tập các thành phần định sẵn cho người sử dụng như layout, button, các ô input, tables ...

5.2 Sơ đồ use-case



Hình 5.1: Sơ đồ use-case của ứng dụng

Trong đồ án này chúng tôi xây dựng một ứng dụng cơ bản cho chức năng cơ bản, cho phép lập lịch vận tải hàng hóa bao gồm lên kế hoạch lập lịch và tải về kết quả. Cụ thể được liệt kê trong hình 5.1:

- Case tạo bộ dữ liệu cho phép người sử dụng tạo một tập điểm cần giao hàng và điểm kho trên bản đồ sau đó lưu lại.
- Case sửa bộ dữ liệu cho phép người sử dụng sửa những tập điểm giao hàng trước đó, thêm điểm, chỉnh sửa vị trí điểm.
- Case lập lịch giao hàng cho phép chọn một tập dữ liệu trong hệ thống và chạy lập lịch.
- Case hiển thị lịch giao hàng cho phép người sử dụng nhìn thấy kết quả của case lập lịch giao hàng một cách động. Người sử dụng cũng có thể tải lên một kết quả có sẵn từ trước.
- Case tải về kết quả lập lịch cho phép người sử dụng tải về kết quả sau khi đã lập lịch.

- Case chuyển đổi ngôn ngữ cho phép chuyển ngôn ngữ của ứng dụng sang tiếng Anh.

5.3 Cơ sở dữ liệu

Nhằm mục đích để lưu trữ các tập điểm yêu cầu của khách hàng, chúng tôi tạo ra một cơ sở dữ liệu [5.2](#) gồm 3 bảng:

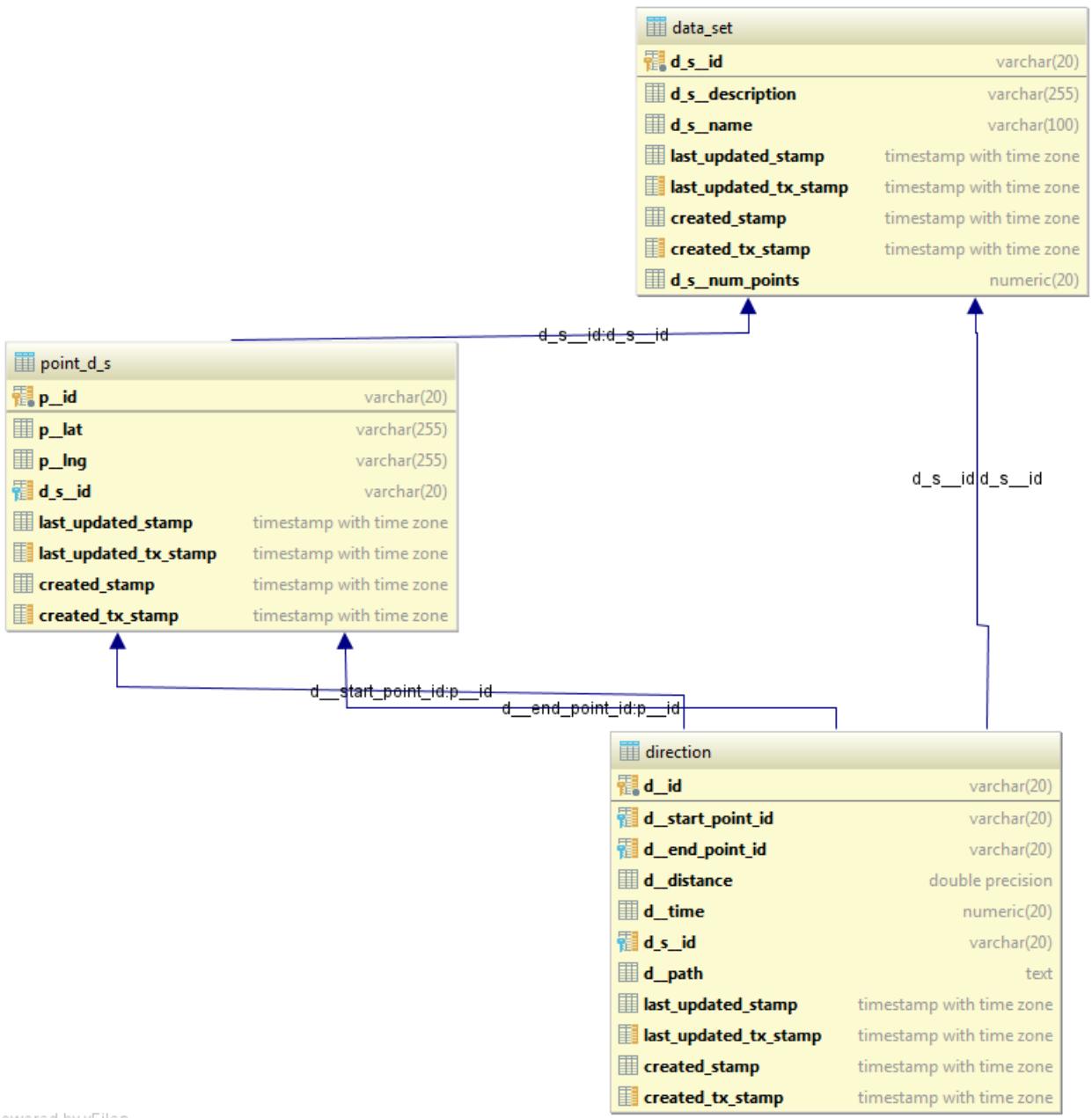
- Bảng point: chứa dữ liệu của một điểm giao hàng của khách hàng bao gồm các thông tin về tọa độ.
- Bảng direction: chứa dữ liệu về khoảng cách, thời gian, chuỗi đường đi giữa hai điểm khách hàng.
- Bảng dataset: chứa thông tin về tập điểm yêu cầu.

Trong data, chúng tôi lưu trữ mỗi dataset gồm nhiều điểm yêu cầu, cứ hai điểm trong dataset thì có một direction chứa thông tin giữa chúng. Thông tin direction được lấy từ GoogleMap.

5.4 Các màn hình

Ứng dụng được thiết kế gồm các màn hình sau:

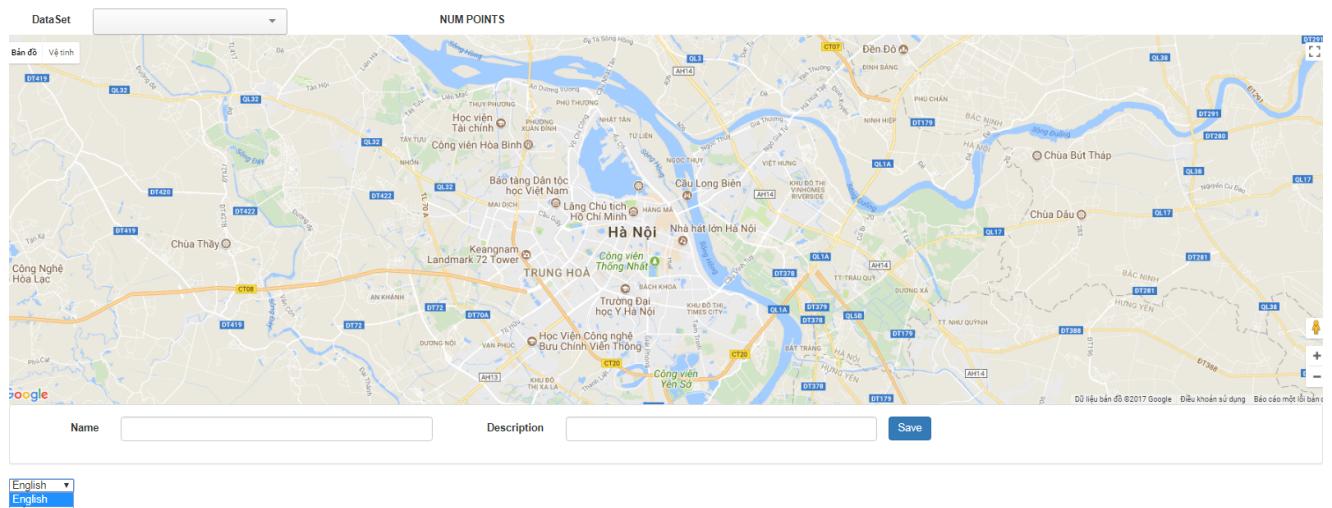
- Màn hình tạo và sửa bộ dữ liệu ([5.4](#), [5.5](#)).
- Màn hình lập lịch ([5.6](#)).
- Màn hình hiển thị kết quả ([5.7](#), [5.8](#)).



Powered by yFiles

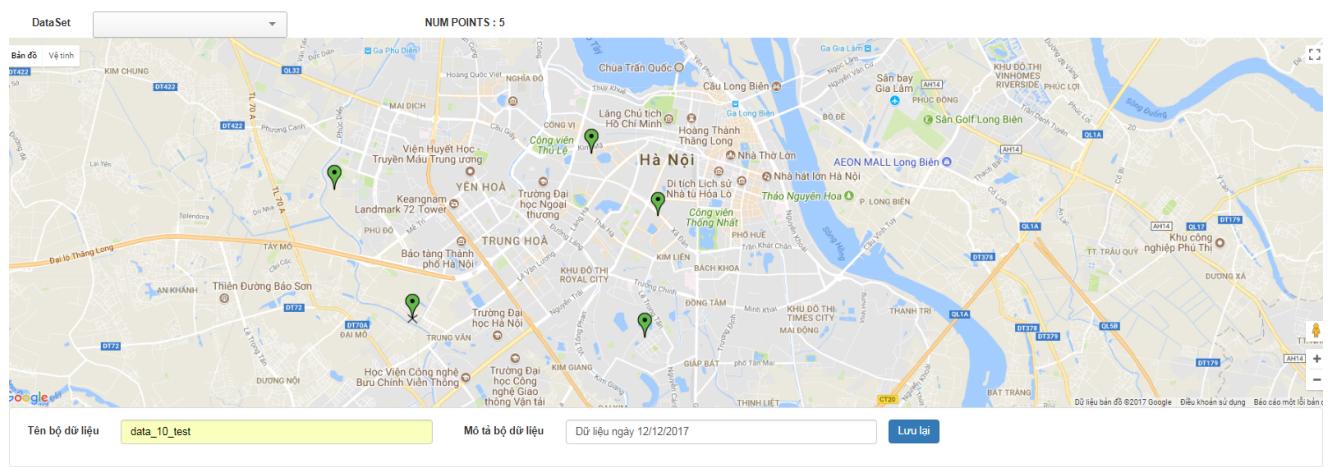
Hình 5.2: Thiết kế cơ sở dữ liệu của ứng dụng

Create data sample



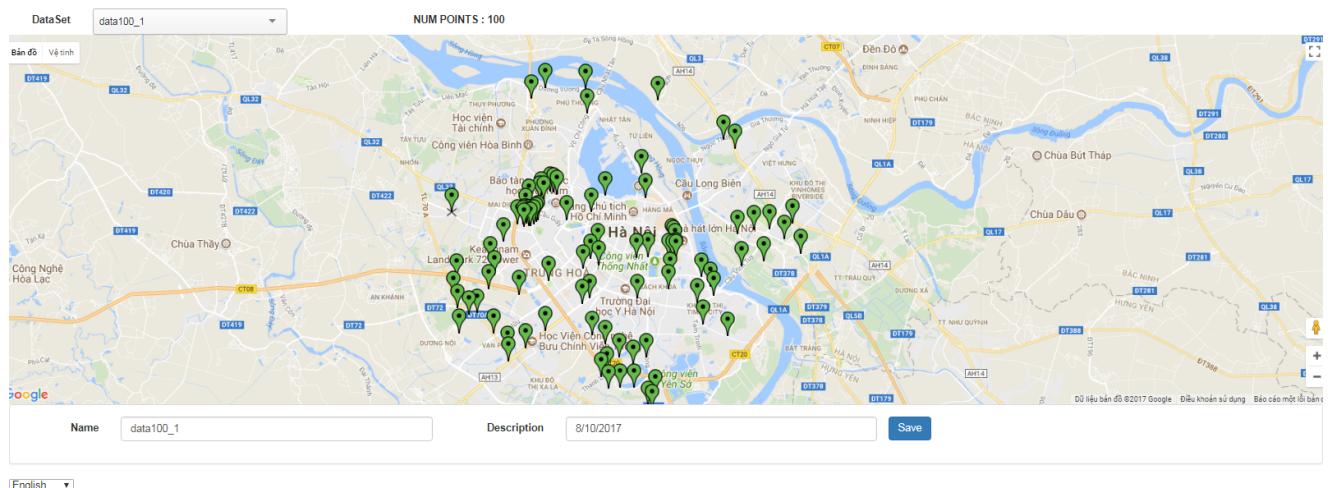
Hình 5.3: Màn hình thay đổi ngôn ngữ sang tiếng Anh

Tạo bộ dữ liệu mẫu



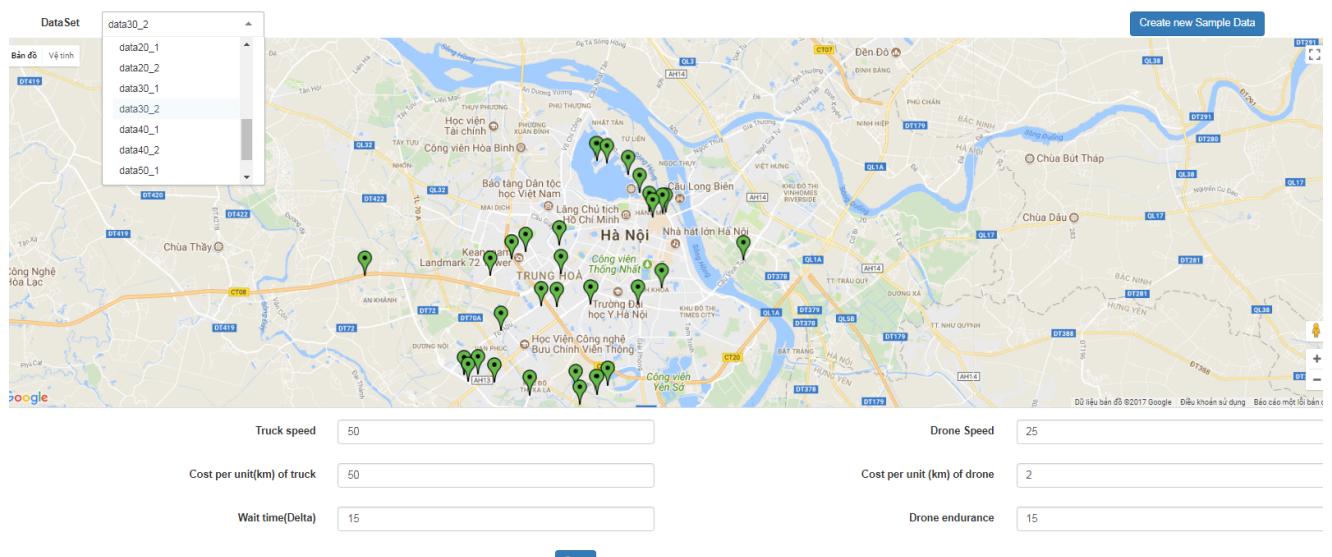
Hình 5.4: Màn hình tạo dữ liệu lập lịch

Create data sample



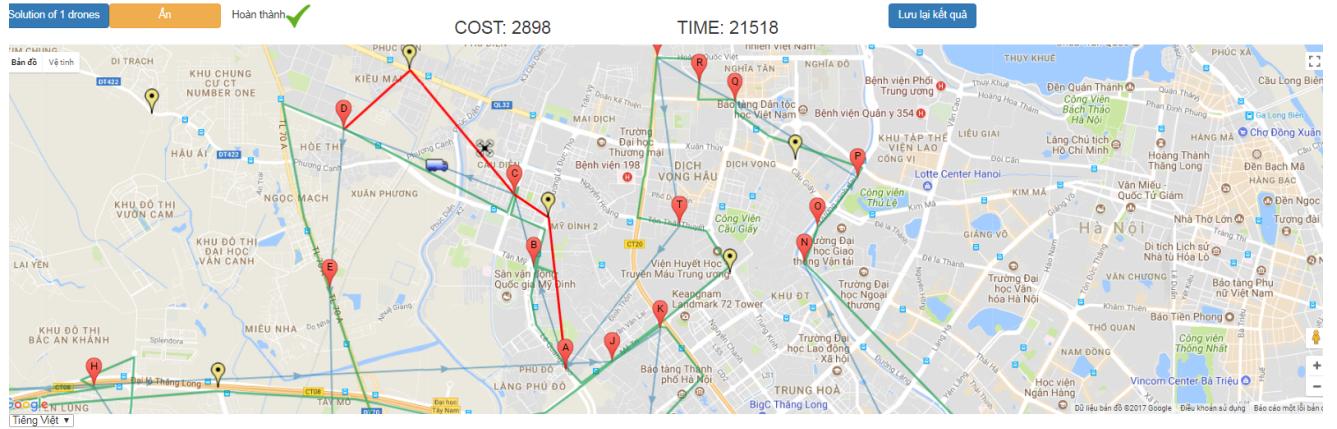
Hình 5.5: Màn hình chỉnh sửa tập dữ liệu

TSPD



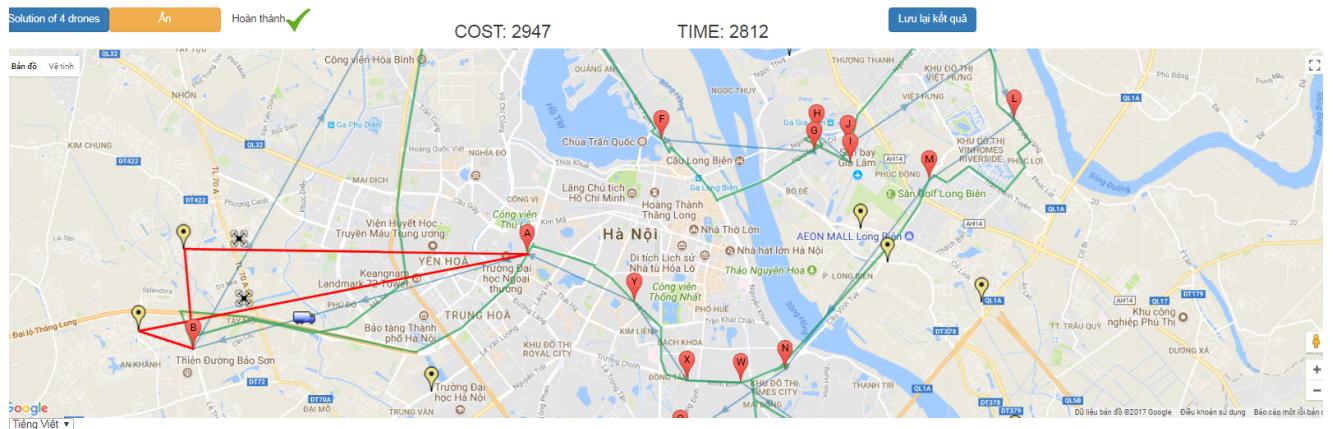
Hình 5.6: Màn hình lập lịch

Kết quả bài toán người du lịch sử dụng nhiều drone



Hình 5.7: Màn hình kết quả với một drone

Kết quả bài toán người du lịch sử dụng nhiều drone



Hình 5.8: Màn hình kết quả với bốn drone

Kết luận và hướng phát triển

Lập kế hoạch vận tải hàng hoá là một lĩnh vực đặc biệt quan trọng trong nền kinh tế mỗi quốc gia. Các mô hình vận tải mới không ngừng được đề xuất mang lại thuận tiện, giảm chi phí vận tải. Trong đồ án này, chúng tôi khảo sát mô hình vận tải hàng hoá kết hợp xe tải và thiết bị bay drone. Xe tải có khả năng vận chuyển các hàng hoá nặng, di chuyển trên hành trình dài, trong khi thiết bị bay drone chỉ có khả năng vận chuyển hàng hoá nhẹ, nhanh, nhưng quãng đường di chuyển ngắn do hạn chế về năng lượng nạp.

Cụ thể, đồ án dựa trên nghiên cứu của Hà Quang Minh và cộng sự, trong đó đề xuất thuật toán heuristics giải bài toán lập lộ trình vận tải kết hợp 1 xe tải và 1 thiết bị bay với mục tiêu là chi phí nhỏ nhất. Chúng tôi đã đề xuất thử nghiệm mô hình 1 xe tải kết hợp với nhiều thiết bị bay (2, 3, 4). Chúng tôi đã cài đặt thuật toán được đề xuất bởi Hà Quang Minh cho mô hình với nhiều drone. Kết quả thử nghiệm cho thấy bằng việc kết hợp với nhiều hơn 1 drone thì chi phí sẽ giảm hơn so với việc sử dụng 1 drone.

Trong đồ án này, chúng tôi còn xây dựng một chương trình ứng dụng cho phép lên kế hoạch và lập lịch sử dụng công nghệ Apache Ofbiz. Module gồm 2 phần là service thuật toán và webApp. Các chức năng có trong ứng dụng như tạo bộ dữ liệu, sửa bộ dữ liệu, lập lịch giao hàng, hiển thị lịch giao hàng, tải về kết quả lập lịch, chuyển đổi ngôn ngữ.

Trong tương lai, chúng tôi sẽ cố gắng phát triển thuật toán trên các mô hình vận chuyển hàng hóa kết hợp xe tải và drone khác như: bài toán vận chuyển hàng hóa kết hợp xe tải và drone nhiều lộ trình, bài toán nhận hàng và đón hàng kết hợp xe tải và drone Ngoài ra chúng tôi cũng thiết kế và cài đặt thêm một vài thuật toán với heuristic mới để đem lại kết quả tốt hơn nữa cho bài toán.

Tài liệu tham khảo

- [1] Balanced Academic Curriculum Problem <http://www.csplib.org/Problems/prob030/>
- [2] Stuart Russell and Peter Norvig *Artificial Intelligence: A Modern Approach* 2nd edition, Prentice Hall, page 137, 2003.
- [3] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein *Introduction to Algorithms* 3rd edition, MIT Press, 2009.
- [4] Francesca Rossi, Peter van Beek, Toby Walsh *Handbook of Constraint Programming* 1st edition, Elsevier Science, Chapter 2 Constraint Satisfaction: An Emerging Paradigm, 2006.
- [5] El-Ghazali Talbi, *Metaheuristics from design to implementation* 1st edition, A Wiley-Interscience Publication John Wiley& Sons, inc , 2009.
- [6] Laurence A. Wolsey, *Integer Programming* 1st edition ,A Wiley-Interscience Publication John Wiley& Sons, inc ,1998.
- [7] Nguyễn Đức Nghĩa, Nguyễn Tô Thành *Toán rời rạc* 3rd edition, Nhà xuất bản Đại học Quốc gia Hà Nội, page 107-108, 2006.
- [8] Nguyễn Đức Nghĩa, *Tối ưu hóa (quy hoạch tuyến tính và rời rạc)* 2nd edition, Nhà xuất bản Giáo dục, 1999.
- [9] Ha Quang Minh and Deville Yves and Pham Quang Dung and Ha Minh Hoang, *On the min cost traveling salesman problem with drone*, arXiv preprint arXiv:1509.08764, 2015
- [10] S. Bunker, Amazon and drones – here is why it will work (dec 2013). URL <http://www.forbes.com/sites/stevebunker/2013/12/19/amazon-drones-here-is-why-it-will-work/>
- [11] C. C. Murray, A. G. Chu, *The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery*, Transportation Research Part C: Emerging Technologies 54 (2015) 86–109.
- [12] N. Agatz, P. Bouman, M. Schmidt, *Optimization approaches for the traveling salesman problem with drone*.
- [13] A. Ponza, *Optimization of drone-assisted parcel delivery*.
- [14] X. Wang, S. Poikonen, B. Golden, *The vehicle routing problem with drones: several worst-case results*, Optimization Letters (2016) 1–19.

- [15] L. K. Nozick, M. A. Turnquist, *Inventory, transportation, service quality and the location of distribution centers*, European Journal of Operational Research 129 (2) (2001) 362–371.
- [16]] K. R. Dawn Russell, John J. Coyle, E. A. Thomchick, *The real impact of high transportation costs (jan 2014)*. URL <http://www.supplychainquarterly.com/topics/Logistics/91020140311-the-real-impact-of-high-transportation-costs/>
- [17] A. Robinson, *Logistics and transportation expenses: Understanding their role in the cost of doing business (feb 2014)*.
- [18] N. Mathew, S. L. Smith, S. L. Waslander, *Planning paths for package delivery in heterogeneous multirobot teams*, Automation Science and Engineering, IEEE Transactions on 12 (4) (2015) 1298–1308.
- [19] G. Gutin, A. P. Punnen, *The traveling salesman problem and its variations*, Vol. 12, Springer Science & Business Media, 2006. URL <http://cerasis.com/2014/02/14/transportation-expenses/>
- [20] Pham Quang Dung, Le Kim Thu, Nguyen Thanh Hoang, Pham Van Dinh, Bui Quoc Trung, *A Constraint-Based Local Search for offline and online general vehicle routing*, International Journal on Artificial Intelligence Tools, Vol. 26, No. 2, 2017 URL <https://doi.org/10.1142/S021821301750004X>.
- [21] Rupert Howell, Jonathon Wong *Apache OFBiz Development: The Beginner's Tutorial*, Paperback , page 53, 2008. truyền.
- [22] Ruth Hoffman *Apache OFBiz Cookbook*, Paperback , page 28, 2010.
- [23] Apache Ofbiz <https://ofbiz.apache.org/>