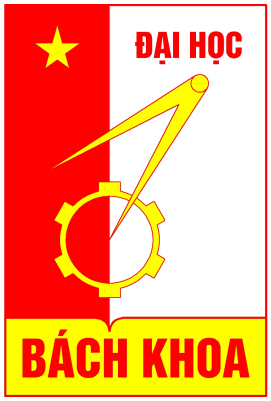
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

_____*******_____



ĐỒ ÁN TỐT NGHIỆP

NGÀNH KHOA HỌC MÁY TÍNH

Phát triển 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

Sinh viên thực hiện:

Nguyễn Tuấn Đạt 20130856 CNTT2.02-K58

Giảng viên:

TS. Phạm Quang Dũng

Hà Nội 5–2017

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 trường 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 long 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, các anh chị ở lab MSO đã nhiệt tình 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, đông viên và giúp đỡ tôi trong suốt quá trình học tập và thực hiện đồ án tốt nghiệp.

Mục lục

Bảng chữ viết tắt

CSP Constraint Satisfaction Problem- Bài toán thỏa mãn ràng buộc

ERP Enterprise Resource Planning

FTL FreeMarker Template Language

GRASP A Greedy Randomized Adaptive Search Procedure

JSP Java Server Pages

SOAP simple object access protocol là một dạng web service

TSPD Traveling salesman problem with Drone

Danh mục

DD Drone delivery - Một chuyến giao hàng bởi drone

TD Truck delivery - Một hành trình giao hàng bởi xe tải

Danh sách hình vẽ

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 quảng đường bay có thể đạt đến hàng chục kilometers. Công nghệ ngày càng phát triển, 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 (ERP)(Enterprise Resource Planning) của các công ti 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 vài 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 viện 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 đồ án bao gồm:

- **Chương 1:**
- **Chương 2:**
- **Chương 3:**
- **Chương 4:**
- **Chương 5:**
- **Chương 6:**

Phần 1

Cơ sở lý thuyết

Trong phần 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[6].

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 dưới dạng tổng quát có thể được phát biểu như sau:

Tìm cực tiểu hay cực đại của phiến hàm

$$f(x) \to min(max), x \in D$$

trong đó D là một tập hữu hạn các phần tử. Hàm $f(x)$ được gọi là hàm mục tiêu của bài toán. Mỗi phần tử $x \in D$ được gọi là một phương án của bài toán, tập D gọi là tập các phương án của bài toán. Phương án $x^* \in D$ đem lại giá trị nhỏ nhất (lớn nhất) cho hàm được gọi là phương án tối ưu, khi đó $f^* = f(x^*)$ được gọi là giá trị tôi ưu của bài toán[6].

Bài toán thỏa mãn ràng buộc

Bài toán thỏa mãn ràng buộc(Constraint Satisfaction Problem) được định nghĩa như sau:
Cho X là một tập các **biến** $x_1, x_2, x_3, \ldots, x_n$ và C là một tập các **ràng buộc** $c_1, c_2, c_3, \ldots, c_m$. Mỗi biến x_i có một tập hữu hạn giá trị xác định D_i. Mỗi ràng buộc c_i thể hiện một mỗi quan hệ phải tuân theo của một tập các biến trong X. 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 Constraint Satisfaction Problem- Bài toán thỏa mãn ràng buộc (CSP) còn yêu cầu tối ưu một hàm

mục tiêu. [3] [2]

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 cân bằng chương trình học.

N con hậu (N-QUEEN)

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

Mô hình toán học

Biến Gọi $X = \{x_i | i \in [1, n]\}$ có giá trị là cột của con hậu hàng thứ i.

Miền giá trị $x_i \in [1, n]$
Vd: $x_1 = 2$ sẽ tương ứng với con hậu hàng thứ nhất đang đứng ở cột thứ 2.

Các ràng buộc

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

$$x_i \neq x_j, \forall i \neq j, i, j \in [1, n]$$

- Các con hậu không cùng trên một hàng chéo.

$$x_i - x_j \neq i - j, \forall i \neq j, i, j \in [1, n]$$

$$x_j - x_i \neq i - j, \forall i \neq j, i, j \in [1, n]$$

Cân bằng lịch học (Balanced Academic Curriculum Problem)

Bài toán yêu cầu thiết kế một chương trình học, được định nghĩa bằng việc sắp xếp các lớp vào các kì học sao cho thỏa mãn các quy định và cân bằng cho mỗi kì. Một chương trình học phải thỏa mãn các quy định sau:

- Một chương trình học định nghĩa bởi một tập các lớp 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 lớp 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 lớp cần có những lớp khác học trước.
- Giới hạn tín chỉ: Trong một kì sinh viên chỉ 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 lớp 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.[1]

Mô hình toán học

Đầu vào

- $nPeriods$ là số lượng kì cần xếp lịch.
- $nCourses$ là số lượng lớp cần xếp lịch.
- $prereq$ là một danh sách các phần tử có dạng $< i, j >$ thể hiện lớp i cần phải được học trước lớp j.
- $credit(j)$ là số lượng tín chỉ của lớp j.

Biến Gọi

$$X = \{X_{ij} | i \epsilon [1, nPeriods], j \epsilon [1, nCourses]\}$$

Miền giá trị

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

Các ràng buộc Ta định nghĩa :

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

- Một lớp chỉ trong một kì

$$\sum_{i \in [1, nPeriods]} X_{i,j} = 1, \forall j \in [1, nCourses]$$

- Điều kiện học phần học trước

$$\forall < i, j > \in prereq, \ \forall k \in [1, nPeriods] : k > 1$$

$$X_{k,j} \leq \sum_{r=1}^{k-1} X_{r,i}$$

- Điều kiện giới hạn số tín chỉ trong một kì

$$load[i] \leq b, \forall i \in [1, nPeriods]$$

- Điều kiện giới hạn số lớp học trong một kì

$$\forall i \in [1, nPeriods]||c \leq \sum_{j \in [1, nCourses]} X_{i,j} \leq d$$

- Điều kiện cân bằng

$$miniumC = (max_{i \in [1, nPeriods]} load[j] - min_{i \in [1, nPeriods]} load[j]$$

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 là những bài toán thực tiễn được phát biểu dưới dạng bài toán tối ưu hóa tổ hợp, qua đó thể hiện được bài toán tối ưu hóa tổ hợp có tính ứng dụng cao, đây cũng là động lực phát trển của lĩnh vực tối ưu hóa tổ hợp[7].

Lập thời khóa biểu (SIE Course Timetabling)

Đầu vào

- Cho một tập ngày $D_1, \ldots, D_q$ mỗi ngày có p tiết, vd: Thứ 2 có 6 tiết.
- Cho một tập các giảng viên $P = \{P_1, \ldots, P_m\}$, mỗi giảng viên có một tập B bao gồm các cặp $< th, tit >$ mà giảng viên đó bận.
- Một tập lớp sinh viên $CL = \{CL_1, \ldots, CL_k\}$
- Một tập các lớp học môn học: $C = \{c_1, \ldots, c_n\}$ mỗi c_i bao gồm 3 thông tin:
  1. $d(c_i)$: sô lượng tiết của lớp c_i
  2. $p(c_i)$: giảng viên được yêu cầu dạy lớp c_i
  3. $cl(c_i)$: tập các lớp sinh viên học trong c_i

Đầu ra Xây dựng thời khóa biểu cho tập lớp C trong tập ngày D.

Các ràng buộc

- Hai lớp môn học bất kì có chung giảng viên thì không được trùng tiết trong một ngày.
- Một buổi học không được trùng vào thời gian bận của giáo sư[8].

Phân công y tá

Đầu vào Bài toán yêu cầu xếp lịch ca trực cho n y tá trong m ngày mỗi ngày. Mỗi ngày có k kíp trực khách 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 kí một số ngày họ được nghỉ.

Ràng buộc mềm

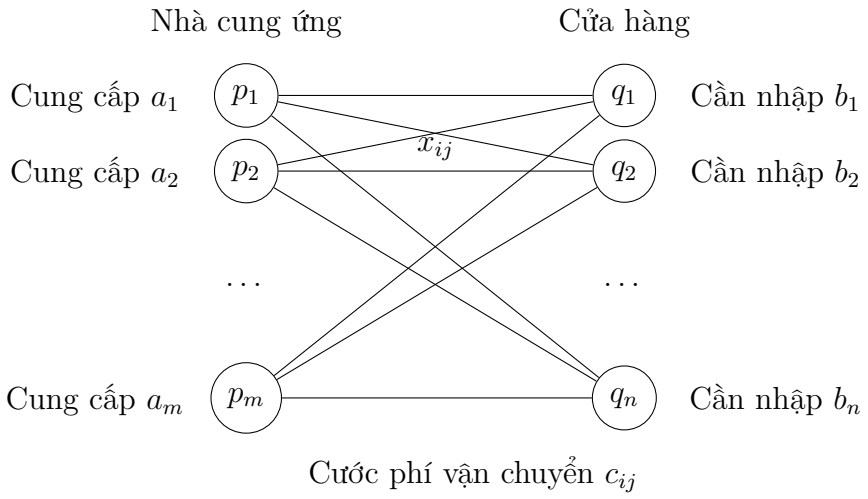
- Các y tá được phép yêu cầu nghỉ hoặc làm một số ca nào đó.
- Bệnh viện yêu cầu số lượng y tá trong mỗi ca trực.

Lập lộ trình vận tải

Lớp bài toán lập lộ trình vận tải là một trọng những lớp bài toán tối ưu quan trọng trong ngành công nghiệp giao vận. 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 phi vận chuyển một đơn vị hàng từ nhà cung cấp i đến cửa hàng j.[7]



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

1.2.1 Thuật toán giải đúng

Trong phần này chũng 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

Quy hoạch động là một kĩ 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. [9]

Việc thiết kế thuật toán giải bài toán quy hoạch động giải bài toán tối ưu hóa tổ hợp phải trải qua các giai đoạn:(1) Phân rã bài toán, (2) Ghi nhận lời giải, (3) Tổng hợp lời giải. [9]

Với giai đoạn phân rã, chúng ta cần chia bài toán thành những bài toán con nhỏ hơn có cùng dạng với bài toán ban đầu sao cho bài toán con kích thước nhỏ nhất có thể giải trực tiếp. [9]

Với giai đoạn ghi nhận lời giải, chúng ta lưu trữ lời giải của các bài toán con vào một bảng. Lời giải của các bài toán con thường được sử dụng lại rất nhiều lần, và để tăng hiệu quả của thuật toán việc lưu lại lời giải của bài toán con là điều tất nhiên.[9]

Với giai đoạn tổng hợp lời giải, lần lượt từ lời giải của bài toán con kích thước nhỏ hơn xây dựng lời giải cùa bài toán kích thước lớn hơn, cho đến khi thu được lời giải của bài toán xuất phát (là bài toán con có kích thước lớn nhất). [9]

Quy hoạch ràng buộc

Quy hoạch ràng buộc cũng là một phương pháp giải đúng bài toán tối ưu hóa tổ hợp. Chúng ta có thể xây dựng chiến lượng tìm kiếm trong quy hoạch ràng buộc, đôi khi ta có thể thêm các nhánh cận để thuật toán hiệu quả hơn[10]. Cơ bản quy hoạch ràng buộc có thể mô tả hai bước như sau:

1. Lan truyền ràng buộc, bao gồm các bước: (1) Giảm kích thước miền giá trị của cá biến, (2) Sử dụng các ràng buộc để loại bổ các phương án sai,(3) Giải bài toán khi đã các ràng buộc đã được lới lỏng.
2. Tìm kiếm, có thể là đệ quy quay lui.[8]

Quy hoạch nguyên tuyến tính

Quy hoạch nguyên tuyến tính cũng là phương pháp mạnh để giải quyết bài toán tối ưu hóa tổ hợp.

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, vi vậy với những bài toán có kích thước đầu vào lớn thì việc sử dụng thuật toán giải đúng dường như là không thể. Trong khi đó các thuật toán giải gần đúng mặc dù không cho lời giải không chính xác tuyệt đối nhưng thời gian thực thi lại hoàn toàn có thể chập 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 làm, 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

Thuật toán tham lam cũng là một hướng tiếp cận cho lời giải gần đúng. Với một trạng thái hiện tại ta đưa ngay ra quyết định từ những thông tin của trạng thái hiện tại mà không phải xem xét cá bước đi trong quá khứ. Chính vì vậy thời gian thực hiện thuật toán nhanh nhưng thường cho kết quả không chính xác tuyệt đối. Sơ đồ hoạt động của thuật toán gồm các bước sau:

1. Lặp:
   (a) Chọn phần ứng cử viên có triển vọng nhất vào lời giải hiện có.
   (b) Kiểm tra tính chấp nhận được của lời giải bộ phận.
2. Kiểm tra tính chấp nhận được của lời giải toàn cục. [9]

Thuật toán tìm kiếm cục bộ

Ý tưởng của thuật toán tìm kiếm cục bộ là cải thiện lời giải qua từng bước lặp bằng việc di chuyển qua một láng giềng của lời giải. Láng giếng của một lời giải là lời giải đó sau khi một hoặc một vài biến đã được cập nhật giá trị mới. Vì vậy, một bước di chuyển tương đương với việc cập nhật một hoặc một vài giá trị mới cho các biến. Tất nhiên ta phải có một hàm dùng để đánh giá chất lượng lời giải (vd: hàm số lượng ràng buộc đang bị vi phạm ở thời điểm hiện tại).

Thuật toán di truyền

Ý tưởng của thuật toán di truyền là 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 cồm 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.[11]

1.3 Công nghệ ofbiz

Apache Ofbiz 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 kì nước nào. Ofbiz là framework mã ngồn 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 truc 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 điễn** 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).

Phần 2

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

Trong phần 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 giới thiệu 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 giả. Các tài liệu tham khảo sử dụng trong phần này bao gồm.

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

2.1.1 Giới thiệu 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(Traveling salesman problem with Drone (TSPD)) trong đó drone tham gia giao hàng cũng với xe tải. Dưới đây là bốn lợi ích nổi bật của drone [4] :

- 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 mới được ra đời có tên gọi là "last mile delivery with drone" [5]. 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 vị nó hoạt động được. Trong lúc đó xe tải vẫn tiếp tục thực hiện tiếp lộ trình của mình. Drone phục vụ khách hàng xong sẽ quay lại xe tải ở một điểm mới khác với điểm nó bay lên[4].

2.1.2 Đặt vấn đề

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. Môi 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 được giao bởi drone là Drone delivery - Một chuyến giao hàng bởi drone (DD)(drone delivery). Một 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:

- Điển 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 điểm nào cũng có thể trở thành điểm giao hàng của drone vì có những điểm ngoài phạm vị phục vụ được của drone.
- Điểm hạ cánh (rendezvous node) là điểm mà drone quay trở lại xe tải. Tất nhiên hai phương tiện có thể phải chờ nhau tại điểm này.

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 đượng là chi phí cho vận chuyển. Trong thực tế, chi phí vẫn 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.

Yêu cầu của bài toán là tối ưu tổng chi phí vận chuyển của cả hai thiết bị.

2.1.3 Mô hình bài toán min-cost TSPD

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

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 $s =< s_1, s_2, .., s_t >, s_i \in V, i = 1, .., t$ 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.
- $sub(i, j, s)i, j \in s$ chuỗi lộ trình con của s.
- $A(s) = (i, next_s(i))|i \in V(s) \setminus last(s)$ là tập cảnh thuộc đường đi của s.

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

$$P = \{\langle i, j, k\rangle : i, j, k \in V, i \neq j, j \neq k, k \neq i, d'_{ij} + d'_{jk} \leq \epsilon\}$$

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

- Một hành trình cho xe tải gọi là Truck delivery - Một hành trình giao hàng bởi xe tải (TD), là một chuỗi lộ trình qua các điểm $< e_0, e_1, .., e_k >$ với $e_0 = 0$ và $e_k = n+1$ 0 và $n + 1$ là điểm kho, $e_i \neq e_j, i \neq j$
- Một tập $DDs = \{DD|DD \subseteq P\}$

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.

$$\forall e \in N : e \in TD \text{ or } \exists\langle i, e, k\rangle \in DDs. \tag{2.1}$$

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

$$\forall\langle i, j, k\rangle, \langle i', j', k'\rangle \in DD : j \neq j' \tag{2.2}$$

(C) Mỗi một DD phải thỏa mãn với TD

$$\forall\langle i, j, k\rangle \in DDs : j \notin TD, i \in TD, k \in TD, pos(i, TD) < pos(k, TD) \tag{2.3}$$

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

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

2.1.5 Hàm mục tiêu

Một vài khái niệm

- $cost(i,j,k) = C_2(d'_{ij} + d'_{jk}) < i,j,k > \in P$: chi phí của một DD.
- $cost(TD) = \sum(i,j) \in A(TD) : C_1.d_{i,j}$: chi phí của một TD.
- $cost(DDs) = \sum < i,j,k > \in DD : cost(i,j,k)$: chi phí của tất cả các DD trong lời giải.
- $cost(TD, DD) = cost(TD) + cost(DD)$: chi phí của lời giải.
- $cost(sub(i,k,s))$: 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 quy hoạch nguyên bộ phận

Bài toán sẽ được trình bày cụ thể trong phần này bởi mô hình quy hoạch nguyên bộ phận.

Biến

Gọi $x_{ij} \in 0,1$ với $i \in V_L = [0,1,..,n], j \in V_R = [1,2,..,n+1$ 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 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 quy hoạch nguyên

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

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

$$\sum_{j \in V_R} x_{0j} = 1 \tag{2.7}$$

$$\sum_{i \in V_L} x_{i,n+1} = 1 \tag{2.8}$$

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

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

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

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

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

$$\sum_{\substack{j \in N \\ j \neq i}} \sum_{\substack{k \in V_R \\ \langle i,j,k \rangle \in P}} y_{ijk} \leq 1 \forall i \in V_L \tag{2.14}$$

$$\sum_{\substack{i \in V_L \\ i \neq k}} \sum_{\substack{j \in N \\ \langle i,j,k \rangle \in P}} y_{ijk} \leq 1 \quad \forall k \in V_R \tag{2.15}$$

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

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

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

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

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

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 luôn chỉ 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 bàn toán min-cost TSPD. Thuật toán được trình bày cụ thể trong 1.

Algorithm 1 A Greedy Randomized Adaptive Search Procedure (GRASP)

Result: bestSolution

```
1:  bestSolution = null;
2:  bestObjectValue = T;
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(tspdSolution) < bestObjectValue then
10:         bestSolution = tspdSolution
11:         bestObjectiveValue = f(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) solution extraction.

Ý tưởng của *SplitAlgorithms* là xây dưng 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) s 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.

Algorithm 2 SplitAlgorithm,Step 1: Xây dựng auxiliary graph và tìm đường đi ngắn nhất

Data: TSP tour s

Result: 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.

$arcs = \oslash$
$T = \oslash$;
for each $i \quad in \quad s, \backslash$last(s) **do**
    $k = pos(i, s) + 1$
    $arcs = arcs \cup (i, k, cost(i, k, s))$
end for
for each $i \; in \; s \setminus \{last(s), s[pos(last(s), s) - 1]\}$ **do**
    for each $k \; in \; s : pos(k, s) \geq pos(i, s) + 2$ **do**
        $minValue = T$;
        $minIndex = T$; Chỗ này hình như sai
        for each $j \; in \; s : pos(i, s) < pos(j, s) < pos(k, s)$ **do**
            if $\langle i, j, k \rangle \in P \; and | d_{i \to k} - (d'_{ij} + d'_{j,k}) | < \triangle$ **then**
                $cost =$
                $cost(sub(i, k, s)) + C_1(d_{prev(j,s)next(j,s)} - d_{prev(j,s),j} - d_{j,next(j,s)} + cost(i, j, k)$
                if cost<minValue **then**
                    $minValue = cost$
                    $minIndex = pos(j, s)$
                end if
            end if
        end for
        $arcs = arcs \cup \{(i, k, minValue)\}$
        if $minIndex \neq T$ **then**
            $T = T \cup (i, s[minIndex], k, minIndex)$
        end if
    end for
end for
$P[0]$
for each $k \; in \; s \backslash \{0\}$ **do**
    for each $(i, k, cost) \in arcs$ **do**
        if (**then**$V[k] > V[i] + cost$)
            $V[k] > V[i] + cost$
            $P[k] = i$
        end if
    end for
end for
return *bestSolution*

Xây dựng auxiliary graph và tìm lộ 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 lộ 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 $\langle i, j, k\rangle \in P$

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

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 depot 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.

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

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

Chúng tôi lưu một mảng lưu vết P, $j = 1, \ldots, n + 1 P(j)$ 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.

Độ phức tạp: ***

Solution extraction

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 $S_a = 0, 1, \ldots, n + 1$ trình bày đường đinh 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 subroute 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 cập điểm $i, i + 1$ trong S_a xác

định số điểm trong tourTSP nằm giữa hai điểm này, nếu hai có ít nhất một điểm năm giữa hai điểm trên thì ta lấy DD với điểm đầu cuối tương ứng trong T bỏ vào DDs (S_d). Thuật toán dược trình bày trong 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 SplitAlgorithm,Step 2: Xây dựng lời giải từ step 1

Data: 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
Result:tspdSolution

1: $i = T$
2: $S_a = j$
3: **while** $i \neq 0$ **do**
4: $\quad i = P[j]$
5: $\quad S_a = S_a \cup i$
6: $\quad j = i$
7: **end while**
8: $S_a = S_a.reverse()$;
9: $S_t = \oslash$
10: **for** $i = 0; i < S_a.size - 1; i++$ **do**
11: $\quad$ **if** betweeb$S_a[i] and S_a[i+1]$ *in* $tspTour, (ctnhtmt\emptyset im)$ **then**
12: $\quad\quad n_drone =$ drone node in T
13: $\quad\quad S_d = S_d \cup \langle S_a[i], n_drone, S_a[i+1]$
14: $\quad$ **end if**
15: **end for**
16: $currentPosition = 0$
17: **while** $currentPosition \neq n+1$ **do**
18: $\quad$ **if** $curentPosition$là một điểm bay của một DD t trong S_d **then**
19: $\quad\quad S_t = S_t +$ tất cả các điểm thuộc $tspTour$ nằm từ $currentPosition$ đến $t_{rendezvous\_node}$ ngoại trừ t_{drone_node}
20: $\quad\quad currentPosition = t_{rendezvous\_node}$
21: $\quad$ **else**
22: $\quad\quad S_t = S_t \cup currentPosition$
23: $\quad\quad curentPosition = tspTour[indexOf(currentPosition) + 1]$
24: $\quad$ **end if**
25: **end while**
26:
27: $tspdSolution = (S_t, S_d)$
28: **return** $tspdSolution$

2.2.2 Các toán tử tìm kiếm cục bộ

Trong phần này chúng tôi trình bày các toán tử được làm mới để phù hợp 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* và

$two_exchange$. Ngoài ra với dựa trên những đặc thù riêng biệt của bài toán min-cost TSPD chúng tôi trình bày thêm hai toán tử mới đó là "*dronerelocate*" dùng để tạo ra một DD và "*droneremoval*" dùng để xóa bỏ một DD. Để bắt đầu chúng ta định nghĩa một vài từ khóa sau:

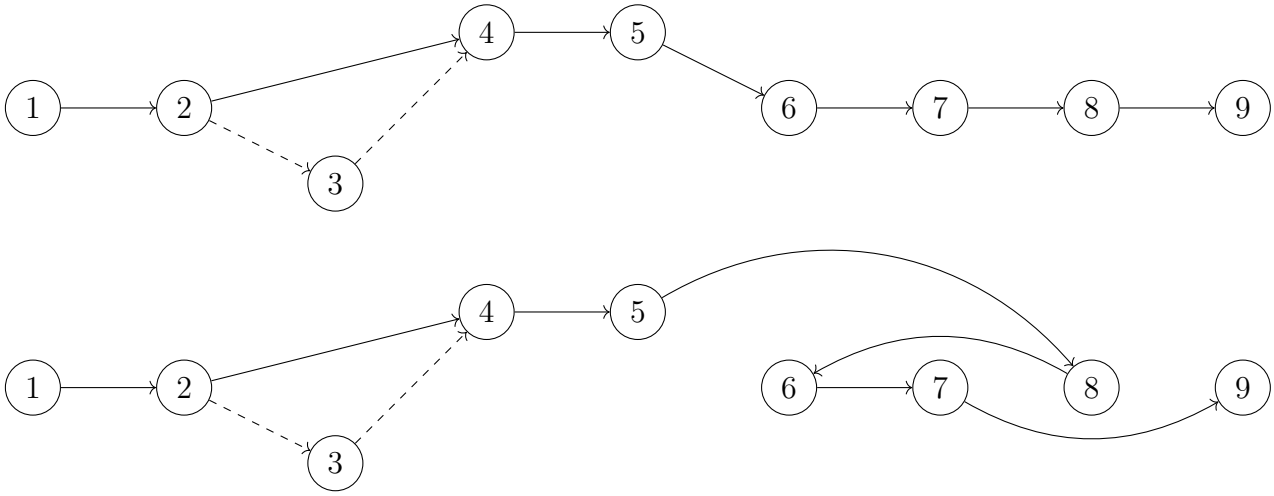
- $N_T(TD, DDs) = \{e : e \in TD, \langle e, ., . \rangle \notin DDs, \langle ., ., e \rangle \notin DDs\}$ là tập các điểm chỉ thuộc TD, không nằm trong bất kì DD nào.
- $N_D(TD, DDs) = \{e : \langle ., e, . \rangle \in DD\}$ là tập các điểm giao hàng drone trong lời giải TSPD.

Tiếp theo chúng tôi định nghĩa mỗi toán tử:

1. *RelocationTruck* : 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

$$relocate_T((TD, DDs), a, b), a \in N_T((TD, DDs)), b \in TD, b \neq a, b \neq 0$$

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

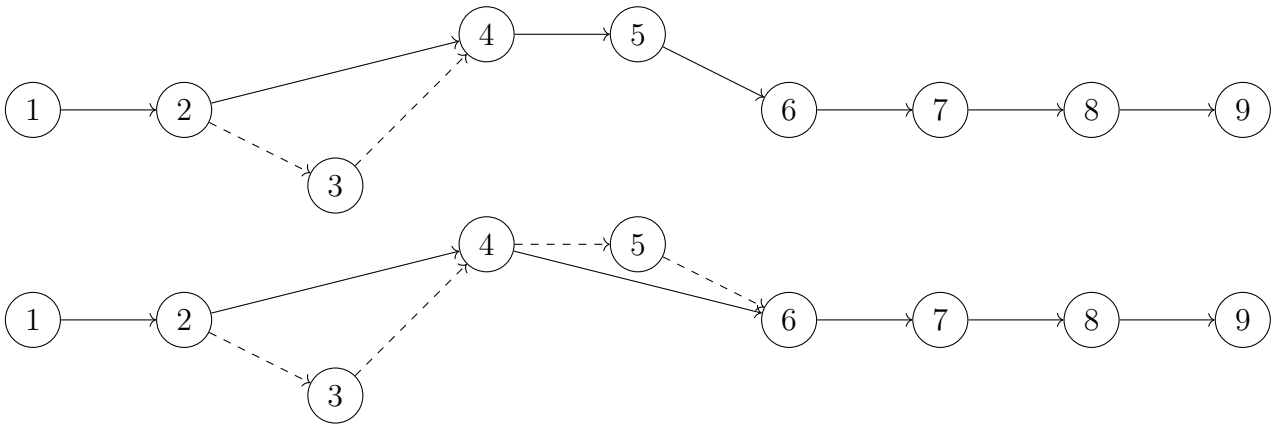


Hình 2.1: Truck relocation

2. *Dronerelocation*: 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. Chi tiết được định nghĩa như sau: (1) chúng ta cân nhắc cả các điểm N_T và N_D, (2) mỗi ***** . Cụ thể chúng tôi định nghĩa :

$$relocate_D((TD, DDs), a, i, k)$$
$$a \in N_T((TD, DD)) \cup N_D((TD, DDs)), i, k, \in TD \backslash \{a\},$$
$$i \neq k, pos(i, TD) < pos(k, TD), \langle i, a, k \rangle \in P$$

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 một nút giao hàng bởi drone toán tử thực hiện thay đổi điểm bay và điểm đón của DD đó thành $\langle i, a, k \rangle$.

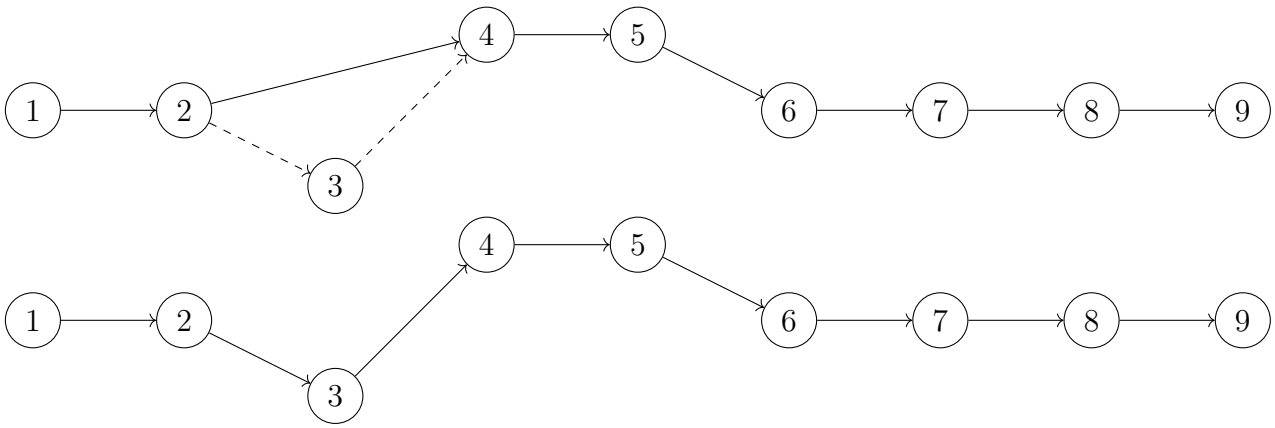


Hình 2.2: Drone relocation

3. *DroneRemoval* : 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 nghía toán tử :

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

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.

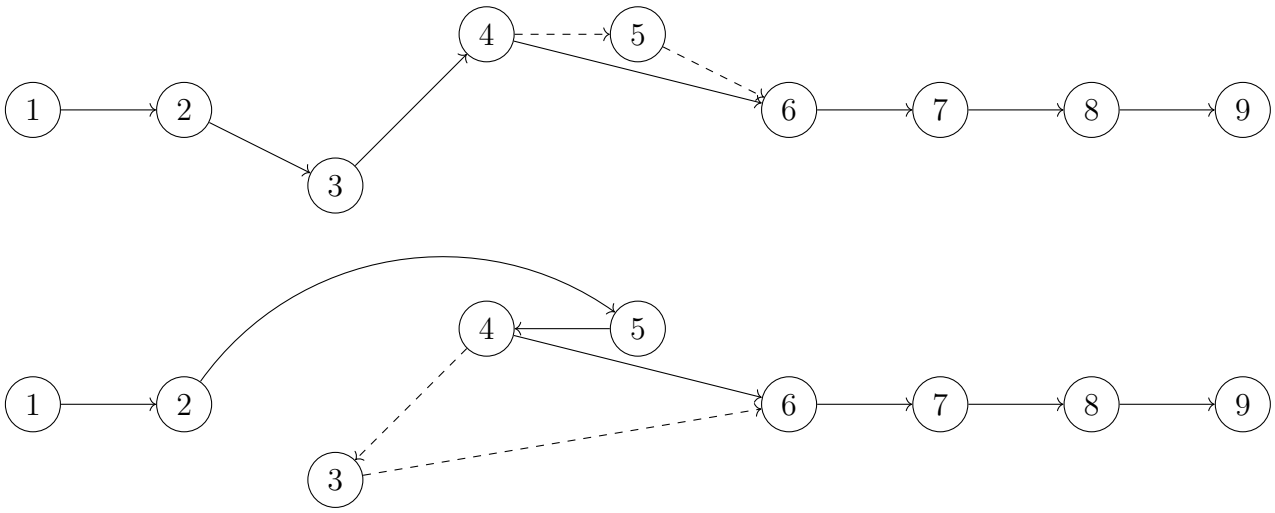


Hình 2.3: Drone removal

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 :

$$two_exchange((TD, DDs), a, b), a, b), a, b \in V \backslash \{0, n+1\}, a \neq b$$

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



Hình 2.4: Two exchange

Tất nhiên 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 băt đầu bằng một hành trình người du lịch sau đó liên tục thay đổi vị trí khách hàng trong lô 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 $1 -> n$.
- một TD ($truckRoute$).
- $truckSubRoute$ chứa tất cả các tập con của $truckRoute$.
- j^* 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 lộ trình.
- $maxSavings$ giá trị giảm được sau khi đặt lại vị trí của j^*.
- $isDroneNode$ xác định
- $Stop$ 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

Data: truck-only sequence truckRoute
Result: TSP-D solution sol

1: $Customers = N$;
2: $truckRoute = solveTSP(N)$;
3: $truckSubRoutes = truckRoute$;
4: $sol = (truckRoute,)$;
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 truckSubRoute **do**
15: **if** drone(subroute,sol) **then**
16: $isDroneNode, maxSavings, i^*, j^*, k^*) = relocateAsTruck(j, subroute, savings)$;
17: **else**
18: $isDroneNode, maxSavings, i^*, j^*, k^*) = relocateAsDrone(j, subroute, savings)$;
19: **end if**
20: **end for**
21: **if** maxSavings>0 **then**
22: $(sol, truckRoute, truckSubRoute, Customers) = applyChange(isDroneNode, i^*, j^*, k^*, sol, trucRoute, trucSubRoute, Customers)$
23: $maxSavings = 0$
24: **else**
25: $Stop = true$
26: **end if**
27: **end for**
28: **until** Stop

Trong mỗi vòng lặp thực hiện hai hai bước:(1) Cân nhắc xem có khách hàng nào là khi di chuyển đến vị trí mới trong lộ trình tạo ra kết quả tốt nhất. (2) Cân nhắn 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, $truckRoutevsubTruckRoute$ 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 hiên từ dòng **đến ** 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 tri phí khi đặt j vào TD và DD được trình bày bởi thuật toán 6 7.

Algorithm 5 calcSavings(j)

Data: j : khách hàng trong TD
Result: Kết quả

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

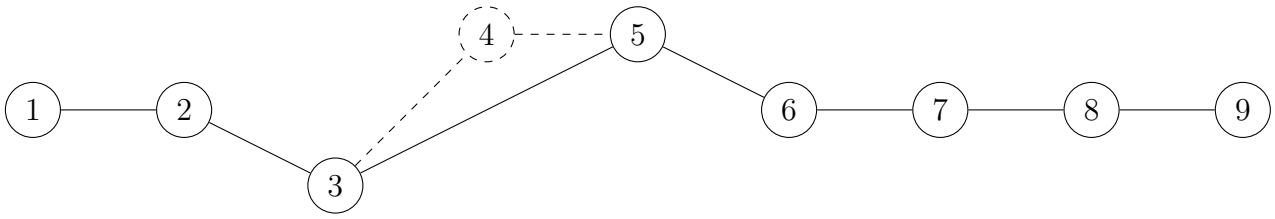
Trong thuật toán 6 mục tiêu của chúng tôi là tìm ra vị trí tốt nhất trong *subroute* s để đặt khách hàng j. Việc này được thực hiện bởi việc đặt thử khách hàng j vào giữa các điểm liên tiếp nhau trong s. Đầu tiên chúng ta sẽ phải kiểm tra xem việc đặt j vào giữa cặp khách hàng i,k có làm *******. Việc đặt j vào phải mất chi phí nhỏ hơn savings. Vì tồn tại một nút DD nằm cùng vơi sub route 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,subreoute,savings)- Tính toán trị phí của việc chuyển khách hàng j đến vị trí khác trong lộ trình

Data: 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

Result: Kết quả cập nhật vào các biến j^* 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 waittime contrait trong các subtruckroute 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 2.5: Mô phỏng thao tác relocateAsTruck

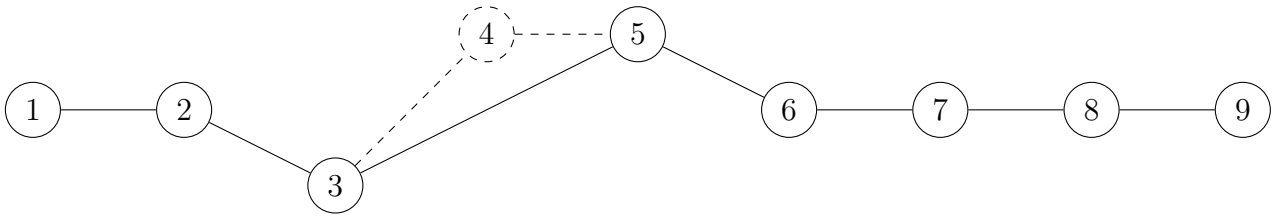
Trong thuật toán 7 chúng tôi tính toán chi phí nếu chuyển j thành khách hàng giao bởi một DD. Điều này được thực hiện bởi biến $\langle i, j, k\rangle$ thành một DD, với i,k là 2 điểm trong s. Tương tự với thuật toán 6 ta phải kiểm tra xem DD vừa tạo có thỏa mãn không. Nếu việc chuyển j vào một DD là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

Algorithm 7 relocateAsDrone(j,subreoute,savings)- Tính toán trị phí của việc chuyển khách hàng j sang giao hàng bằng drone

Data: 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

Result: Kết quả cập nhật vào các biến j^* j^* k^* và $isDroneNode$

```
for i = 0, size(s) − 2 do
    for k = i + 1, size(s) − 1 do
        if ⟨s[i], j, s[k]⟩ ∈ P then
            △ = (d'_{s[i],j} + d'_{j,s[k]})C_2
            if savings − △ > maxSavings then
                isDroneNode = True;
                j* = j; i* = s[i]; k* = s[k]
                maxSavings = saving − △
            end if
        end if
    end for
end for
return (isDroneNode, maxSavings, i*, j*, k*)
```



Hình 2.6: 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 $truckRoute, subTruckRoute$ 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

Data: $isDroneNode, j^*\ j^*\ k^*, sol, truckRoute, truckSubRoute, Customers$
Result: Cập nhật $truckRoute, truckSubRoute, t$

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

Phần 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 phần này chúng tôi trình bày bài toán lập lộ trình vận tải giao hàng kết hợp một xe tải và nhiều thiết bị bay và đề xuất một thuật toán tìm kiếm cục bộ giải quyết bài toán. Cụ thể, phần 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 nhiêu thiết bị bay không người lái, phần 2 sẽ trình bày phát biểu bài toán, biến và các mô hình, phần 3 chúng tôi sẽ trình bày một thuật toán tìm kiếm cục bộ giải quyết bài toán trên.

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 ta chúng tôi cũng đa 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 nhiểu 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.
- Mối 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 các drone coi như được thả đi cùng một thời điểm.
- Tại một thời điêm bất kì không được có vượt quá k thiết bị bay cùng đang hoạt động.
- Chi phí trên một đơn vị độ dài của drone C_2 là của xe tải là C_1 . Drone chỉ mất phí khi đang trong một chuyến giao hàng.Tổng chi phí của chuyền giao hàng bằng tổng chi phí của xe tải và drone công 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

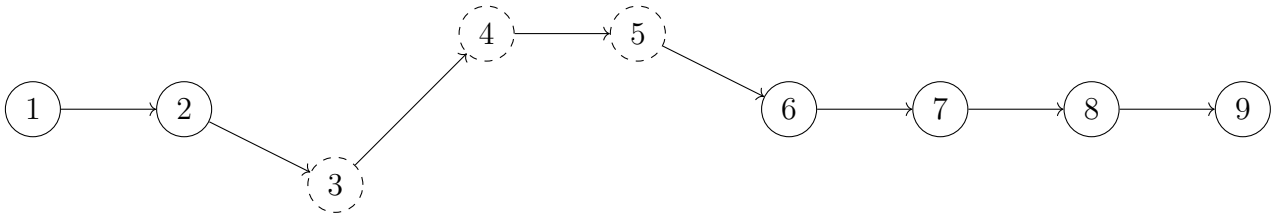
Nhận thầy bài toán min-cost tspkd sẽ dần tới bài toán tsp khi ε tiến dần đến 0, vì vậy bài toán là np-hard. Chính vì vậy chúng tôi lựa chon hướng tiếp cận thuật giải gần đúng cụ thể là thuật toán tìm kiếm cục bộ để giải quyết bài toán. Trong phần 1 chúng tôi sẽ trình bày toán tử move mà chúng tôi sử dụng trong thuật toán, phần 2 là trình bày chi tiết thuật toán.

3.3.1 Toán tử move

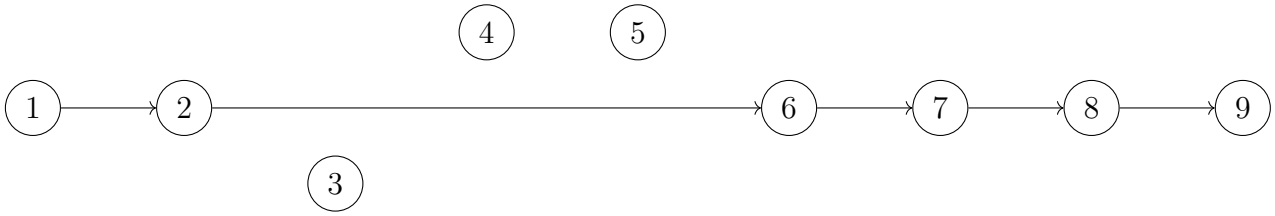
Trong phần này chúng tôi trình bày toán từ move cho bài toán min-cost tspdkd.

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

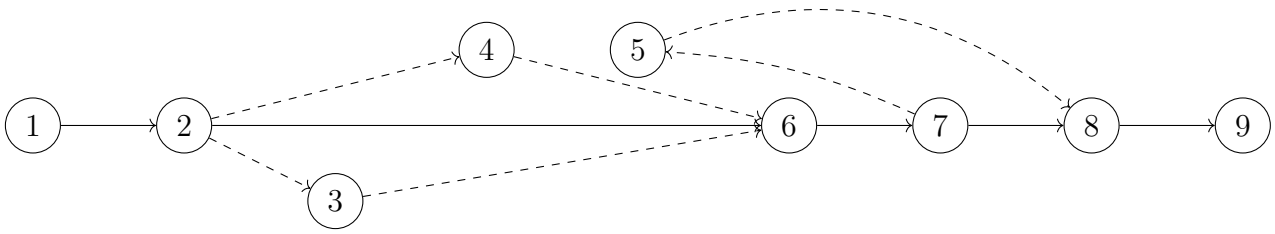
- Bước 1: Chọn t điểm giao hàng liên tiếp trong lộ trình $\langle v_1, v_2, \ldots, v_t \rangle$
- Bước 2: Xóa bỏ t điểm giao hàng vừa chọn ra khỏi lộ trình.
- Bước 3: Với mỗi v_i ta thực hiện các bước:
  1. Tìm hai điểm v_{il} v_{ir} trong lộ trình sao cho lượng chi phí .
  2. Tạo DD mới với v_{il}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



Hình 3.1: Chọn 3 điểm liên tiếp – thao tác move-3-point



Hình 3.2: Xóa bỏ t điểm vừa chọn ra khỏi lộ trình – thao tác move-3-point



Hình 3.3: Chuyển các nút chọn thành DD – thao tác move-3-point

Mô phỏng một thao tác move-3-point 3.1 3.2 3.3 . Bước 1 ta chon 3 điểm liên tiếp nhau vd: điểm 3, 4, 5 (Hình 3.1). Bước 2 ta xóa bỏ 3 điểm vừa chọn ra khỏi lộ trình (Hình 3.2). Bước 3 với điểm giao hàng 3:

1. Thực hiện chon 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 6.

Lặp lại bước 3 với điểm 4,5.

3.3.2 Thuật toán tìm kiếm cục bộ

Trong phần này chúng tôi trình bay 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: Băt đầu từ 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 TSP-LS heuristic

Data: truck-only sequence truckRoute
Result: TSP-D solution tour

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

Trong thuật toán 9 chúng tôi tình 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 kpointmove được trình bày trong thuật toán 10.

Algorithm 10 k-point move

Data: tour, maxRangeMove
Result: 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: $savings = caculateSavingKpoint(j, i)$
6: Lấy toàn bộ i point từ điểm j ra khỏi truckRoute.
7: **if** Nếu có một điểm không hợp lệ **then**
8: continue
9: **end if**
10: **for** $t = j, j + i$ **do**
11: $aPointMaxSaving = 0$
12: $la^* = -1$
13: $dr^* = -1$
14: $re^* = -1$
15: **for each** subroute in truckSubRoute **do**
16: $aPointSavings = 0$
17: $(pointmaxSavings, la, dr, re) = relocateAsDrone(t, subroute, aPointSavings)$;
18: **if** $aPointMaxSaving < pointmaxSavings$ **then**
19: $aPointMaxSaving = pointmaxSavings$
20: $la^* = la$
21: $dr^* = dr$
22: $re^* = re$
23: **end if**
24: **end for**
25: savings=savings+aPointMaxSaving
26: Thêm DD vào danh sách DD cho move.
27: **end for**
28: savings=savings/i;
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: **end for**
36: **return** $moveMaxSavings$ và trạng thái đạt $moveMaxSavings$.

Ý tưởng của thủ tục *kpointmove* 10 như sau. Chúng tôi xem xét tất cả các $move - k - point$ với $k < maxRangeMove$, vd: với $maxRangeMove = 3$ chúng tôi sẽ xem xét $move - 1 - point, move - 2 - point, move - 3 - point$. Đương nhiên để công bằng chúng tôi thực hiện chia chi phí giảm được cho số lượng point trong move. 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

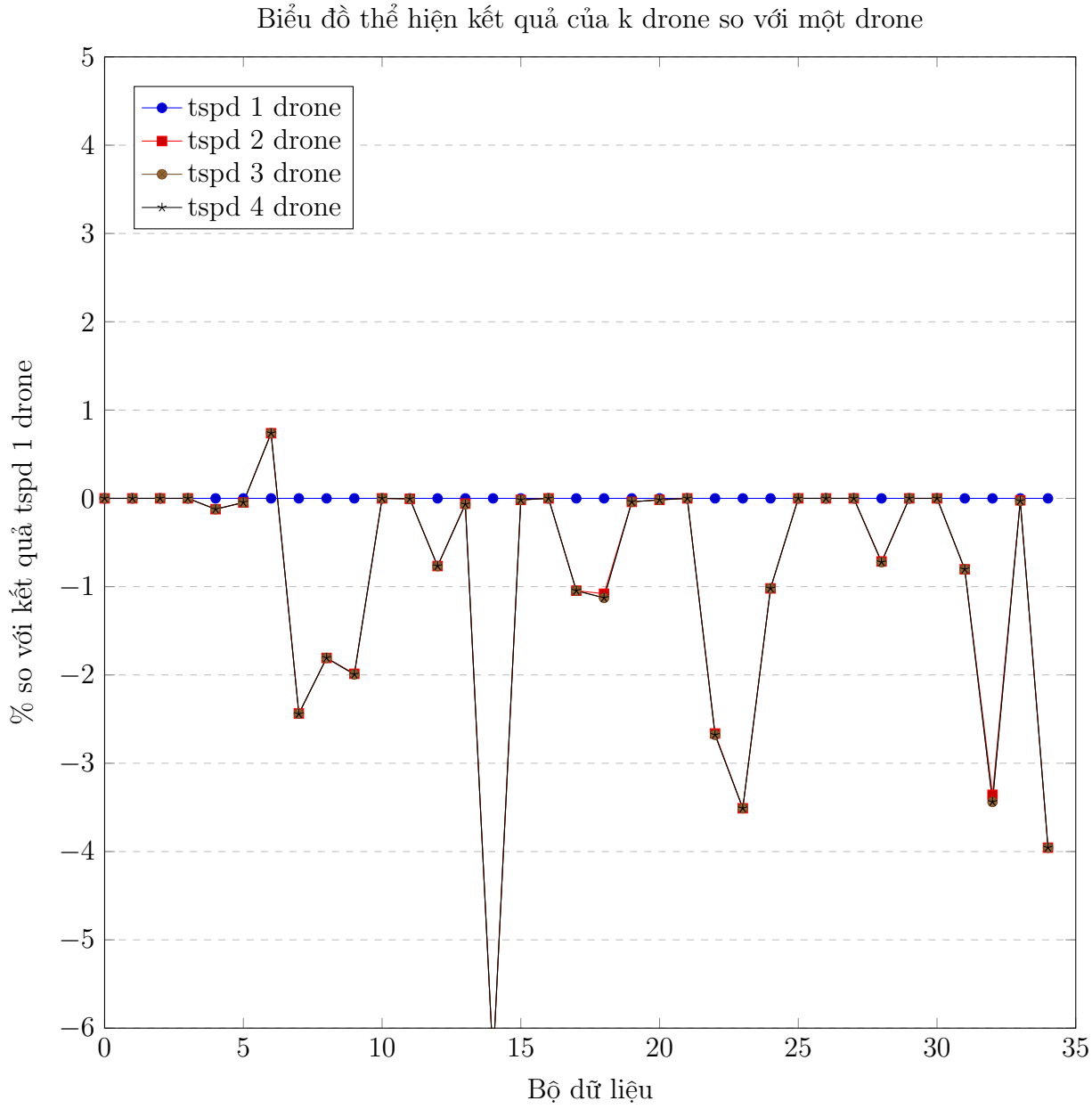
phí giảm thu được có lớn nhất hiện tại không. Nếu có, ta lưu lại chi phí và chuỗi điểm. Hàm $relocateAsDrone$ là hàm tương tự phần trước. Hàm $caculateSavingKpoint(j, i)$ là hàm tính chi phí giảm đi khi $truckRoute$ bị lấy i phần tử từ j.

Phần 4

Kết quả thử nghiệm

Trong phần này chúng tôi thực nghiệm với hai bộ dữ liệu một bộ theo bài báo [4] và một bộ sinh bằng cách chọn ngẫu nhiên trên googlemap. Các tham số của mỗi bộ như sau:

- Bộ 1 (theo bài báo):
  - 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
  - Sức chịu đựng của drone: 13,33333 km


Biểu đồ thể hiện kết quả của k drone so với một drone
tspd 1 drone
tspd 2 drone
tspd 3 drone
tspd 4 drone
% so với kết quả tspd 1 drone
Bộ dữ liệu
5
4
3
2
1
0
−1
−2
−3
−4
−5
−6
0
5
10
15
20
25
30
35

Phần 5

Thiết kế và xây dựng chương trình ứng dụng

Trong đồ án này chúng tôi xây dựng một chương trình ứng dựng 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 bootstap.

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 đông chính của ofbiz như : kết nỗi cơ sở dữ liệu, caching, render screens, quản ly 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 các component mới cho người sử dụng.

Để tạo một component mới trong hotdeploy ta đơn giản chị 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* chưa 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.

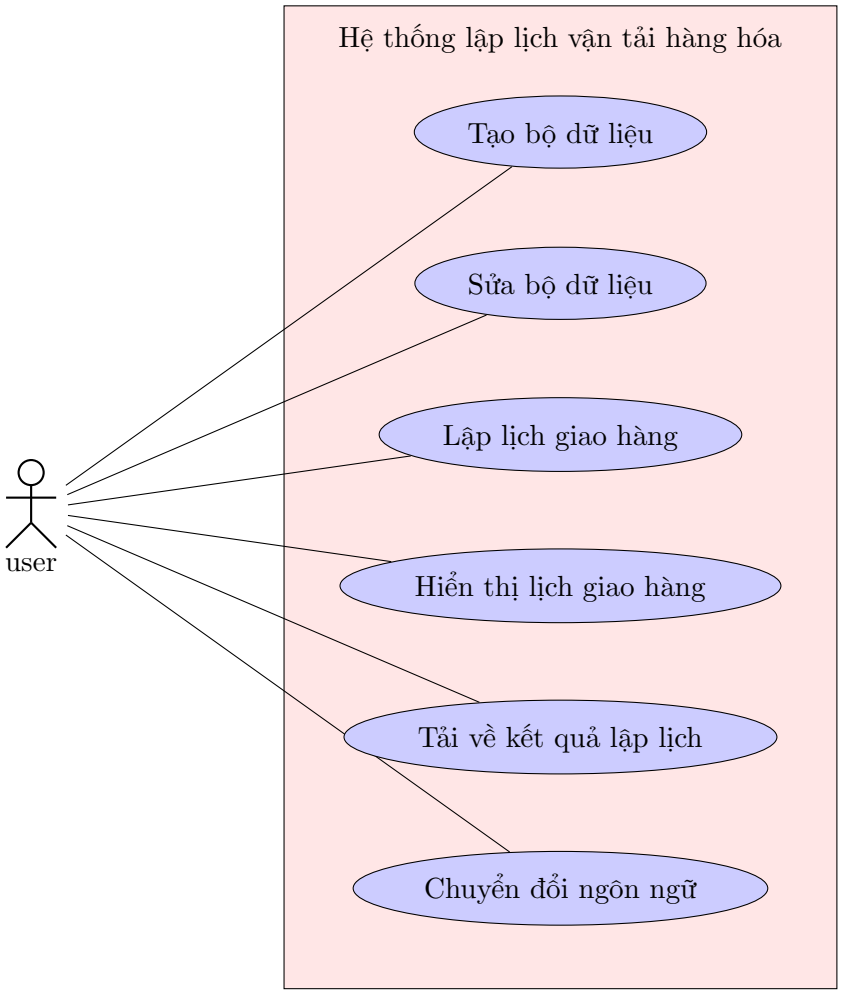
5.1.2 GoogleMap API

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 dược sử dụng ở client để hiển thị bản đồ cũng như những thao tác với bản đồ.

5.1.3 Boostrap

Boostrap là một framework kết hợp html, css và javascript cho phép ngươi sử dụng xây dựng view một cách đơn giản hơn mà vẫn đảm bảo reponsive. Boostrap 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:

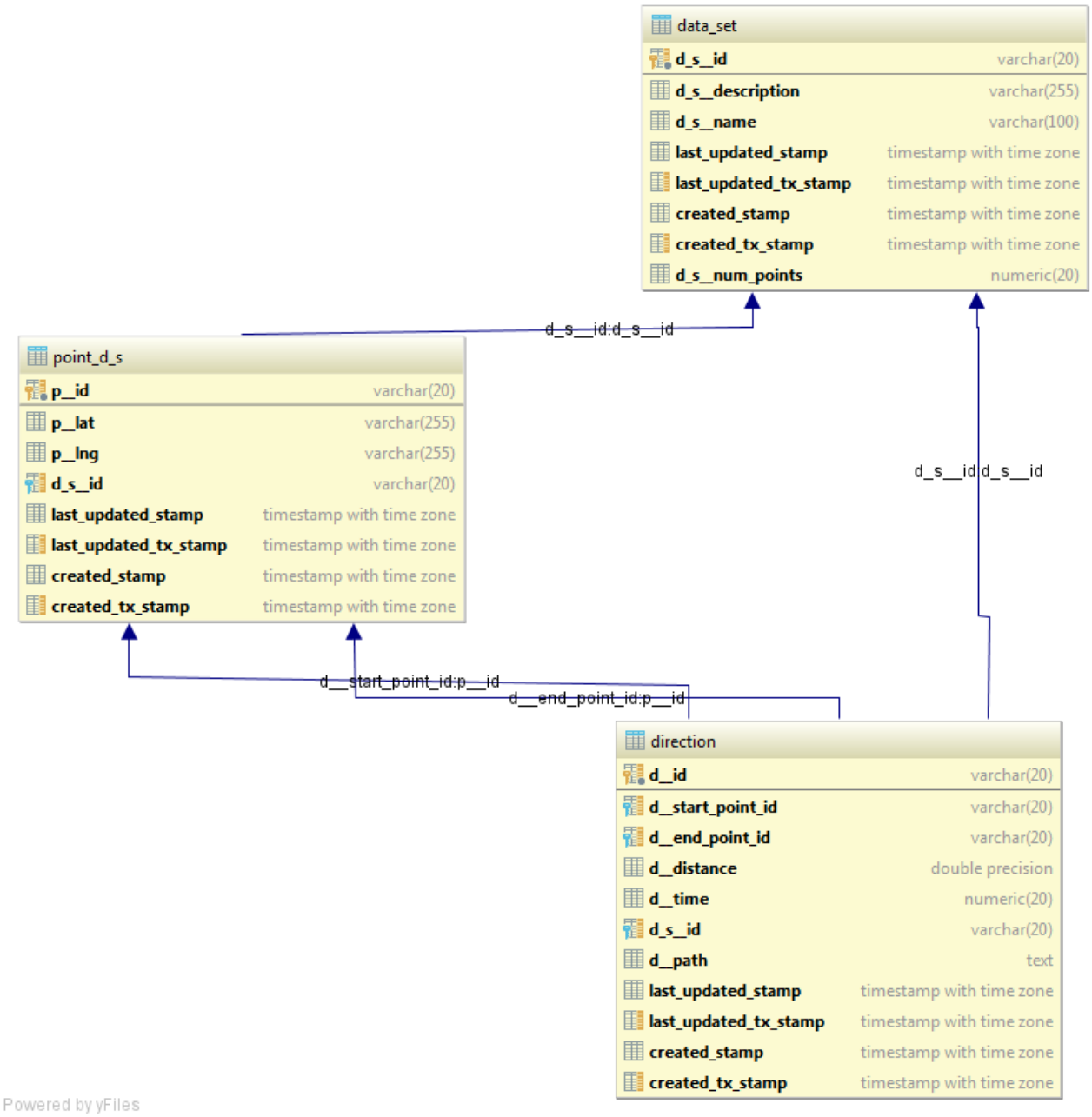
- Bảng point: chứa dữ liệu của một điểm giao hàng của khách hàng bào gồm các thông tin về tọa độ.
- Bảng direction: chứa dữ liệu về khoảng cách, thời gian, chuỗi đường đi giữa hai điểm khách hàng.
- Bảng dataset: chứa thông tin về tập điểm yêu cầu.

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

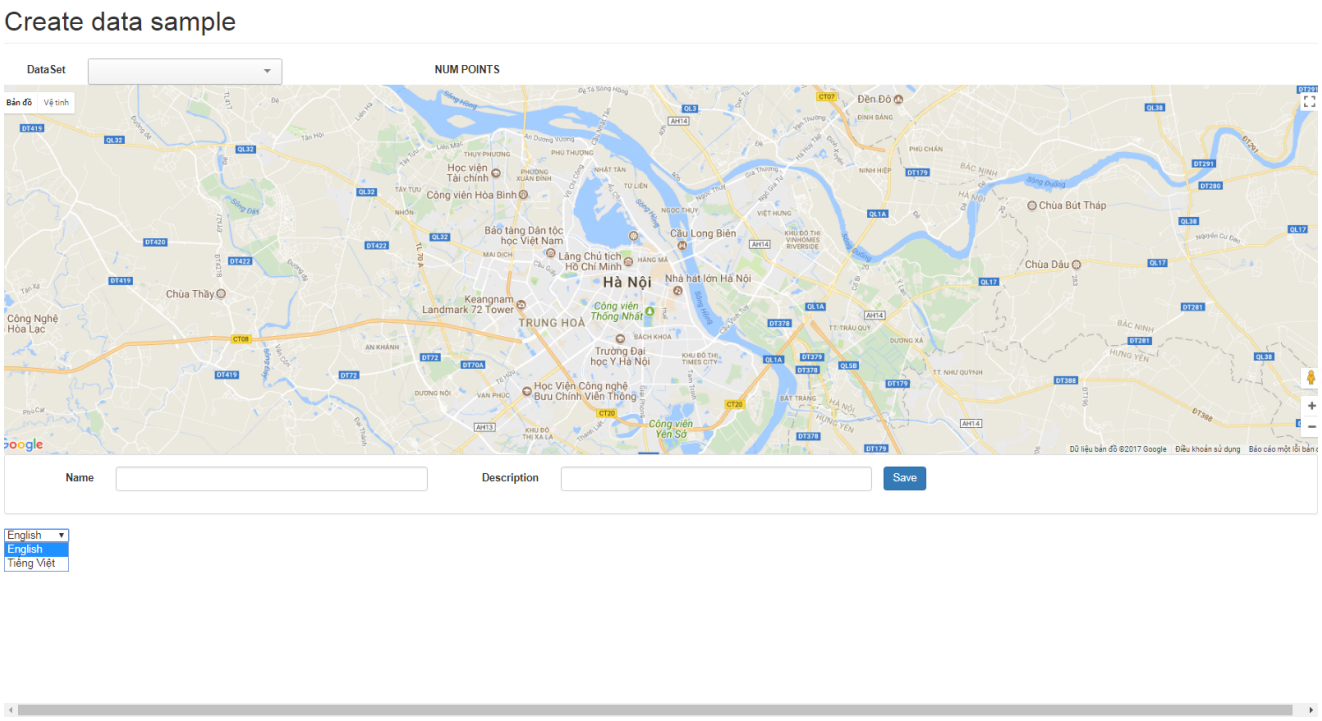
5.4 Các màn hình

Ứng dựn được thiết kế gồm các màn hình sau:

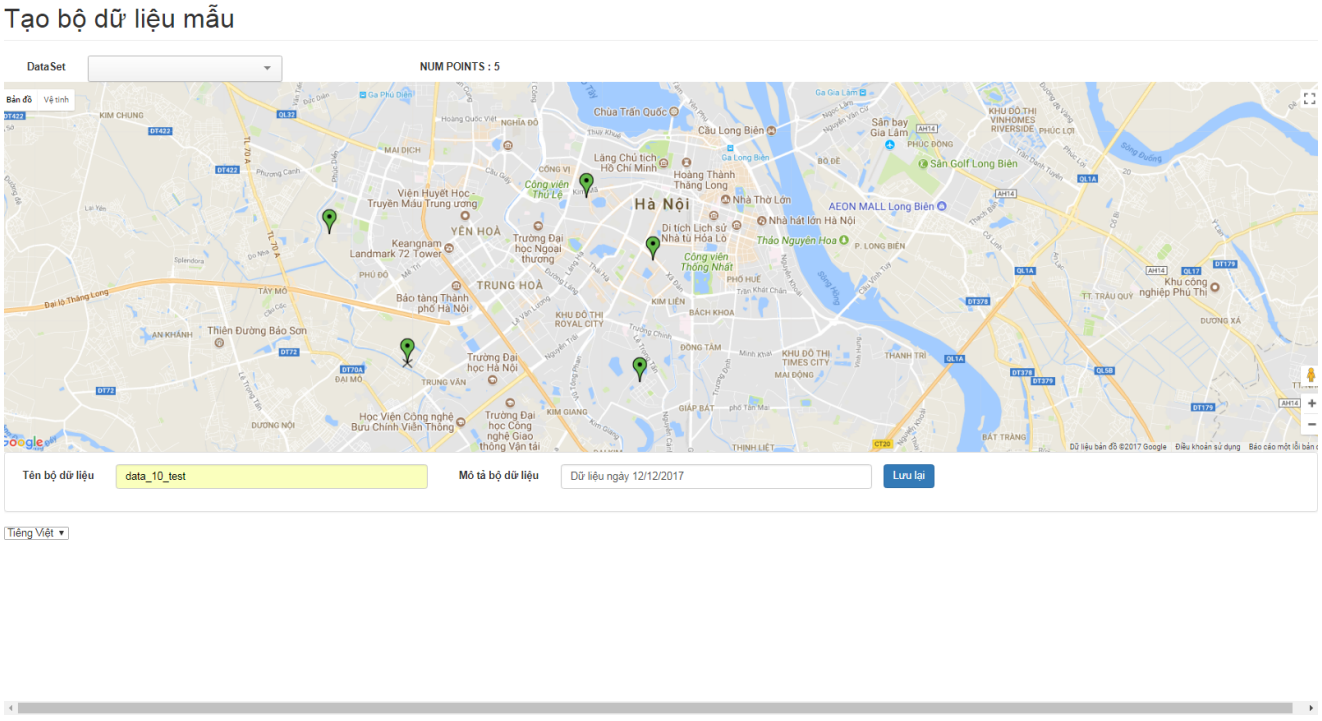
- Màn hình tạo và sửa bộ dữ liệu (5.4, 5.5).
- Màn hình lập lịch (5.6)
- Màn hình hiển thị kết quả (5.7, 5.8).



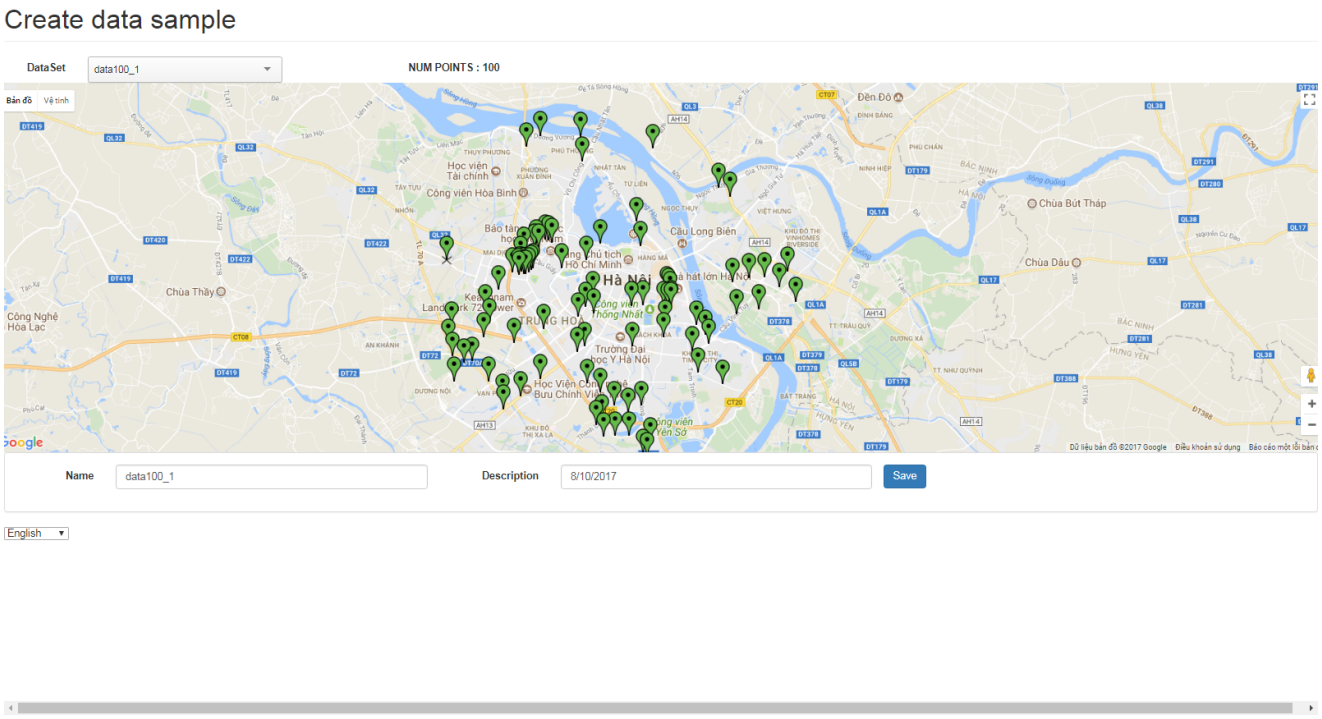
Hình 5.2: Thiết kế cơ sở dữ liệu của ứng dụng



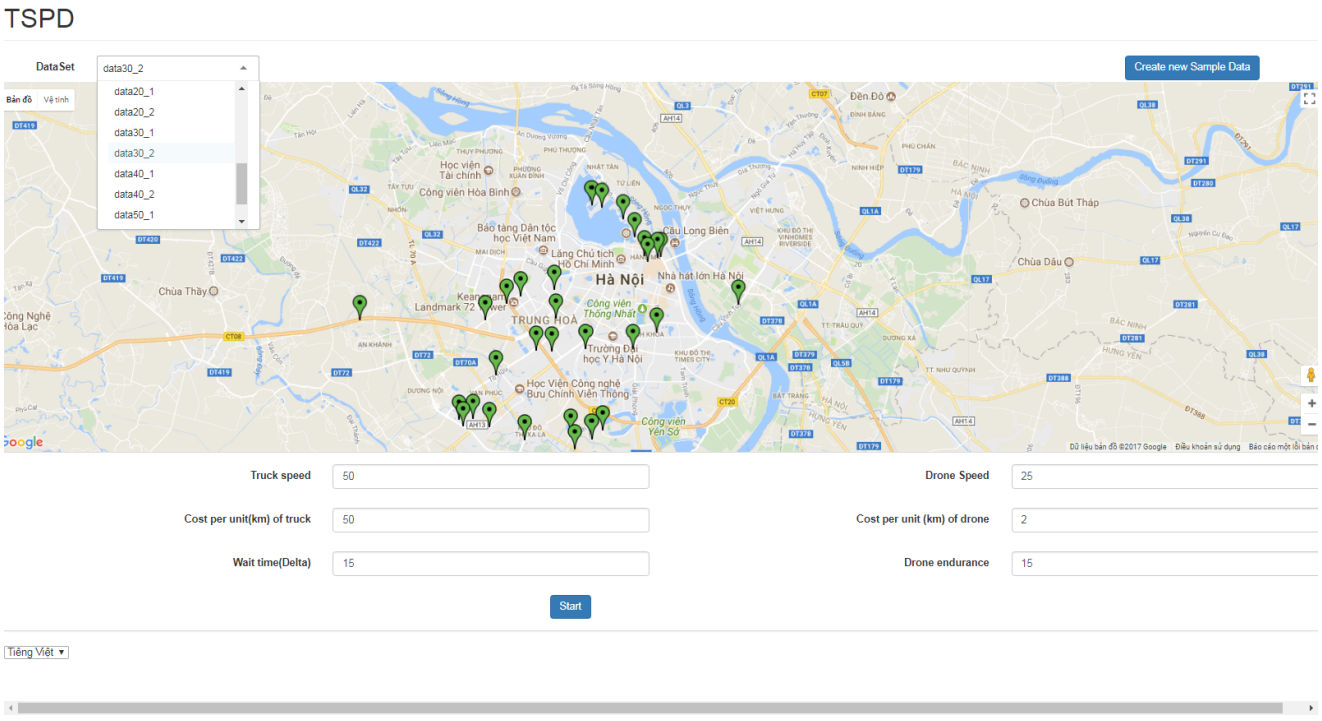
Hình 5.3: Màn hình thay đổi ngôn ngữ sang tiếng Anh



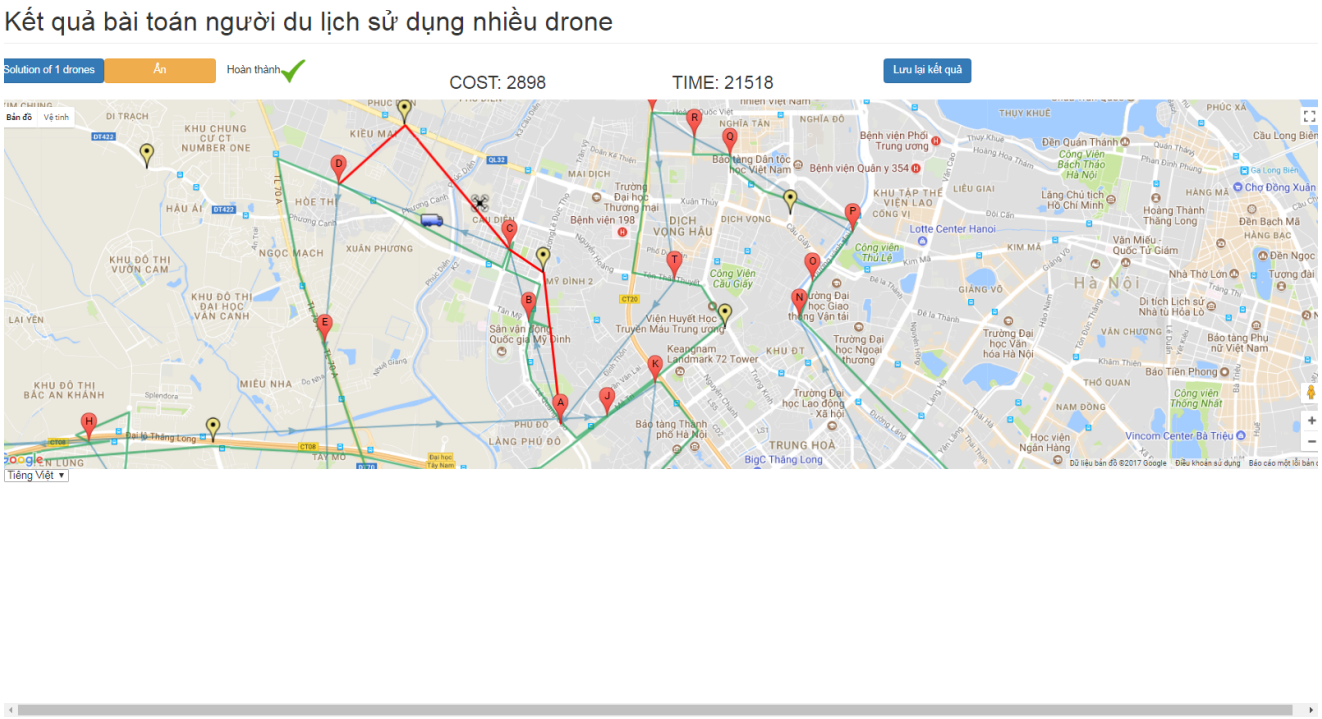
Hình 5.4: Màn hình tạo dữ liệu lập lịch



Hình 5.5: Màn hình chỉnh sửa tập dữ liệu



Hình 5.6: Màn hình lập lịch



Hình 5.7: Màn hình kết quả với một drone

Hình 5.8: Màn hình kết quả với bốn drone

Kết luận và hướng phát triển

Tài liệu tham khảo

[1] http://www.csplib.org/Problems/prob030/

[2] Slide Trí tuệ nhân tạo thầy Nguyễn Nhật Quang. Chương 5 : Thỏa mãn ràng buộc

[3] Stuart Russell and Peter Norvig *Artificial Intelligence: A Modern Approach* 2nd edition, Prentice Hall, page 137, 2003.

[4] 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

[5] S. Banker, Amazon and drones – here is why it will work (dec 2013). URL http://www.forbes.com/sites/stevebanker/2013/12/19/amazon-drones-here-is-why-it-will-work/

[6] Nguyễn Đức Nghĩa, Nguyễn Tô Thành *Toán rời rạc* 3rd edition, Nhà xuất bản đại học quốc gia Hà Nội, page 107-108, 2006.

[7] Slide Tối ưu hóa tổ hợp thầy Nguyễn Đức Nghĩa. Chương mở đầu, Bài toán vận tải.

[8] Slide Tìm kiếm cục bộ dựa trên ràng buộc thầy Phạm Quang Dũng. Chương 5: Contraint-base local search applications.

[9] Slide Phân tích và thiết kế thuật toán thầy Nguyễn Đức Nghĩa. Chương 5: Quy hoạch động, Chương 3: Greedy Algorithms.

[10] Francesca Rossi, Peter van Beek, Toby Walsh *Handbook of Constraint Programming* 1st edition, Elsevier Science, page 107-108, 2006.

[11] Bài giảng môn tính toán tiến hóa cô Huỳnh Thị Thanh Bình. Thuật giải di truyền.