ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC MÁY TÍNH

LÊ THẾ VIỆT

HUỲNH HOÀNG VŨ

KHÓA LUẬN TỐT NGHIỆP

GIẢI THUẬT ĐÀN KIẾN TỰ THÍCH ỨNG CHO BÀI TOÁN ĐIỀU HƯỚNG THU THẬP

CỬ NHÂN NGÀNH KHOA HỌC MÁY TÍNH

TP. HỒ CHÍ MINH, 2024

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC MÁY TÍNH

LÊ THẾ VIỆT – 20520093

HUỲNH HOÀNG VŨ – 20520864

KHÓA LUẬN TỐT NGHIỆP

GIẢI THUẬT ĐÀN KIẾN TỰ THÍCH ỨNG CHO BÀI TOÁN ĐIỀU HƯỚNG THU THẬP

CỬ NHÂN NGÀNH KHOA HỌC MÁY TÍNH

GIẢNG VIÊN HƯỚNG DẪN

TS. LƯƠNG NGỌC HOÀNG

TP. HỒ CHÍ MINH, 2024

DANH SÁCH HỘI ĐỒNG BẢO VỆ KHÓA LUẬN

Hội đồng chấm khóa luận tốt nghiệp, thành lập theo Quyết định số …… ngày …….. của Hiệu trưởng Trường Đại học Công nghệ Thông tin.

1. PGS.TS. Lê Đình Duy – Chủ tịch.
2. ThS. Nguyễn Thị Ngọc Diễm – Thư ký.
3. TS. Võ Nguyễn Lê Duy – Ủy viên.

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc Lập - Tự Do - Hạnh Phúc

TP. HCM, ngày…..tháng…..năm……..

NHẬN XÉT KHÓA LUẬN TỐT NGHIỆP

(CỦA CÁN BỘ HƯỚNG DẪN)

Tên khóa luận:

GIẢI THUẬT ĐÀN KIẾN TỰ THÍCH ỨNG CHO BÀI TOÁN ĐIỀU HƯỚNG THU THẬP

Nhóm SV thực hiện:		**Cán bộ hướng dẫn:**
Lê Thế Việt	20520093	TS. Lương Ngọc Hoàng
Huỳnh Hoàng Vũ	20520864	

Đánh giá Khóa luận

1. Về cuốn báo cáo:

Số trang	_____	Số chương	_____
Số bảng số liệu	_____	Số hình vẽ	_____
Số tài liệu tham khảo	_____	Sản phẩm	_____

Một số nhận xét về hình thức cuốn báo cáo:

.....

.....

.....

.....

2. Về nội dung nghiên cứu:

.....

.....

.....

3. Về chương trình ứng dụng:

.....

.....

.....

4. Về thái độ làm việc của sinh viên:

.....

.....

.....

<u>Đánh giá chung:</u>

.....

.....

<u>Điểm từng sinh viên:</u>

Lê Thế Việt**:……../10**

Huỳnh Hoàng Vũ**:……../10**

Người nhận xét

(Ký tên và ghi rõ họ tên)

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc Lập - Tự Do - Hạnh Phúc

TP. HCM, ngày…..tháng…..năm……..

NHẬN XÉT KHÓA LUẬN TỐT NGHIỆP

(CỦA CÁN BỘ PHẢN BIỆN)

Tên khóa luận:

GIẢI THUẬT ĐÀN KIẾN TỰ THÍCH ỨNG CHO BÀI TOÁN ĐIỀU HƯỚNG THU THẬP

Nhóm SV thực hiện:		**Cán bộ phản biện:**
Lê Thế Việt	20520093	TS. Võ Nguyễn Lê Duy
Huỳnh Hoàng Vũ	20520864	

Đánh giá Khóa luận

1. Về cuốn báo cáo:

Số trang	_____	Số chương	_____
Số bảng số liệu	_____	Số hình vẽ	_____
Số tài liệu tham khảo	_____	Sản phẩm	_____

Một số nhận xét về hình thức cuốn báo cáo:

.....

.....

.....

.....

2. Về nội dung nghiên cứu:

.....

.....

.....

3. Về chương trình ứng dụng:

.....

.....

.....

4. Về thái độ làm việc của sinh viên:

.....

.....

.....

<u>Đánh giá chung:</u>

.....

.....

<u>Điểm từng sinh viên:</u>

Lê Thế Việt**:……../10**

Huỳnh Hoàng Vũ**:……../10**

Người nhận xét

(Ký tên và ghi rõ họ tên)

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc Lập - Tự Do - Hạnh Phúc

ĐỀ CƯƠNG CHI TIẾT

TÊN ĐỀ TÀI: GIẢI THUẬT ĐÀN KIẾN TỰ THÍCH ỨNG CHO BÀI TOÁN ĐIỀU HƯỚNG THU THẬP

Cán bộ hướng dẫn: TS. Lương Ngọc Hoàng

Thời gian thực hiện: Từ ngày 11/09/2023 đến ngày 31/12/2023

Sinh viên thực hiện:

Lê Thế Việt – 20520093

Huỳnh Hoàng Vũ – 20520864

Nội dung đề tài:

1. Đối tượng

Bài toán Điều Hướng Thu Thập (Thief Orienteering Problem, ThOP) là một bài toán tối ưu hóa tổ hợp mới và thách thức, kết hợp các đặc điểm của hai bài toán NP-khó cổ điển: Bài toán điều hướng (Orienteering Problem, OP) và bài toán ba lô (Knapsack Problem, KP). ThOP được giới thiệu bởi Santos và Chagas vào năm 2018 như một sự mở rộng của Bài toán du lịch thu thập (Traveling Thief Problem, TTP), mà chính nó là một sự kết hợp của bài toán người bán hàng du lịch (Traveling Salesman Problem, TSP) và KP. ThOP nhằm mô hình hóa các tình huống thực tế hơn, nơi mà người thu thập phải lập kế hoạch cho tuyến đường với thành phố bắt đầu và kết thúc cố định nhằm chọn các vật phẩm trong các thành phố viếng thăm dưới thời gian di chuyển và sức chứa của ba lô giới hạn.

2. Phạm vi

Bộ trường hợp đánh giá do Santos và Chagas thiết kế khi giới thiệu ThOP, gồm 432 thể hiện của bài toán Điều Hướng Thu Thập với đa dạng về số lượng thành phố, số lượng vật phẩm, kích thước ba lô, thời gian di chuyển tối đa và sự tương quan giữa trọng lượng và lợi nhuận của các vật phẩm.

3. Mục tiêu

- Đề xuất phương pháp cải thiện thuật toán ThOP tân tiến để mang lại **kết quả tốt hơn trong cùng chi phí tính toán**.
- Đề xuất phương pháp giúp **các tham số tự thích ứng** mà không cần điều chỉnh riêng cho từng nhóm trường hợp cụ thể.

4. Nội dung và phương pháp thực hiện

Nội dung 1: Khảo sát các phương pháp hiện có. Phương pháp thực hiện:

Chúng tôi xem xét các bài báo khoa học hiện có về ThOP, tập trung vào các phương pháp heuristic và metaheuristic đã được đề xuất để giải quyết bài toán với kết quả xấp xỉ. Chúng tôi xác định các thách thức và hạn chế chính của các phương pháp này, như sự phụ thuộc vào các tham số cố định hoặc điều chỉnh thủ công.

Nội dung 2: Thiết kế thuật toán. Phương pháp thực hiện:

Chúng tôi đề xuất phương pháp cải tiến cho hai bài toán con, phát triển dựa trên các biến thể của giải thuật đàn kiến (Ant Colony Optimization, ACO) cho bài toán con OP và các thuật toán sắp xếp ba lô cho bài toán con KP. Chúng tôi thử nghiệm các kỹ thuật giúp các tham số tự động thích nghi với các trường hợp bài toán trong thời gian chạy.

Nội dung 3: Thực nghiệm và so sánh. Phương pháp thực hiện:

Chúng tôi tiến hành các thí nghiệm phương pháp đề xuất trên một bộ dữ liệu thử nghiệm riêng cho ThOP, có sự khác biệt về số lượng thành phố, số lượng vật phẩm, giới hạn thời gian, dung lượng ba lô, phân bố vật phẩm, tương quan vật phẩm. Chúng tôi so sánh phương pháp của chúng tôi với nhiều phương pháp mới nhất hiện có, như ILS, BRKGA, ACO, ACO++.

5. Kết quả mong đợi

Một thuật toán mới cho bài toán Điều Hướng Thu Thập mang lại **kết quả tốt hơn trong cùng chi phí tính toán** so với các thuật toán hiện có, đồng thời, sở hữu **khả năng thích ứng các tham số** của mình nhờ vào các phương pháp đề xuất.

Kế hoạch thực hiện:

Công việc	Thời gian thực hiện	Phân công
Khảo sát các phương pháp hiện có	11/09 - 31/10	Lê Thế Việt Huỳnh Hoàng Vũ
Tích hợp, đề xuất các kỹ thuật điều chỉnh tham số	15/10 - 30/11	Lê Thế Việt
Tích hợp, đề xuất các kỹ thuật giúp giảm chi phí tính toán	15/10 - 30/11	Huỳnh Hoàng Vũ
Thực nghiệm và so sánh	31/10 - 31/12	Lê Thế Việt Huỳnh Hoàng Vũ
Viết báo cáo	30/11 - 31/12	Lê Thế Việt Huỳnh Hoàng Vũ

Xác nhận của CBHD (Ký tên và ghi rõ họ tên)	**TP. HCM, ngày….tháng …..năm…..** **Sinh viên** (Ký tên và ghi rõ họ tên)

LỜI CẢM ƠN

Chúng tôi chân thành bày tỏ lòng biết ơn đối với thầy Hoàng, TS. Lương Ngọc Hoàng, với sự kiên nhẫn và tận tâm hỗ trợ. Dưới sự hướng dẫn của thầy, chúng tôi đã đạt được kết quả nghiên cứu ấn tượng. Mặc dù nhiều lúc tiến độ thực hiện đề tài của chúng tôi gặp phải nhiều chậm trễ, nhưng thầy vẫn luôn tận tình hướng dẫn. Chân thành cảm ơn thầy.

Ngoài ra, chúng tôi cũng muốn bày tỏ lòng biết ơn đối với các thành viên trong nhóm nghiên cứu ELO và các bạn thuộc lớp KHTN2020. Các bạn là nguồn hỗ trợ tinh thần đáng quý khi được đồng hành cùng các bạn trong suốt quá trình học đại học cũng như quá trình thực hiện đề tài.

Cuối cùng, chúng tôi xin bày tỏ lòng biết ơn đến các thầy cô giảng viên và cán bộ công chức tại trường Đại học Công Nghệ Thông Tin. Sự nỗ lực hàng ngày của thầy cô đã tạo nên một môi trường học tập rèn luyện vô cùng bổ ích.

MỤC LỤC

DANH SÁCH HÌNH VẼ

DANH SÁCH BẢNG

DANH SÁCH TỪ VIẾT TẮT

AACO-NC	**A**daptive **A**nt **C**olony **O**ptimization with **n**ode **c**lustering
ACO++	MAX-MIN ant colony optimization for ThOP
ACO	**A**nt **C**olony **O**ptimization
BRKGA	**B**iased **R**andom-**K**ey **G**enetic **A**lgorithm
CMA-ES	**C**ovariance **M**atrix **A**daptation **E**volution **S**trategy
ES	**E**volution **S**trategy
GA	**G**enetic **A**lgorithm
ILS	**I**terated **L**ocal **S**earch
KP	**K**napsack **P**roblem
MINLP	**M**ixed **I**nteger Non-**L**inear **P**rogramming
MMAS	**M**AX-**M**IN **a**nt **s**ystem
OP	**O**rienteering **P**roblem
SAAS	**S**elf-**A**daptive **A**nt **S**ystem
TSP	**T**raveling Salesman Problem
TTP	**Th**ief **T**raveling **P**roblem
ThOP	**Th**ief **O**rienteering **P**roblem

TÓM TẮT

Bài toán Điều Hướng Thu Thập (Thief Orienteering Problem) là một vấn đề đa thành phần với hai bài toán con tương tác là bài toán Ba Lô (Knapsack Problem) và bài toán Điều Hướng (Orienteering Problem). ACO++, một metaheuristic hiện đại cho ThOP, kết hợp thuật toán MAX-MIN Ant System để xây dựng đường đi, một thuật toán ngẫu nhiên cho việc tạo chiến lược thu thập, và phương pháp 2-OPT cho tìm kiếm cục bộ (local search). Tuy nhiên, hiệu suất xuất sắc được báo cáo của ACO++ được đạt được bằng cách sử dụng các bộ giá trị tham số khác nhau đã được điều chỉnh tỉ mỉ cho từng nhóm cụ thể của các trường hợp đánh giá. Trong công trình này, chúng tôi đề xuất một biến thể tự thích ứng mới của ACO++. Không đòi hỏi quá trình điều chỉnh phức tạp, phương pháp của chúng tôi sử dụng các cơ chế thích ứng để điều chỉnh các tham số cho từng trường hợp vấn đề cụ thể trong quá trình chạy thuật toán. Chúng tôi cũng sử dụng kỹ thuật bay hơi lười biếng và tận dụng phân cụm thứ bậc để cải thiện hiệu suất của đàn kiến trong việc khám phá không gian tìm kiếm. Trong số 432 trường hợp đánh giá, phương pháp Self-Adaptive Ant System (SAAS) của chúng tôi tạo ra kết quả vượt trội hơn so với các phương pháp tân tiến trước đó. Mã nguồn đã được công bố tại `https://github.com/ELO-Lab/SAAS-HC`.

Chương 1

TỔNG QUAN

1.1 Đối tượng nghiên cứu

Các bài toán đa thành phần liên quan đến nhiều thành phần tương tác và phổ biến trong các tình huống thực tế, như định tuyến xe có ràng buộc về tải trọng [13] hoặc tối ưu hóa việc sử dụng vật liệu trong lịch trình sản xuất [17]. Những vấn đề này tạo thách thức do sự phụ thuộc lẫn nhau giữa các thành phần và thường liên quan đến các nhiệm vụ tối ưu hóa NP-Hard như đóng gói, lập lịch và định tuyến [2]. Bài toán Người Thu Thập Du Lịch (Thief Traveling Problem, TTP) [1] là một vấn đề đa thành phần đặc trưng kết hợp bài toán Ba Lô (Knapsack Problem, KP) và bài toán Người Bán Hàng Du Lịch (Traveling Salesman Problem, TSP).

Trong công trình này, chúng tôi giải quyết bài toán Điều Hướng Thu Thập (Thief Orienteering Problem, ThOP) [19], một vấn đề đa thành phần được biến thể từ TTP với các tương tác và ràng buộc mới. ThOP kết hợp bài toán Điều hướng (Orienteering Problem, OP) [10] và KP. Mục tiêu là tìm một tuyến đường từ thành phố xuất phát đến thành phố đích (thành phần OP) sao cho tối đa hóa tổng lợi nhuận của các vật phẩm được thu thập (thành phần KP), đồng thời tuân thủ giới hạn sức chứa của ba lô và giới hạn thời gian di chuyển. ThOP được lấy cảm hứng từ TTP và xem xét các tình huống thực tế nơi người thu thập không cần phải ghé thăm tất cả các thành phố và tương tác được xác định bởi thời gian và giới hạn của ba lô. Vấn đề này được ứng dụng trong

logistics ngược (reverse logistics) [19], nơi công ty cần thu nhận hàng hóa từ khách hàng, mỗi món hàng mang lại lợi ích cụ thể, dưới sự giới hạn của khả năng chở hàng và giờ làm việc của tài xế.

1.2 Phạm vi nghiên cứu

Chúng tôi tập trung nghiên cứu các giải pháp với hướng tiếp cận heuristic, metaheuristic cho ThOP. Các thuật toán heuristic giúp ta đạt được kết quả tương đối tốt với chi phí thời gian chấp nhận được. Chúng trái ngược với các phương pháp chính xác (exact methods), cái mà sẽ đảm bảo tìm được lời giải tối ưu. Đối với bài toán NP-Hard như ThOP, thời gian chạy cần thiết của phương pháp chính xác sẽ tăng nhanh chóng với kích thước của bài toán dẫn đến không khả thi để ứng dụng vào thực tế. Do đó, nghiên cứu giải pháp heuristic sẽ có tính ứng dụng cao hơn.

Bên cạnh hướng tiếp cận thuật toán, chúng tôi đã tiến hành thực nghiệm trên Bộ trường hợp đánh giá do Santos và Chagas thiết kế [19] khi giới thiệu ThOP. Nó bao gồm 432 thể hiện của bài toán Điều Hướng Thu Thập với đa dạng về số lượng thành phố, số lượng vật phẩm, kích thước ba lô, thời gian di chuyển tối đa và sự tương quan giữa trọng lượng và lợi nhuận của các vật phẩm.

1.3 Các công trình liên quan

Các công trước đây cho ThOP đa dạng về hướng tiếp cận. Một phương pháp chính xác Lập Trình Phi Tuyến Tính Hỗn Hợp (Mixed Integer Non-Linear Programming, MINLP) [19]. Một phương pháp tìm kiếm cục bộ Tìm Kiếm Cục Bộ Tuần Tự (Iterated Local Search, ILS) [19]. Hai phương pháp thuật giải di truyền Thuật Giải Di Truyền Điểm Thiên Lệch Ngẫu Nhiên (Biased Random-Key Genetic Algorithm, BRKGA) [19], Thuật Giải Di Truyền (Genetic Algorithm, GA) [9]. Một phương pháp thông

minh bầy đàn Giải Thuật Tối Ưu Hóa Đàn Kiến (Ant Colony Optimization, ACO) [4]. Và sự kết hợp giữa thông minh bầy đàn và tìm kiếm cục bộ ACO++ [3].

Chagas và Wagner [3] đã chỉ ra rằng ACO++ vượt trội hơn so với các thuật toán khác trong hơn 90% số lượng trường hợp (trường hợp đánh giá). Để đạt được những kết quả xuất sắc này, ACO++ trải qua quá trình điều chỉnh chi tiết để xác định cấu hình siêu tham số phù hợp cho mỗi nhóm trường hợp trong benchmark (bộ đánh giá) của ThOP.

1.4 Mục đích nghiên cứu

Trong công trình này, chúng tôi đề xuất một thuật toán cải tiến dựa trên ACO++, Giải Thuật Đàn Kiến Tự Thích Ứng (Self-Adaptive Ant System, SAAS) có khả năng điều chỉnh động các tham số của mình theo đặc trưng của mỗi trường hợp và quá trình tìm kiếm. Sự cải tiến này giúp giảm thiểu tính tốn thời gian và phi thực tế của việc điều chỉnh nhiều cấu hình siêu tham số. Thuật toán của chúng tôi sử dụng một cấu hình duy nhất cho toàn bộ trường hợp trong ThOP benchmark, đồng thời mang lại hiệu suất vượt trội.

Đề cải tiến, chúng tôi tích hợp bốn kỹ thuật vào ACO++. Hai cơ chế kiểm soát tham số, tự thích ứng (self-adaptive) và thích ứng (adaptive), giúp cải thiện khả năng thích nghi của thuật toán. Hai kỹ thuật Bay hơi lười biếng và Phân cụm thứ bậc giúp giảm độ phức tạp về thời gian.

Cơ chế tự thích ứng sử dụng chiến lược tiến hóa (Evolution Strategy, ES), trong khi đó cơ chế thích ứng sử dụng thông tin sự đa dạng lợi nhuận để điều chỉnh các tham số (Bảng 3.1). Các tham số được điều chỉnh đồng thời với quá trình tìm kiếm lời giải. Phân cụm phân cấp được sử dụng để giảm chi phí xây dựng các tuyến đường. Bay hơi lười biếng được sử dụng để giảm thời gian cần thiết để bay hơi dấu vết pheromone của kiến.

Chương 2

KIẾN THỨC NỀN TẢNG

Ở chương này, chúng tôi sẽ cung cấp các kiến thức được sử dụng xuyên suốt trong khóa luận này. Ở phần 2.1, chúng tôi sẽ mô tả chi tiết bài toán mà chúng tôi giải quyết, bài toán Điều Hướng Thu Thập. Chúng tôi sẽ trình bày thế nào là một bài toán tối ưu hóa đa thành phần, các bài toán con, các mục tiêu đặt ra khi giải quyết bài toán này. Ở phần 2.2, chúng tôi sẽ trình bày các kiến thức nền tảng về giải thuật tối ưu hóa đàn kiến và biến thể MAX-MIN ACO của nó, thuật toán mà chúng tôi sử dụng làm nền tảng để phát triển nên thuật toán chúng tôi đề xuất. Bên cạnh đó chúng tôi trình bày các kiến thức cơ sở của các phương pháp chúng tôi sử dụng để thích ứng các tham số (phần 2.3, 2.4 và 2.5) và làm giảm độ phức tạp của thuật toán (phần 2.6).

2.1 Bài toán Điều Hướng Thu Thập

2.1.1 Mô tả bài toán

Bài toán chúng tôi giải quyết ở công trình này là bài toán Điều Hướng Thu Thập (Thief Orienteering Problem, ThOP), là một biến thể của bài toán Điều Hướng (Orienteering Problem) được lấy cảm hứng từ bài toán Người Thu Thập Du Lịch (Travelling Thief Problem, TTP). TTP được đề xuất bởi Bonyadi, Michalewicz và Barone [1]. Đây là sự kết hợp của hai bài toán kinh điển nổi tiếng: bài toán Người Bán Hàng Du lịch (Traveling Salesman Problem, TSP) và bài toán Ba Lô (Knapsack Problem, KP). Điểm

chính mà tác giả trình bày để đề xuất TTP là các thang đo của các bài toán NP-Hard kinh điển không phản ánh đúng các đặc điểm chính của các vấn đề thực tế, nên các thuật toán metaheuristics hiệu quả mà chúng ta có cho những thang đo đó không nhất thiết hiệu quả cho các vấn đề thực tế. Tác giả cho rằng sự phức tạp của các vấn đề thực tế không chỉ đến từ kích thước của chúng, mà chủ yếu là do chúng là sự kết hợp của hai hoặc nhiều vấn đề tối ưu hóa con, và vì những vấn đề con đó là tương phụ thuộc lẫn nhau. Điều này có nghĩa là việc giải quyết một vấn đề con ảnh hưởng đến chất lượng của việc giải quyết các vấn đề con khác, do đó chúng không nên được giải quyết độc lập. Trong TTP, một người thu thập phải ghé thăm mỗi thành phố trong một tập hợp gồm n thành phố (bài toán con TSP) và trong quá trình ghé thăm có thể thu thập các vật phẩm nằm ở các thành phố đó để đựng trong ba lô của mình (bài toán con KP). Tuy nhiên, khi các vật phẩm được thu thập, balo trở nên nặng hơn và người thu thập đi chậm hơn. Ba lô của người thu thập thuê, và giá phải trả là tỉ lệ với thời gian thuê. Người thu thập sau đó phải tối đa hóa tổng lợi nhuận của các vật phẩm đã thu thập và đồng thời giảm thiểu tổng thời gian của hành trình. Tác giả chỉ ra rằng giải pháp tối ưu của một trường hợp TTP có thể không bao gồm giải pháp tối ưu của bài toán con KP hoặc bài toán con TSP, chứng minh sự phụ thuộc lẫn nhau của các bài toán con trong đó.

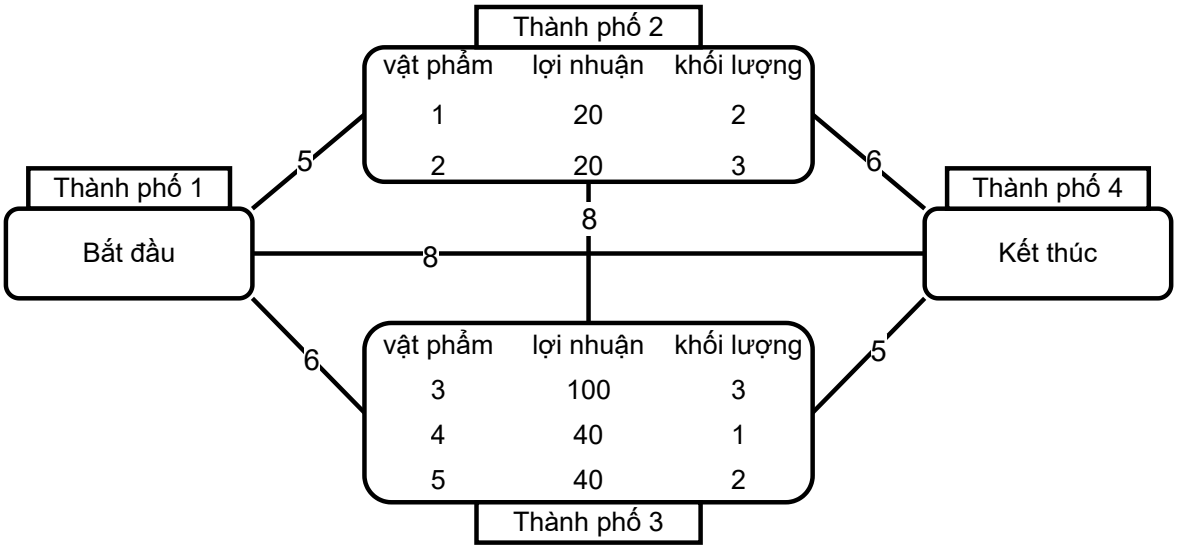
Lấy cảm hứng từ TTP, Santos và Changas [19] giới thiệu ở một bài toán đa thành phần mới, đó là bài toán Điều Hướng Thu Thập (ThOP), được xây dựng trên cơ sở của bài toán Điều Hướng (OP) thay vì bài toán Người Bán Hàng Du Lịch như TTP. Bài toán Điều Hướng dựa trên một trò chơi thể thao địa hình. Trong trò chơi, các đối thủ bắt đầu từ một điểm cho trước, đi qua một khu vực ghé thăm các điểm kiểm tra và phải quay lại một điểm kiểm soát trong một khoảng thời gian nhất định. Họ có một bản đồ của khu vực và phải tự quyết định tuyến đường qua các điểm kiểm tra dựa trên kỹ năng định hình và cấp độ thể dục của họ. Trong OP, mỗi điểm kiểm tra có điểm

số, vì vậy mục tiêu là tìm ra tuyến đường tối đa hóa tổng điểm số, tức là tổng số điểm của các điểm kiểm tra đã ghé thăm là tối đa. Trong biến thể ThOP, một đối thủ hay chúng tôi gọi là người thu thập không ghi điểm chỉ bằng cách ghé thăm các điểm trên bản đồ, mà phải thu thập các vật phẩm tại các điểm và mang theo chúng đến điểm kết thúc. Mỗi đối thủ có một ba lô với sức chứa giới hạn cho các vật phẩm được thu thập. Hơn nữa, tốc độ của người thu thập bị ảnh hưởng trực tiếp bởi trọng lượng của ba lô. Như ở bài toán TTP, khi các vật phẩm được thu thập, ba lô sẽ nặng dần, và tốc độ của người thu thập sẽ giảm xuống. Gọi v_{min}, v_{max} là tốc độ tối thiểu và tốc độ tối đa mà người thu thập di chuyển tương ứng khi ba lô đạt sức chứa giới hạn W hoặc ba lô rỗng. Tốc độ di chuyển v của người thu thập với ba lô có sức nặng w, $0 \leq w \leq W$ được tính bằng $v = v_{max} - w \cdot (v_{max} - v_{min})/W$. Lưu ý rằng giá trị $(v_{max} - v_{min})/W$ là hằng số và thể hiện độ giảm của tốc độ với một đơn vị cân nặng của ba lô.

Mục tiêu của ThOP là cung cấp một đường đi từ thành phố bắt đầu 1 đến thành phố kết thúc n, cũng như một tập các vật phẩm được chọn từ các thành phố đã ghé thăm trong suốt hành trình để tối đa hóa tổng lợi nhuận đã lấy, đồng thời đảm bảo rằng sức chứa của balo không vượt quá sức chứa giới hạn W và tổng thời gian di chuyển của người thu thập nằm trong giới hạn thời gian T. Người thu thập không cần phải ghé thăm tất cả các thành phố.

Để hiểu rõ bài toán Điều Hướng Thu Thập ThOP, chúng tôi minh họa trong hình 2.1 một ví dụ nhỏ của một trường hợp ThOP liên quan đến 4 thành phố và 5 vật phẩm. Lưu ý rằng không có vật phẩm nào ở thành phố bắt đầu (1) và thành phố kết thúc (4), các số vật phẩm có trọng lượng và lợi nhuận khác nhau được phân phối ở các thành phố khác (2 và 3). Các khoảng cách từ mỗi cặp thành phố được cho trong các cạnh. Dưới đây, chúng tôi trình bày chi tiết một số lời giải cho trường hợp này. Chúng tôi đặt các ràng buộc của ví dụ này như sau: $v_{min} = 0.1$, $v_{max} = 1.0$, $W = 3$ và $T = 75$.

Chúng tôi biểu diễn một lời giải cho bài toán ThOP dưới dạng hai phần (π, z).



Hình 2.1: Minh họa một ví dụ của bài toán Điều Hướng Thu Thập bao gồm 4 thành phố, 5 vật phẩm.

Phần đầu tiên bao gồm tuyến đường di chuyển của người thu thập $\pi = \langle 1, \pi_2, \pi_3, ..., \pi_k, n \rangle$, một véc tơ chứa danh sách thành phố đã ghé thăm theo thứ tự di chuyển. Lưu ý rằng thành phố đầu tiên và cuối cùng được cố định cho mọi lời giải hợp lệ. Phần thứ hai là chiến lược thu thập vật phẩm $z = \langle z_1, z_2, ..., z_m \rangle$, một véc tơ nhị phân đại diện cho trạng thái của các vật phẩm ($z_i = 1$ nếu vật phẩm i được lấy, và 0 nếu ngược lại). Theo biểu diễn này, hãy xem xét các lời giải ThOP sau cho trường hợp đã mô tả ở trên:

- $(\langle 1, 2, 3, 4 \rangle, \langle 1, 0, 0, 1, 0 \rangle)$: là một lời giải hợp lệ với tổng lợi nhuận là $20 + 40 = 60$. Tổng khối lượng các vật phẩm được thu thập là 3 và thời gian di chuyển là 75, thỏa mãn cả hai ràng buộc giới hạn sức chứa ba lô W và thời gian di chuyển T. Tổng thời gian di chuyển được tính như sau:
  - di chuyển từ thành phố bắt đầu đến thành phố thứ 2 với tốc độ tối đa: thời gian di chuyển là $d_{12}/v_{max} = 5/1.0 = 5$

  - tại thành phố thứ 2, người thu thập lấy vật phẩm 1: vận tốc giảm xuống $v = 1.0 - 2 \times (1.0 - 0.1)/3 = 0.4$
  - di chuyển từ thành phố thứ 2 sang thành phố thứ 3: tổng thời gian di chuyển là $5 + d_{23}/v = 5 + 8/0.4 = 5 + 20 = 25$
  - tại thành phố thứ 3, vật phẩm thứ 4 được lấy: vận tốc giảm xuống $v = 1.0 - 3 \times (1.0 - 0.1)/3 = 0.1$
  - di chuyển từ thành phố thứ 3 đến thành phố kết thúc 4: tổng thời gian di chuyển là $5 + 20 + d_{34}/v = 5 + 20 + 5/0.1 = 5 + 20 + 50 = 75$

- $(\langle 1, 3, 2, 4\rangle, \langle 1, 0, 0, 1, 0\rangle)$: là một lời giải không hợp lệ. Mặc dù chiến lược thu thập với lời giải ơ trên nhưng tổng thời gian di chuyển (83.43) vượt quá ràng buộc thời gian:
  - di chuyển từ thành phố bắt đầu đến thành phố thứ 3 với tốc độ tối đa: thời gian di chuyển là $d_{13}/v_{max} = 6/1.0 = 6$
  - tại thành phố thứ 3, người thu thập lấy vật phẩm 4: vận tốc giảm xuống $v = 1.0 - 1 \times (1.0 - 0.1)/3 = 0.7$
  - di chuyển từ thành phố thứ 3 sang thành phố thứ 2: tổng thời gian di chuyển là $6 + d_{32}/v = 6 + 8/0.7 = 6 + 11.43 = 17.43$
  - tại thành phố thứ 2, vật phẩm thứ 1 được lấy: vận tốc giảm xuống $v = 1.0 - 3 \times (1.0 - 0.1)/3 = 0.1$
  - di chuyển từ thành phố thứ 2 đến thành phố kết thúc 4: tổng thời gian di chuyển là $6 + 17.43 + d_{24}/v = 6 + 17.43 + 6/0.1 = 83.43$

- $(\langle 1, 3, 4\rangle, \langle 0, 0, 1, 0, 0\rangle)$: là một lời giải tối ưu cho trường hợp ví dụ này với tổng lợi nhuận là 100. Tổng khối lượng ba lô là $3 \leq W$ và tổng thời gian di chuyển là $56 \leq T$:

- di chuyển từ thành phố bắt đầu đến thành phố thứ 3 với tốc độ tối đa: thời gian di chuyển là $d_{13}/v_{max} = 6/1.0 = 6$
- tại thành phố thứ 3, người thu thập lấy vật phẩm 3: vận tốc giảm xuống $v = 1.0 - 3 \times (1.0 - 0.1)/3 = 0.1$
- di chuyển từ thành phố thứ 3 đến thành phố kết thúc 4: tổng thời gian di chuyển là $6 + d_{34}/v = 6 + 5/0.1 = 6 + 50 = 56$

Lưu ý rằng chiến lược thu thập của lời giải tối ưu cho trường hợp ví dụ giống như lời giải tối ưu cho bài toán con Ba Lô. Tuy nhiên, không phải lúc nào người thu thập cũng có chiến lược thu thập vật phẩm tốt nhất giống với lời giải tốt ưu cho bài toán con Ba Lô trong giới hạn thời gian T. Để minh họa điều này, hãy xem xét một giới hạn thời gian chặt chẽ hơn là 20 với ví dụ trước đó. Trong trường hợp này, lời giải tối ưu sẽ là $(\langle 1, 3, 4\rangle, \langle 0, 0, 0, 1, 1\rangle)$, có tổng lợi nhuận là 80 và tổng thời gian di chuyển là 18.5.

2.1.2 Biểu diễn toán học

Ở phần này chúng tôi trình bày mô hình bài toán Điều Hướng Thu Thập dưới dạng công thức toán học không tuyến tính. Với bài toán ThOP, chúng ta có một tập hợp gồm n thành phố: 1 là thành phố bắt đầu, n là thành phố kết thúc, và các thành phố còn lại $(2, ..., n-1)$ là các thành phố trung gian. Tại mỗi thành phố trung gian có một hoặc nhiều vật phẩm, và đối với mỗi vật phẩm i, chúng ta có lợi nhuận p_i và trọng lượng w_i của nó. Chúng ta cũng được cho biết giới hạn sức chứa W của ba lô, giới hạn thời gian di chuyển T để đến thành phố kết thúc, tốc độ tối đa và tối thiểu của người thu thập lần lượt là v_{max} và v_{min}, và khoảng cách d_{ij} giữa mọi cặp điểm i và j. Mục tiêu là tìm ra một tuyến đường di chuyển từ thành phố bắt đầu đến thành phố kết thúc, vật phẩm được chọn cẩn thận vào ba lô tại các thành phố trung gian đã ghé thăm, để tối

đa hóa tổng lợi nhuận của ba lô, với ràng buộc không vượt quá khả năng sức chứa của ba lô W và tổng thời gian di chuyển nằm trong giới hạn thời gian T.

Công thức toán học sau bao gồm tất cả các đặc tính của bài toán. Để có thể mô tả rõ ràng, các tác giả cho S_i là tập các vật phẩm được đặt tại thành phố i. Với mỗi $s \subseteq S_i$, w_i^s biểu diễn tổng khối lượng của các vật phẩm trong s và p_i^s là tổng lợi nhuận trong tập con này. Hơn nữa, hằng số $v = (v_{max} - v_{min})/W$ đại diện cho độ mất mát vận tốc cho mỗi đơn vị cân nặng bên trong ba lô, và M' và M'' là hai hằng số đủ lớn. Các biến quyết định bao gồm:

- x_{ij}^s: biến nhị phân nhận giá trị 1 nếu người thu thập đi qua cạnh (i, j) sau khi thu thập các vật phẩm trong tập con $s \subseteq S_i$ và ngược lại nhận giá trị 0.
- q_i: cho biết khối lượng của ba lô sau khi thu thập các vật phẩm tại thành phố i.
- t_i: cho biết thời gian người thu thập tới thành phố i.

$$\max \sum_{i=1}^{n-1} \sum_{j=2}^{n} \sum_{s \subseteq S_i} p_i^s \cdot x_{ij}^s \tag{2.1}$$

$$\sum_{j=2}^{n} x_{1j}^{\varnothing} = 1 \tag{2.2}$$

$$\sum_{i=1}^{n-1} \sum_{s \subseteq S_i} x_{in}^s = 1 \tag{2.3}$$

$$\sum_{i=1}^{n-1} \sum_{s \subseteq S_i} x_{ij}^s - \sum_{i=2}^{n} \sum_{s \subseteq S_j} x_{ji}^s = 0 \quad \forall j = 2..n-1 \tag{2.4}$$

$$q_j \geq q_i + \sum_{j'=2}^{n} \sum_{s \subseteq S_j} w_j^s \cdot x_{jj'}^s - M' \cdot \left(1 - \sum_{s \subseteq S_i} x_{ij}^s\right) \quad \forall i = 1..n, \forall j = 1..n \tag{2.5}$$

$$t_j \geq t_i + \frac{d_{ij}}{v_{\max} - \nu \cdot q_i} - M'' \cdot \left(1 - \sum_{s \subseteq S_i} x_{ij}^s\right) \quad \forall i = 1..n, \forall j = 1..n \tag{2.6}$$

$$x_{ij}^s \in \{0,1\} \quad \forall i = 1..n, \forall j = 1..n \tag{2.7}$$

$$0 \leq q_i \leq W \quad \forall s \subseteq S_i \tag{2.8}$$

$$0 \leq t_i \leq T \quad \forall i = 1..n, \forall i = 1..n \tag{2.9}$$

Mục tiêu (2.1) là tối đa hóa tổng lợi nhuận của các vật phẩm thu thập. Người thu thập phải bắt đầu tại thành phố 1 mang theo một ba lô rỗng (2.2) và đến thành phố n (2.3), ghé thăm bất kỳ thành phố nào trung gian trên đường (2.4). Sau khi ghé thăm một thành phố, người thu thập phải rời đi sau khi thu thập một phần của các vật phẩm của thành phố đó (2.4), điều này làm tăng trọng lượng của balo (2.5) và giảm tốc độ di chuyển tương ứng, ảnh hưởng đến thời gian để đến thành phố tiếp theo (2.6). Trọng lượng của ba lô và tổng thời gian đi bộ phải luôn nằm trong các giới hạn đã cho (2.8)-(2.9).

Lưu ý rằng bộ ràng buộc này là đủ để tránh các chu kỳ con trên đường đi, vì (2.4) đảm bảo luồng đường đi và (2.5)-(2.6) đảm bảo rằng trọng lượng của balo và thời gian của đường đi tăng dần theo chiều dài của đường đi.

Mặc dù đầy đủ, nhưng công thức hiện tại công thức không thể được sử dụng để giải quyết vấn đề do sự phức tạp của nó: số lượng biến là số mũ của số lượng vật phẩm của một thành phố cụ thể bởi vì số lượng tập con có thể xảy ra; và ràng buộc (2.6) là phi tuyến tính, khoảng cách được chia cho một biến liên tục. Các thuật toán để giải quyết bài toán này hầu hết là các thuật toán heuristics.

2.2 Giải thuật tối ưu hóa đàn kiến cho bài toán ThOP

Ở phần này chúng tôi sẽ trình bày một thuật toán cho bài toán ThOP được dựa trên giải thuật tối ưu hóa đàn kiến được gọi là ACO++. Thuật toán này sử dụng một biết thể của giải thuật đàn kiến là MAX-MIN Ant System [22] để xử lý bài toán con OP và một thuật toán heuristic cho bài toán con KP. Thuật toán ACO++ thể hiện kết quả vượt trội so với các thuật toán khác tại thời điểm được công bố và là thuật toán cơ sở để chúng tôi phát triển lên thuật toán SAAS kế thừa các ưu điểm và khắc phục các nhược điểm của ACO++. Chúng tôi sẽ mô tả chi tiết thuật toán ACO++ ở phần 2.2.1 và thuật toán thu thập heuristic ở phần 2.2.2. Ở phần 2.2.3 chúng tôi sẽ trình bày thực nghiệm của chúng tôi về độ nhạy của thuật toán ACO++ với các siêu tham số của nó.

2.2.1 Thuật toán tối ưu hóa đàn kiến

Thuật toán ACO++ của Changas và Wagner là một hướng tiếp cận heuristic cho bài toán ThOP. Nó được dựa trên công trình của Wagner cho bài toán TTP. Thuật toán ACO++ được dựa trên biến thể thuật toán tối ưu hóa đàn kiến MAX-MIN Ant System (MMAS) để giải bài toán con tìm tuyến đường của ThOP, trong khi đó một thuật toán heuristic khác đảm nhận việc thu thập các vật phẩm.

Các thuật toán dựa trên thuật toán tối ưu hóa đàn kiến bao gồm một lớp quan trọng của các kỹ thuật tìm kiếm theo xác suất được lấy cảm hứng từ hành vi của kiến thật. Những thuật toán này đã chứng minh được tính hiệu quả trong việc giải quyết nhiều bài toán tổ hợp [6]. Ý tưởng cơ bản của thuật toán tối ưu hóa đàn kiến là đàn kiến tạo các giải pháp cho một vấn đề cụ thể bằng cách thực hiện các bước đi trên một đồ thị gọi là đồ thị xây dựng. Những bước đi này được ảnh hưởng bởi giá trị pheromone được lưu trữ dọc theo các cạnh của đồ thị. Trong quá trình tối ưu hóa, giá trị pheromone được cập nhật dựa trên các giải pháp tốt được tìm thấy trong quá trình tối ưu hóa, điều

này sau đó sẽ dẫn dắt đàn kiến đến các giải pháp tốt hơn trong các lần lặp tiếp theo của thuật toán.

Biến thể MMAS của giải thuật đàn kiến được thiết kế để cải thiện hiệu suất của thuật toán gốc. Đặc điểm thứ nhất, nó tận dụng mạnh mẽ các đường đi tốt nhất được tìm thấy: chỉ có một trong hai con kiến được dùng để cập nhật pheromone, đó là kiến đã tạo ra đường đi tốt nhất trong vòng lặp hiện tại, hoặc kiến tốt nhất cho tất cả vòng lặp. Thật không may, chiến lược như vậy có thể dẫn đến tình trạng đình trệ trong đó tất cả các kiến đi theo cùng một đường đi, do sự tăng quá mức của đường mùi trên các cung của một đường đi tốt, mặc dù không tối ưu. Để chống lại hiệu ứng này, đặc điểm thứ hai được giới thiệu bởi MMAS là giới hạn khoảng giá trị đường mùi có thể đạt được trong khoảng $[\tau_{min}, \tau_{max}]$. Đặc điểm thứ ba, các đường mùi được khởi tạo với giá trị nồng độ pheromone tối đa cùng với một tỷ lệ bay hơi pheromone nhỏ, làm tăng cường sự khám phá của các đường đi ở giai đoạn đầu quá trình tìm kiếm. Cuối cùng, trong MMAS, nồng độ pheromone trên các đường đi được khởi tạo lại mỗi khi thuật toán có tình trạng đình trệ hoặc khi không có đường đi cải thiện nào được tạo ra trong một số lượng vòng lặp liên tiếp nhất định.

Ở thuật toán ACO++ cho ThOP, để xây dựng các tuyến đường một đàn kiến gồm n_{ants} kiến được đặt ở thành phố xuất phát. Với mỗi bước đàn kiến đi tới một thành phố trong tập các thành phố chưa được thăm (ký hiệu là $V_{unvisited}$) dựa trên xác suất di chuyển được tính bằng công thức 2.10. Xác suất di chuyển được ảnh bởi vết pheromone τ_{ik} và thông tin heuristic η_{ij} (thường được chọn là nghịch đảo của khoảng cách của cạnh nối thành phố i và thành phố j). Độ quan trọng của vết pheromone và thông tin heuristic được quyết định với 2 hệ số α và β. Quá trình xây dựng các tuyến đường dừng lại khi tất cả các con kiến dừng chân tới thành phố kết thúc.

$$p\left(v_i \to v_j\right) = \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{v_k \in V_{\text{unvisited}}} \tau_{ik}^{\alpha} \cdot \eta_{ik}^{\beta}} \text{ for all } v_j \in V_{\text{unvisited}}\,. \tag{2.10}$$

Vết pheromone được trình bày ở công thức 2.11, sẽ phải bay hơi một lượng tương ứng với ρ qua từng lần lặp của thuật toán. Sau đó, lời giả của con kiến có lời giải tốt nhất trong lần lặp hiện tại được dùng để cập nhật pheromone, được ký hiệu là $\Delta\tau_{ij}$.

$$
\begin{aligned}
&\tau_{ij} \leftarrow (1-\rho)\cdot\tau_{ij} + x_{ij}\cdot\delta\cdot\Delta\tau_{ij} \text{ for all } v_i, v_j \in V,\\
&x_{ij} = \begin{cases} 1 \text{ nếu cạnh}_{ij} \text{ nằm trên tuyến đường của con kiến có lời giải tốt nhất,} \\ 0 \text{ ngược lại.} \end{cases}\\
&\delta = 1 \text{ là hệ số cập nhật pheromone.}
\end{aligned}
\tag{2.11}
$$

Trong giải thuật 1, chúng tôi trình bày mã giả đơn giản của ACO++. Ở đầu (dòng 1), lời giải tốt nhất được tìm thấy (tuyến đường đi và chiến lược thu thập) được khởi tạo là một lời giải rỗng. Thuật toán thực hiện chu kỳ lặp của mình (dòng 2 đến 18) đến khi điều kiện dừng chưa được thỏa mãn. Tại dòng 3, mỗi con kiến tạo một đường đi cho người thu thập, sau đó chiến lược thu thập được tạo ra (dòng 4 và 5). Sau đó ACO++ áp dụng một số thuật toán heuristic tìm kiếm cục bộ kinh điển: 2-OPT, 2:5-OPT và 3-OPT [5] để cải thiện chí phí khoảng cách. Nếu bất kỳ tìm kiếm cục bộ nào được kích hoạt trong thuật toán (dòng 6), thủ tục tìm kiếm cục bộ đó sẽ được thực hiện trên mỗi tuyến đường đi π, do đó tạo ra các tuyến đường đi mới π' (dòng 7), có thể tốt hơn π khi so với chi phí khoảng cách. Trong bước tiếp theo, một chiến lược thu thập z' được tạo từ π' (dòng 8). Nếu z' tốt hơn z (dòng 9), pi và z được thay thế bằng π' và z' (dòng 10). Tại dòng 13 đến 15, thuật toán cập nhật lời giải tốt nhất. Lưu ý rằng, để đạt được các đường đi hiệu quả hơn, thuật toán loại bỏ từ tuyến đường đi π tất cả các thành phố mà không có vật phẩm nào được lấy (dòng 14). Sau khi xem xét tất cả các đường đi, vết pheromone được cập nhật dựa trên chất lượng của các lời giải (dòng 17). Cuối cùng, lời giải tốt nhất được tìm thấy được trả về.

Giải thuật 1: Thuật toán ACO++ cho bài toán ThOP

$\pi^{best} \leftarrow \varnothing, z^{best} \leftarrow \varnothing$;
repeat
  $\Pi \leftarrow$ xây dựng các tuyến đường bằng đàn kiến ;
  foreach *chu trình TSP* $\pi \in \Pi$ **do**
    $z \leftarrow$ xây dựng chiến lược thu thập từ π bằng $Pack(\pi, ptries)$;
    if *tìm kiếm địa phương được bật* **then**
      $\pi' \leftarrow$ thực hiện tìm kiếm địa phương trên truyền đường π;
      $z' \leftarrow$ xây dựng chiến lược thu thập từ π' bằng $Pack(\pi', ptries)$;
      if *giá trị lợi nhuận của* z' *lớn hơn giá trị lợi nhuận của* z **then**
        $\pi \leftarrow \pi', z \leftarrow z'$
      end
    end
    if *giá trị lợi nhuận của* z *lớn hơn giá trị lợi nhuận của* z^{best} **then**
      $\pi^{best} \leftarrow \zeta(\pi), z^{best} \leftarrow z$
    end
  end
  cập nhật số liệu của ACO và dấu vết của pheromone;
until *điều kiện dừng được thỏa mãn*;
return π^{best}, z^{best}

$\zeta(\pi)$ bỏ đi các thành phố không có vật phẩm được thu tập từ chiến lược thu thập z từ π.

2.2.2 Thuật toán thu thập heuristic

Trong giải thuật 2, chúng tôi trình bày thuật toán thu thập heuristic, của Changas và Wagner đề xuất [4], để xây dựng một chiến lược thu thập từ một tuyến đường đi cố định. Lưu ý rằng ngay cả khi đường đi của người thu thập được giữ cố định, việc tìm một chiến lược thu thập tối ưu là bài toán NP-khó.

Thuật toán thu thập heuristic này cố gắng tìm một chiến lược thu thập tốt từ nhiều thử nghiệm cho cùng một tuyến đường đi π. Số lượng thử nghiệm được xác định bởi $ptries$. Mỗi lần thử nghiệm được mô tả từ dòng 2 đến 24. Ở đầu mỗi lần thử nghiệm (dòng 3), thuật toán chọn đồng đều ba giá trị ngẫu nhiên (θ, δ, và γ) giữa 0 và 1, sau đó chuẩn hóa chúng sao cho tổng của chúng bằng 1. Các giá trị này được sử dụng để tính điểm cho mỗi vật phẩm $i \in \{1, ..., m\}$ (dòng 4 đến 5), trong đó θ, δ, và γ xác định, tương ứng, các số mũ được áp dụng cho lợi nhuận p_i, trọng lượng w_i, và khoảng cách d_i để quản lý tác động của chúng. Khoảng cách d_i được tính toán theo tuyến đường đi π bằng cách tính tổng tất cả các khoảng cách từ thành phố chứa vật phẩm i đến thành phố cuối cùng. Công thức 2.12 cho thấy cách điểm của vật phẩm i được tính.

$$s_i = \frac{p_i^\theta}{w_i^\delta \cdot d_i^\gamma}. \tag{2.12}$$

Lưu ý rằng mỗi điểm số s_i tích hợp một sự cân đối giữa khoảng cách mà vật phẩm i phải được mang qua, trọng lượng của nó và lợi nhuận của nó. Công thức 2.12 dựa trên thuật toán heuristic PackIterative đã được phát triển cho TTP [8]. Tuy nhiên, khác với [8] thuật toán xem xét một số mũ cho tham số của khoảng cách để xét tầm quan trọng của ảnh hưởng của nó. Hơn nữa, các giá trị của tất cả các số mũ đều được chọn ngẫu nhiên từ 0 đến 1 cho mỗi lần thử nghiệm (và sau đó được chuẩn hóa) để tìm kiếm không gian cho các chiến lược thu thập tham lam.

Sau khi tính toán điểm cho tất cả các vật phẩm, thuật toán sử dụng giá trị của chúng

Giải thuật 2: Thuật toán thu thập: $Pack(\pi, ptries)$

```
1  z ← ∅, try ← 1;
2  repeat
3      chọn một số thực cho mỗi tham số θ, δ, và γ từ một phân phối chuẩn
         trong khoảng [0, 1], để θ + δ + γ = 1;
4      foreach i ← 1 to m do
5          tính giá trị điểm số s_i cho vật phẩm i;
6      end
7      z ← ∅ ;
8      for j ← 1 to m do
9          i ← chọn vật phẩm có điểm số cao thứ j ;
10         z′ ← z′ ∪ {i} ;
11         if trọng lượng của z′ lớn hơn W then
12             z′ ← z′\{i} ;
13         else
14             t ← tính thời gian cần thiết để thu thập theo chiến lược z′ bằng
                 cách đi qua các thành phố trong hành trình π ;
15             if nếu thời gian t lâu hơn ràng buộc T then
16                 z′ ← z′\{i} ;
17             end
18         end
19     end
20     if giá trị lợi nhuận của z′ lớn hơn giá trị lợi nhuận của z then
21         z ← z′;
22     end
23     try ← try + 1;
24 until try > ptries;
25 return z
```

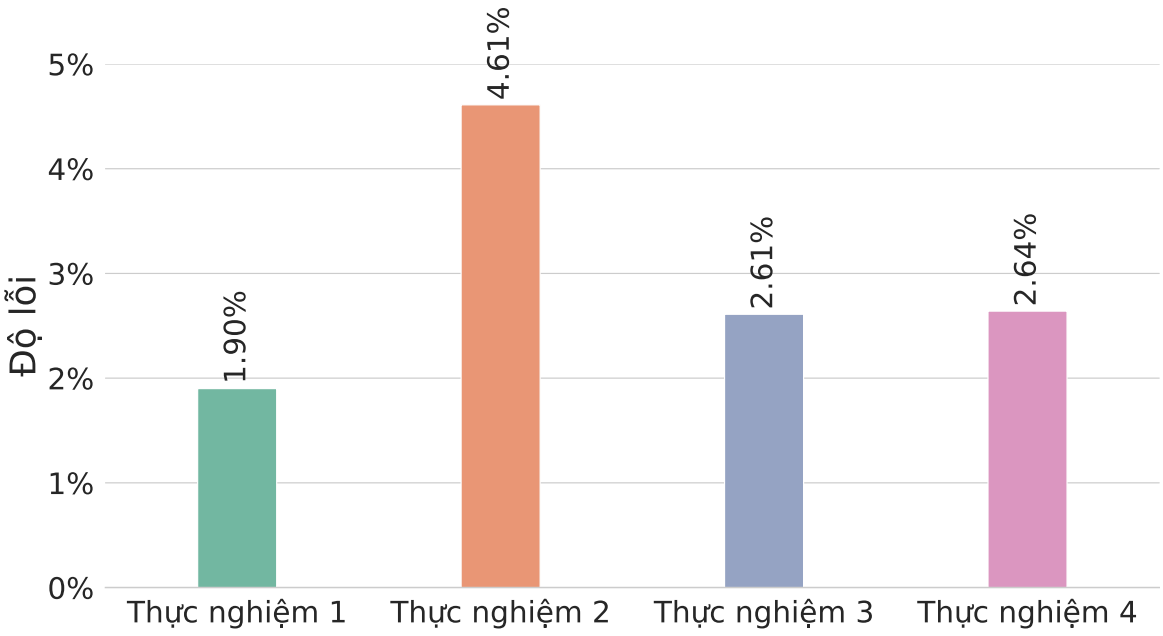
để xác định ưu tiên của mỗi vật phẩm trong chiến lược thu thập. Điểm số của một vật phẩm càng lớn, ưu tiên của nó càng cao. Giữa các dòng 8 và 19, chúng tôi tạo chiến lược thu thập cho lần thử nghiệm hiện tại bằng cách xem xét các vật phẩm theo ưu tiên của chúng. Nếu một vật phẩm vi phạm ràng buộc của bài toán ThOP (dòng 11 và 15), nó sẽ không được chọn. Lưu ý rằng thuật toán tính thời gian di chuyển (dòng 14) từ các thành phố được liệt kê trên đường đi π, nhưng thuật toán bỏ qua những thành phố nơi không có vật phẩm nào được chọn. Sau khi hoàn thành chiến lược thu thập của lần thử nghiệm hiện tại, tổng lợi nhuận của nó được so sánh với chiến lược thu thập tốt nhất cho đến nay (dòng 20), sau đó cập nhật chiến lược thu thập tốt nhất cho đến nay (dòng 21). Ở cuối tất cả các lần thử nghiệm, chiến lược thu thập tốt nhất được tìm thấy được trả về (25).

2.2.3 Đánh giá độ nhạy với siêu tham số của thuật toán ACO++

Trong phần này, chúng tôi tiến hành một nghiên cứu thực nghiệm để thể hiện mối quan hệ giữa cấu hình tham số và hiệu suất tổng thể của thuật toán ACO++. Nghiên cứu của chúng tôi bao gồm bốn thí nghiệm khác nhau, mỗi thí nghiệm sử dụng các bộ tham số đa dạng, thực hiện trên hai nhóm trường hợp từ bộ thang đo ThOP: `a280_01_unc` và `dsj1000_01_unc`. Mô tả chi tiết về các trường hợp trong bộ thang đo này được trình bày trong phần 4.2. Các thí nghiệm được tổ chức như sau:

- Thí nghiệm 1: Chúng tôi cho thuật toán ACO++ chạy bằng cách sử dụng các bộ tham số được điều chỉnh kỹ lưỡng cho các nhóm trường hợp được đề cập trước đó, các bộ tham số này được trình bày trong bài báo ACO++.
- Thí nghiệm 2: Chúng tôi hoán đổi các bộ tham số giữa hai nhóm trường hợp.

- Thí nghiệm 3: Chúng tôi điều chỉnh giá trị tham số một cách nhẹ bằng cách tăng α, β và ρ khoảng 3%, và tăng số lần thử nghiệm thuật thu thập heuristic thêm một đơn vị.
- Thí nghiệm 4: Chúng tôi tính giá trị trung bình của mỗi tham số trong tập hợp 48 cấu hình tham số ACO++ được thiết lập cho ThOP và thực hiện nó trên hai nhóm trường hợp này để thử nghiệm cách một bộ tham số ảnh hưởng đến hiệu suất ACO++.



Hình 2.2: Minh họa tỷ lệ lỗi trung bình từ mỗi thí nghiệm so với kết quả tốt nhất giữa bốn thí nghiệm cho mỗi nhóm trường hợp. Chúng tôi thực hiện 30 lần chạy độc lập cho mỗi trường hợp.

Kết quả của những thí nghiệm này được trình bày trong Hình 2.2. Các thí nghiệm của chúng tôi cho thấy sự biến động đáng kể qua các bộ siêu tham số khác nhau cho thuật toán ACO++. Đặc biệt là sự chênh lệch đáng chú ý được quan sát trong Thí nghiệm 2, nơi việc hoán đổi các bộ tham số dẫn đến sự tăng đáng kể 4,61% trong tỷ lệ lỗi - hơn gấp đôi con số của Thí nghiệm 1, nơi sử dụng cấu hình tham số cụ thể

cho từng trường hợp, mang lại tỷ lệ lỗi chỉ là 1,90%. Ngược lại, Thí nghiệm 3 và 4 thể hiện tỷ lệ lỗi tương đương khoảng 2,60%. Các thí nghiệm của chúng tôi ở đây cho thấy sự nhạy cảm của ACO++ đối với các siêu tham số của nó.

2.3 Cơ chế thích ứng tốc độ bay hơi pheromone

Các thuật toán heuristics thường rất nhạy cảm đối với cài đặt tham số và do đó, chúng thiếu tính ổn định để giải quyết mọi cấu hình vấn đề một cách tổng quát. Vấn đề này thường được xử lý bằng cách sử dụng mỗi bộ tham số riêng cho từng trường hợp. Điều này rất tốn thời gian khi ta xem xét đến tính thực tế của thuật toán.

Từ quan điểm này, thì có vẻ như tham số quan trọng nhất đối với các thuật toán dựa trên tối ưu hóa đàn kiến là hệ số bay hơi mùi ρ. Hệ số này kiểm soát tốc độ bay hơi của pheromone, ảnh hưởng quyết định đến quá trình hội tụ của thuật toán. Giá trị càng lớn, quá trình hội tụ đến một điểm tối ưu cục bộ càng nhanh. Việc chọn giá trị bay hơi ρ phù hợp phụ thuộc vào bản đồ mà trường hợp thuật toán đang giải.

Công trình về phương pháp AACO-NC [21], một thuật toán cho TSP, đã đề xuất một phương pháp thích ứng tham số tốc độ bay hơi pheromone ρ cho thuật toán dựa trên tối ưu hóa đàn kiến. Ý tưởng của phương pháp này là tham số ρ sẽ không cố định xuyên suốt quá trình thuật toán xử lý. Tham số ρ sẽ thay đổi trong quá trình tối ưu hóa dựa trên thông tin đa dạng của các lời giải sinh ra bởi đàn kiến, thông tin này gọi là entropy. Tại điểm bắt đầu tối ưu hóa, khi mà độ đa dạng của lời giải đàn kiến lớn (giá trị entropy lớn) bởi vì các lời giải hoàn toàn ngẫu nhiên, thì giá trị của tốc độ bay hơi ρ phải lớn để đảm bảo thuật toán hội tụ nhanh. Tuy nhiên, trong quá trình tối ưu hóa, khi mà các lời giải của đàn kiến bắt đầu hội tụ vào một điểm tối ưu cục bộ nào đó, giá trị tốc độ bay hơi ρ nên giảm xuống để mở rộng không gian tìm kiếm.

Giá trị độ đa dạng của các lời giải trong đàn kiến được tính dựa trên công thức Shannon entropy H được trình bày bởi công thức dưới đây.

$$H = -\sum_{i=2}^{n}\sum_{j=1}^{i-1} p_{ij} \cdot \log_2 p_{ij} \tag{2.13}$$

Trong đó p_{ij} là xác suất mà cạnh E_{ij} nối 2 đỉnh v_i và v_j ($i \neq j$) xuất hiện trong bất kỳ lời giải của đàn kiến. Giá trị xác suất này được tính bằng số lần xuất hiện của cạnh trong các lời giải theo công thức sau

$$p_{ij} = \frac{\#\,\textit{số lần xuất hiện}(E_{ij})}{n \cdot n_{ants}} \tag{2.14}$$

với *số lần xuất hiện*(E_{ij}) là số lần cạnh E_{ij} nối 2 đỉnh v_i và v_j xuất hiện trong các lời giải của đàn kiến.

Giá trị giới hạn cực tiểu và cực đại của entropy cho các lời giải của đàn kiến được tính theo hai công thức 2.15 và 2.16.

$$H_{min} = \log_2 \frac{1}{n} \tag{2.15}$$

$$H_{min} = \log_2 \frac{1}{n \cdot n_{ants}} \tag{2.16}$$

với n là số lượng đỉnh trong đồ thị (hay số lượng thành phố trong bản đồ); n_{ants} là số lượng lời giải sinh ra bởi đàn kiến.

Tham số tốc độ bay hơi ρ thay đổi trong khoảng giới hạn được gán tại thời điểm bắt đầu thuật toán:

- ρ_{min}: cận dưới của tham số tốc độ bay hơi ($0 < \rho_{min} \leq \rho_{max}$)
- ρ_{min}: cận trên của tham số tốc độ bay hơi ($\rho_{min} \leq \rho_{max} < 1$)

Giá trị của tham số tốc độ bay hơi pheromone ρ ở mỗi thế hệ được tính theo công thức 2.17. Công thức này thể hiện tham số ρ phụ thuộc tuyến tính vào giá trị entropy giới hạn.

$$\rho = \rho_{\min} + (\rho_{\max} - \rho_{\min}) \cdot \frac{H - H_{\min}}{H_{\max} - H_{\min}}. \tag{2.17}$$

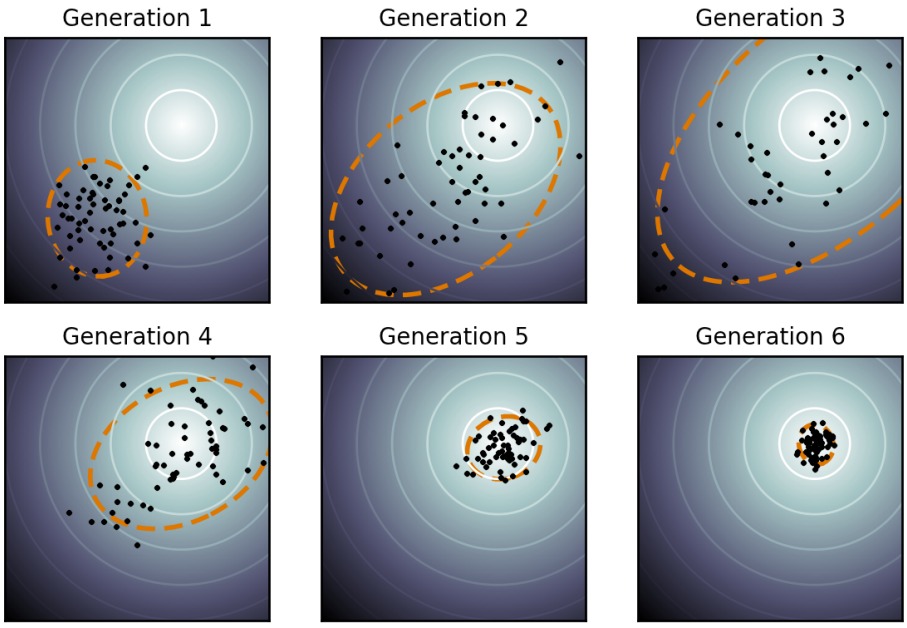
Điểm mạnh của nguyên tắc này là tham số tốc độ bay hơi phản ánh sự tiến triển hiện tại trong quá trình tối ưu hóa. Khi quần thể có xu hướng hội tụ đến một tối ưu cục bộ, tham số cũng được giảm, do đó, pheromone bay hơi chậm hơn với ảnh hưởng của sự hội tụ chậm hơn. Điều này mở rộng không gian tìm kiếm dẫn đến khả năng khám phá một tối ưu cục bộ tốt hơn (hoặc có thể là toàn cục).

2.4 Thuật toán tiến hóa CMA-ES

Thuật toán Covariance Matrix Adaptation Evolution Strategy (CMA-ES) là một phương pháp tối ưu hóa tiến hóa được thiết kế để giải quyết các bài toán tối ưu hóa không đạo hàm. Được đề xuất bởi Nikolaus Hansen vào năm 2003 [11], CMA-ES đã trở thành một trong những thuật toán quan trọng trong lĩnh vực tối ưu hóa tiến hóa.

CMA-ES kết hợp giữa hai yếu tố chính là ma trận hiệp phương sai (covariance matrix) và chiến lược tiến hóa (evolution strategy). Trong quá trình tối ưu hóa, thuật toán không chỉ tìm kiếm giải pháp tốt mà còn điều chỉnh ma trận hiệp phương sai để thích ứng với cấu trúc hình dạng của không gian tìm kiếm.

Một đặc điểm nổi bật của CMA-ES là khả năng xử lý các bài toán tối ưu hóa không đạo hàm và không yêu cầu thông tin đạo hàm của hàm mục tiêu. Điều này làm cho nó trở thành lựa chọn phù hợp trong nhiều ứng dụng thực tế, khi mà việc tính toán đạo hàm có thể khó khăn hoặc tốn kém. Thuật toán thực hiện quá trình tối ưu hóa thông qua việc tạo ra và cập nhật một quần thể các cá thể (solutions) theo thời gian. Các cá thể này được tổ chức thành một phân phối xác suất, và chiến lược tiến hóa được sử dụng để điều chỉnh phân phối này sao cho các cá thể có khả năng sinh sản tốt nhất được ưu tiên. Với sự linh hoạt và hiệu suất của mình, CMA-ES đã được áp dụng rộng



Hình 2.3: Minh họa quá trình CMA-ES tối ưu hóa một bài toán 2 chiều đơn giản. Địa hình tối ưu hóa hình cầu được biểu diễn dưới các đường liền. Quần thể được biểu diễn dưới các chấm và phân phối của CMA-ES được biểu diễn bằng đường đứt nét thay đổi qua các thế hệ. Trong bài toán đơn giản này, quần thể tập trung vào cực trị toàn cục chỉ trong vài thế hệ.

rãi trong nhiều lĩnh vực như tối ưu hóa tham số, máy học, và các ứng dụng trong khoa học dữ liệu.

Trong giải thuật 3, chúng tôi trình bày mã giả tóm tắt đầy đủ cho thuật toán CMA-ES. Giải thuật 3 bắt đầu bằng cách khởi tạo các tham số ở dòng 1 là 2. Các tham số của thuật toán bao gồm vị trí trung tâm của phân phối m, ma trận hiệp phương sai C, và bước nhảy σ. Ba tham số này m, C, σ sẽ được tuần tự cập nhật từ dòng 3 đến dòng 11 của thuật toán. Dòng 4 là lấy mẫu một quần thể từ phân phối chuẩn với vị trí trung tâm m và hiệp phương sai C. Dòng 5 đánh giá các cá thể trong quần thể bằng một hàm hộp đen $f(.)$. Sau đó thuật toán sẽ dùng thông tin đánh giá này để cập nhật vị trí trung tâm m bằng cách lấy tổng μ cá thể tốt nhất ở dòng 6, các trọng số được tính bằng log(μ/2) - log(i). Ký hiệu $y_{i:\lambda}$ nghĩa là các cá thể tốt nhất từ y_i, ..., y_λ. Ma trận hiệp

phương sai được cập nhật ở dòng 10 bao gồm 3 phần: (1) thông tin cũ, (2) cập nhật bậc 1 (được tính bằng sự thay đổi của vị trí trung tâm hay còn gọi là đường tiến hóa p_c), và (1) cập nhật bậc μ được tính bằng các các thể tốt ở quần thể vừa rồi. Tham số bước nhảy được cập nhật ở dòng 8 dựa trên đường tiến hóa liên hợp p_σ. Mục tiêu của nó là tăng tốc tốc độ hội tụ vào cực trị, trong khi ngăn chặn việc hội tụ sớm. Các tham số khác bao gồm: μ_w là giá trị chọn hiệu quả phương sai, các tham số tốc độ học c_1, c_c, c_σ, và d_σ là giá trị làm mượt cho σ. Các thông số của các tham số này được nghiên cứu kỹ lưỡng và thảo luận sâu trong [12]. Điều kiện để kết thúc có thể được cài đặt nhiều cách khác nhau tùy thuộc vào không gian tìm kiếm, hoặc hàm thích nghi.

Giải thuật 3: Thuật toán CMA-ES

1 khởi tạo $\mathbf{m} \in \Re^n, \sigma \in \Re^+, \lambda, \mu$;
2 khởi tạo $\mathbf{C} = \mathbf{I}, \mathbf{p}_c = \mathbf{0}, \mathbf{p}_\sigma = \mathbf{0}$;
3 **repeat**
4 lấy mẫu: $\theta_i = \mathbf{m} + \sigma \mathbf{y}_i, \mathbf{y}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{C}), i = 1, \ldots, \lambda$;
5 đánh giá: $f(\theta_i), i = 1, \ldots, \lambda$;
`// cập nhật giá trị trung bình`
6 $\mathbf{m} \leftarrow \mathbf{m} + \sigma \overline{\mathbf{y}}$, với $\overline{\mathbf{y}} = \sum_1^\mu w_i \mathbf{y}_{i:\lambda}$;
`// cập nhật bước nhảy`
7 $\mathbf{p}_\sigma \leftarrow (1 - c_\sigma)\, \mathbf{p}_\sigma + \sqrt{c_\sigma (2 - c_\sigma)\, \mu_w}\, \mathbf{C}^{-\frac{1}{2}} \overline{\mathbf{y}}$;
8 $\sigma \leftarrow \sigma \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\mathbf{p}_\sigma\|}{\mathbb{E}\|\mathcal{N}(\mathbf{0},\mathbf{I})\|} - 1\right)\right)$;
`// cập nhật ma trận hiệp phương sai`
9 $\mathbf{p}_c \leftarrow (1 - c_c)\, \mathbf{p}_c + \sqrt{c_c (2 - c_c)\, \mu_w}\, \overline{\mathbf{y}}$;
10 $\mathbf{C} \leftarrow \left(1 - c_1 - c_\mu\right) \mathbf{C} + c_1 \mathbf{p}_c \mathbf{p}_c^\top + c_\mu \sum_1^\mu w_i \mathbf{y}_{i:\lambda} \mathbf{y}_{i:\lambda}^\top$;
11 **until** *điều kiện dừng được thỏa mãn*;

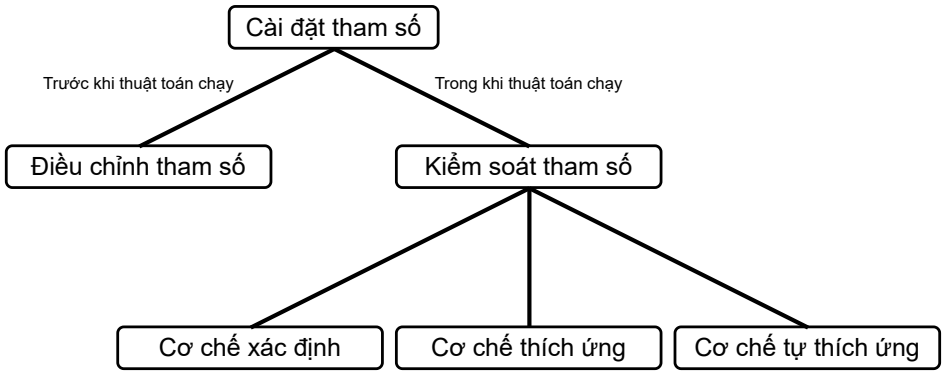
2.5 Kỹ thuật cài đặt tham số

Với các thuật toán heurisitic như thuật toán tối ưu hóa đàn kiến và thuật toán tiến hóa thì hiệu suất của các thuật toán này phụ thuộc phần lớn vào các tham số của chúng. Ở khóa luận này chúng tôi đặt trọng tâm vào việc tự động thích ứng các tham số của thuật toán ACO++, một thuật toán heuristic. Để dễ dàng nắm bắt được những kỹ thuật mà chúng tôi trình bày, ở phần này chúng tôi trình bày kiến thức cơ bản về các kỹ thuật cài đặt siêu tham số khác nhau cho một thuật toán heuristic.

Chúng tôi xem rằng có hai loại cài đặt tham số chính là điều chỉnh siêu tham số (parameter tuning) và kiểm soát siêu tham số (parameter control) [7]. Với điều chỉnh siêu tham số, là phương pháp cài đặt tham số thường xuyên được sử dụng, thì quá trình tìm ra các giá trị tham số tốt được thực hiện trước khi thuật toán chạy và sau đó dùng những tham số được tìm thấy cho thuật toán và không thay đổi trong quá trình chạy. Phương pháp điều chỉnh siêu tham số là hướng tiếp cận điển hình khi thiết kế thuật toán. Quá trình điều chỉnh được thực hiện bằng cách thử nghiệm nhiều giá trị tham số và chọn thủ công ra bộ tham số có kết quả tốt nhất. Tuy nhiên, với phương pháp này thì số lượng tham số và khoảng giá trị của chúng có thể dẫn đến quá trình điều chỉnh siêu tham số này rất tốn thời gian. Và phương pháp này có thể dẫn đến hiệu suất tổng quát thuật toán không tốt nếu các tham số này chỉ được thử nghiệm trên một tập trường hợp nhất định.

Phương pháp kiểm soát tham số là phương pháp đối nghịch với phương pháp điều chỉnh tham số trên. Các tham số sẽ được gán các giá trị khởi tạo khi bắt đầu thuật toán và thay đổi trong quá trình thuật toán chạy. Phương pháp này có thể giải quyết được vấn đề hiệu suất tổng quát của thuật toán bởi vì các tham số sẽ thay đổi giá trị theo từng trường hợp trong quá trình chạy của thuật toán. Hơn nữa với phương pháp này, nếu ta thiết kế nó đủ thông minh, nó có thể không cần tới quá trình điều chỉnh tham số thủ công tốn thời gian mà vẫn tìm ra tham số tốt cho thuật toán. Ở kỹ thuật kiểm

soát tham số này, các tác giả của công trình [7] đã phân loại ra ba cơ chế con dựa trên cách hoạt động của chúng bao gồm cơ chế xác định, cơ chế thích ứng, cơ chế tự thích ứng. Hình 2.4 Minh họa phân loại các kỹ thuật cài đặt tham số.



Hình 2.4: Minh họa phân loại các kỹ thuật cài đặt tham số

Cơ chế kiểm soát tham số xác định là cơ chế mà khi giá trị của một tham số được thay đổi bởi một quy tắc xác định. Quy tắc này điều chỉnh tham số theo một cách cố định, quy tắc xác định được định nghĩa sẵn (tức là được định nghĩa thủ công) mà không sử dụng bất kỳ phản hồi nào từ quá trình tìm kiếm. Thông thường, thì cơ chế này giống như cơ chế lập lịch mà tham số biến đổi theo thời gian.

Cơ chế kiểm soát tham số thích ứng là cơ chế mà khi có một hình thức phản hồi từ quá trình tìm kiếm được sử dụng làm đầu vào cho một một thủ tục để điều chỉnh tham số. Việc gán giá trị cho tham số có thể dựa trên chất lượng của các lời giải được tìm ra bởi thuật toán. Ví dụ cho cơ chế này là cơ chế thích ứng tốc độ bay hơi pheromone chúng tôi đã trình bày ở phần 2.3, tham số tốc độ bay hơi của thuật toán được điều chỉnh dựa trên chất lượng của các lời giải mà thuật toán tìm được ở vòng lặp hiện tại.

Cuối cùng là cơ chế kiểm kiểm soát tự thích ứng, ví dụ điển hình cho cơ chế này là thuật toán CMA-ES mà chúng tôi đã trình bày ở phần 3.3. Ở đây, các tham số cần được điều chỉnh được mã hóa vào các cá thể và trải qua quá trình đột biến và tái kết hợp. Các giá trị tốt hơn của các tham số được mã hóa này dẫn đến các cá thể tốt hơn,

từ đó có khả năng cao hơn để sống sót, tạo ra con cái và do đó lan truyền những giá trị tham số tốt hơn này. Điều này là sự phân biệt quan trọng giữa các cơ chế thích ứng và cơ chế tự thích ứng: trong trường hợp tự thích ứng, các cơ chế cho việc gán điểm và cập nhật các tham số chiến lược khác nhau là hoàn toàn ngụ ý, tức là chúng là các toán tử lựa chọn và biến đổi của chu kỳ tiến hóa chính nó.

2.6 Phương pháp phân cụm thứ bậc

Phân cụm thứ bậc là một kỹ thuật phân cụm nhằm xây dựng một cấu trúc cây của các cụm hay cây cụm thứ bậc. Cây cụm thứ bậc mang cấu trúc dữ liệu cây với mỗi nút là một cụm. Trong đó, tập điểm dữ liệu của một nút con là một phần trong tập điểm dữ liệu của nút ba mẹ. Phân cụm thứ bậc có thể giúp biểu hiện cấu trúc phân cấp tự nhiên có trong các điểm dữ liệu.

Phân cụm thứ bậc có hai loại là phân cụm phân chia (divisive clustering) và phân cụm hội tụ (agglomerative clustering) [20]. Phân chia là một phương pháp từ trên xuống bắt đầu với một cụm bao gồm tất cả các điểm dữ liệu. Nó chia các cụm thành các cụm con một cách tuần tự cho đến khi bản thân cụm là một điểm dữ liệu. Ngược lại, phân cụm hội tụ là một phương pháp từ dưới lên bắt đầu với mỗi điểm dữ liệu như một cụm đơn và sau đó hợp nhất các cụm gần nhau để tạo ra các cụm lớn hơn cho đến khi chỉ còn lại một cụm chứa tất cả các điểm dữ liệu. Cách phân chia hoặc hợp nhất các cụm phụ thuộc vào ứng dụng và đặc tính của dữ liệu. Do đó, phân cụm thứ bậc có nhiều biến thể và độ phức tạp không gian, thời gian khác nhau [14, 15].

Chương 3

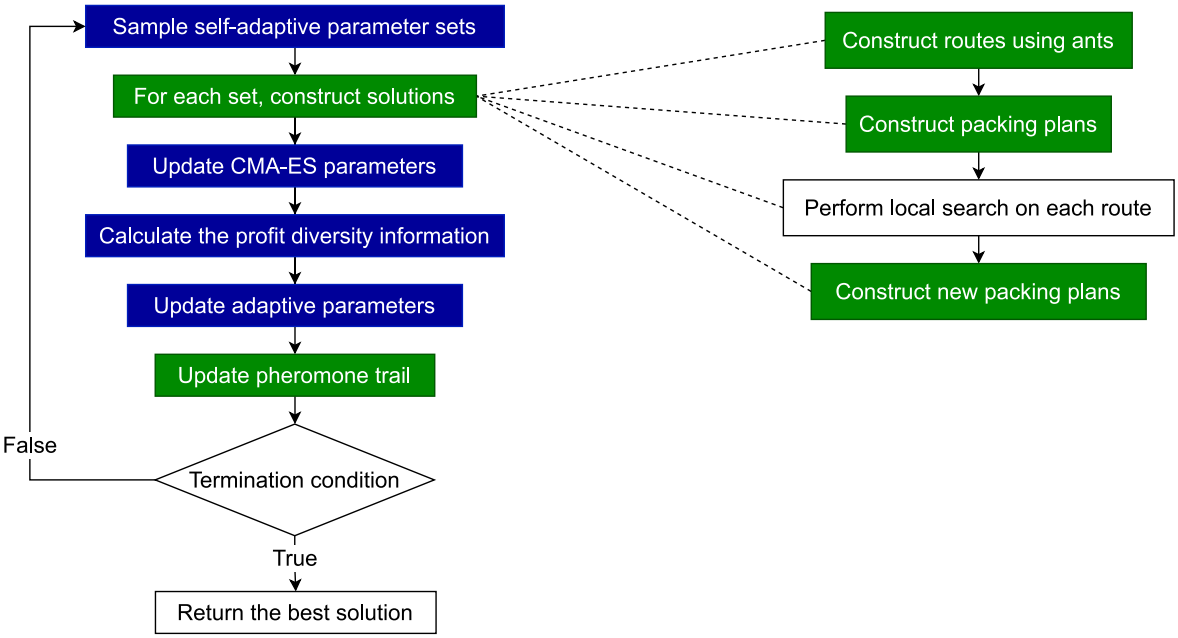
THUẬT TOÁN ĐỀ XUẤT

Trong chương này, chúng tôi sẽ trình bày chi tiết về thuật toán mà chúng tôi đã đề xuất để giải quyết bài toán điều hướng thu thập (ThOP). Mục tiêu của thuật toán là đạt được hiệu suất cao và đồng thời có khả năng tự động thích ứng với các tham số mà không làm tăng độ phức tạp của thuật toán.

3.1 Tổng quan thuật toán

Thuật toán của chúng tôi được phát triển dựa trên nền tảng của thuật toán ACO++ đã được giới thiệu trước đó. Như chúng tôi đã mô tả trong phần kiến thức nền tảng, ACO++ là một biến thể của giải thuật đàn kiến MAX-MIN Ant System (MMAS). Tuy nhiên, một trong nhược điểm của ACO++ là yêu cầu sự đặt trước các siêu tham số trước khi thực hiện thuật toán. Điều này tạo ra một thách thức do ảnh hưởng lớn của các siêu tham số đối với kết quả của thuật toán.

Để khắc phục hạn chế này, chúng tôi đã thực hiện hai phương pháp. Đầu tiên, chúng tôi đã cải tiến thuật toán để tự động thích ứng các tham số trong quá trình hoạt động. Thứ hai, chúng tôi đã áp dụng hai kỹ thuật nhằm giảm độ phức tạp của thuật toán, từ đó nâng cao khả năng tìm kiếm lời giải tốt cho bài toán ThOP. Hình 3.1 cho thấy bức tranh toàn cảnh về các cải tiến của SAAS so với ACO++. Trong đó, màu xanh dương thể hiện thành phần mới, màu xanh lá thể hiện sự điều chỉnh thành phần



Hình 3.1: Các cải tiến của SAAS so với ACO++. Trong đó, màu xanh dương thể hiện thành phần **mới**, màu xanh lá thể hiện sự **điều chỉnh** thành phần cũ và màu trắng thể hiện sự giữ nguyên thành phần cũ.

cũ và màu trắng thể hiện sự giữ nguyên thành phần cũ. Mã giả của thuật toán được trình bày chi tiết trong Giải Thuật 4.

Khi bắt đầu, thuật toán khởi tạo lời giải tốt nhất bằng một lời giải trống với một tuyến đường và một chiến lược thu thập (dòng 1). Các tham số của cơ chế tự thích ứng của CMA-ES cũng được khởi tạo (dòng 2). Đồng thời các cây cụm thứ bậc cho từng thành phố cũng được khởi tạo (dòng 3). Theo sau đó thuật toán vào vòng lặp chính (dòng 4), và vòng lặp này tiếp tục lặp lại cho đến khi điều kiện kết thúc được thỏa mãn (dòng 30). Điều kiện để kết thúc vòng lặp là khi thời gian thuật toán tìm kiếm vượt quá thời gian giới hạn của trường hợp đang xét.

Tại vòng lặp chính (dòng 4 đến dòng 30), đầu tiên thuật toán lấy mẫu một quần thể từ cơ chế tự thích ứng của CMA-ES (dòng 5). Với mỗi cá thể trong quần thể vừa được lấy mẫu (từ dòng 6 đến dòng 22), một tập con của đàn kiến tìm kiếm một tập các lời giải bằng cách sử dụng các tham số được lấy từ thông tin của mỗi cá thể (dòng 7).

Giải thuật 4: Thuật toán SAAS cho bài toán ThOP

$\pi^{best} \leftarrow \varnothing, z^{best} \leftarrow \varnothing$;
khởi tạo CMA-ES;
xây dựng cây cụm thứ bậc;
repeat
  pop ← lấy mẫu một quần thể từ CMA-ES;
  foreach *indv* ∈ *pop* **do**
    $\alpha, \beta, \theta, \delta, \gamma, \rho_{\min}, \rho_{\max} \leftarrow$ indv.genome;
    cập nhật ρ bằng phương trình 2.17;
    for $i := 1$ ***to*** n_{indv} **do**
      $\pi \leftarrow$ xây dựng tuyến đường bằng một con kiến;
      $\pi' \leftarrow$ thực hiện tìm kiếm địa phương trên tuyến đường π;
      $z \leftarrow$ xây dựng chiến lược thu thập cho π;
      $z' \leftarrow$ xây dựng chiến lược thu thập cho π';
      if *giá trị lợi nhuận của* z' *cao hơn giá trị lợi nhuận của* z **then**
        $\pi \leftarrow \pi', z \leftarrow z'$;
      end
      if *giá trị lợi nhuận của* z *cao hơn giá trị lợi nhuận của* z^{best} **then**
        $\pi^{best} \leftarrow \zeta(\pi), z^{best} \leftarrow z$;
      end
    end
    indv.fitness ← giá trị lợi nhuận cao nhất trong các lời giải của quần thể;
  end
  cập nhật tham số của CMA-ES ;
  if *điều kiện khởi động lại CMA-ES được thỏa mãn* **then**
    khởi động lại CMA-ES;
  end
  tính giá trị độ đa dạng lợi nhuận bằng công thức 3.1 và 3.2;
  cập nhật n_{indv} bằng công thức 3.4;
  cập nhật dấu vết của pheromone và số liệu của MMAS;
until *Điều kiện dừng được thỏa mãn*;
return π^{best}, z^{best};

$\zeta(\pi)$ bỏ đi các thành phố không có vật phẩm được thu tập từ chiến lược thu thập z từ π.

Với mỗi con kiến tập con đó (dòng 9), nó sẽ khám phá ra một tuyến đường π dựa trên thông tin dấu vết pheromone (dòng 10. Sau đó thuật toán tạo ra một tuyến đường khác π' từ tuyền đường vừa tìm được với con kiến với độ dài được thử để giảm quãng đường bằng phương pháp tìm kiếm địa phương 2-OPT[5]. Tiếp theo, thuật toán sử dụng phương pháp thu thập vật phẩm để sinh ra chiến lược thu tập cho hai tuyền đường đó (dòng 12 và 13). Nếu lợi nhuận của chiến lược thu thập của tuyến đường đường cải thiện π' tốt hơn tuyến đường gốc π' thì thay thế tuyến đường gốc và chiến lược thu thập bằng tuyến đường được cải thiện và chiến lược thu thập của nó (dòng 14). Sau đó cập nhật lời giải tốt nhất bằng cách so sánh với lời giải hiện tại (dòng 17).

Phương pháp thu thập vật phẩm của thuật toán chúng tôi giống với phương pháp được sử dụng trong thuật toán ACO++[3]. Tuy nhiên chúng tôi đã chỉnh sửa các giá trị tham số của θ, δ, và γ không cần phải sinh bằng hàm ngẫu nhiên đồng nhất mà lấy từ thông tin của mỗi cá thể. Vậy nên thuật toán chúng tôi chỉ gọi hàm sinh ra chiến lược thu thập vật phẩm này một lần thay vì nhiều lần như thuật toán ACO++. Bởi vì CMA-ES của thuật toán chúng tôi có khả năng tự học để cải thiện các tham số này qua các cá thể của quần thể, trong khi trong thuật toán ACO++ cần gọi nhiều lần để tăng khả năng hàm sinh giá trị ngẫu nhiên sinh ra các giá trị tham số θ, δ, và γ tốt cho phương pháp thu thập tham lam.

Giá trị sinh tồn của mỗi cá thể (giá trị indv.fitness) được quyết định bằng giá trị lợi nhuận (giá trị hàm mục tiêu) cao nhất của tập đàn kiến con sử dụng thông tin di truyền của nó để sinh ra các lời giải (dòng 21). Khi tất cả các cá thể trong quần thể của CMA-ES đã tính xong giá trị sinh tồn của chúng, cơ chế tự thích ứng CMA-ES sử dụng các thông tin này để cập nhật học và cập nhật các tham số của nó để ở vòng lặp tiếp theo cơ chế sẽ sinh ra quần thể có các cá thể tốt hơn (dòng 23). Nếu các tham số của CMA-ES chạm điều kiện khởi động lại thì cơ chế tự thích ứng CMA-ES sẽ khởi tạo lại các tham số của nó (dòng 25). Chi tiết về cách cơ chế tự thích ứng CMA-ES

hoạt động được trình bày ở phần 3.3.

Thuật toán sau đó tính giá trị đa dạng lợi nhuận của đàn kiến (Line 27) của cơ chế thích ứng để điều chỉnh số lượng kiến ở trong mỗi đàn kiến con và giá trị bay hơi của pheromone (dòng 8). Chi tiết về cơ chế thích ứng này được trình bày ở phần 3.2. Ở cuối mỗi vòng lặp chính, thuật toán cập nhật dấu vết pheromone and các số liệu của phương pháp MMAS (dòng 29).

Lưu ý rằng chúng tôi đã thay thế cơ chế bay hơi của MMAS bằng cơ chế bay hơi lười biến ở những lúc thuật toán sử dụng thông tin dấu vết bay hơi (dòng 9 và dòng 29). Cơ chế bay hơi lười biến được trình bày chi tiết ở phần 3.5. Ngoài ra khi mỗi con kiến tìm tuyến đường di chuyển (dòng 9), đàn kiến trong giải thuật của chúng tôi di chuyển trên các cạnh của cây cụm thứ bậc, thay vì di chuyển trên các cạnh nối giữa các thành phố. Các thông tin dấu vết pheromone cũng được lưu và cập nhật trên các cạnh của cây. Mô tả và cách hoạt động của Kiến di chuyển trên cây cụm thứ bậc được trình bày ở phần 3.4

Thuật toán sẽ tiếp tục lặp lại vòng lặp chính để cải thiện lời giải tốt nhất và giá trị của các tham số cho đên khi điều kiện dừng được thỏa mãn. Sau khi kết thúc vòng lặp chính thuật toán của chúng tôi trả về lời giải hiệu quả bao gồm tuyến đường cùng với chiến lược thu thập vật phẩm.

3.2 Tận dụng thông tin sự đa dạng lợi nhuận cho cơ chế thích ứng

Cơ chế kiểm soát tham số đầu tiên chúng tôi cài đặt trong SAAS là cơ chế dựa trên giá trị lợi nhuận của các lời giải được tìm bởi đàn kiến. Cơ chế này được chúng tôi dựa vào cơ chế thích nghi của thuật toán AACO-NC[21], được trình bày ở phần 2.3. Cơ chế của chúng tôi khác với của thuật toán AACO-NC ở hai điểm. Thứ nhất, cơ chế

của chúng tôi thích ứng không chỉ tham số tốc độ bay hơi pheromone ρ mà còn cho lượng kiến cho mỗi cá thể n_{indv} của quần thể CMA-ES. Cũng giống như tham số ρ, số lượng kiến cho mỗi cá thể n_{indv} đóng vai trò quan trọng việc cân bằng giữa khám phá và khai phá không gian tìm kiếm của thuật toán.

Điểm khác biệt thứ hai là cơ chế của chúng tôi sử dụng giá trị lợi nhuận của các lời giải được tìm bởi đàn kiến ở vòng lặp hiện tại để thích ứng hai tham số trên. Khác với thuật toán AACO-NC giải quyết bài toán TSP, độ đa dạng của đàn kiến thể hiện qua sự khác nhau giữa các cạnh trong chu trình của lời giải, bài toán ThOP bao gồm hai cần tuyến đường và chiến lược thu thập nên chúng tôi chọn giá trị lợi nhuận của mỗi lời giải để tính độ đa dạng bởi vì nó đại diện cho cả hai thành phần này. Bởi vì sự khác biệt này nên chúng tôi đã chỉnh sửa lại các công thức của thuật toán AACO-NC.

Ở cuối mỗi vòng lặp chính, thuật toán của chúng tôi sẽ tính thông tin sự đa dạng lợi nhuận entropy H bằng các lời giải trong đàn kiến bằng công thức Shannon entropy dưới đây:

$$\begin{aligned} &S = \{P \mid P \text{ là giá trị lợi nhuận của một lời giải được tìm thấy bởi đàn kiến hiện tại}\}, \\ &H = -\sum_{i=1}^{|S|} p_i \cdot \log_2 p_i \end{aligned} \tag{3.1}$$

Trong đó p_i là xác suất của giá trị P_i, giá trị lợi nhuận thứ i trong tập giá trị lợi nhuận S được tìm thấy bởi đàn kiến. Xác suất này là tần suất xuất hiện của giá trị lợi nhuận P_i trong tập lời giải được tìm bởi đàn kiến:

$$p_i = \frac{\#\text{ số lần xuất hiện}(P_i)}{n_{\text{ants}}} \tag{3.2}$$

Trong đó n_{ants} là số con kiến trong đàn. Phạm vi giá trị entropy được giới hạn bởi $H_{\min}$ và $H_{\max}$:

$$H_{\min} = -\log_2 \frac{n_{\text{ants}}}{n_{\text{ants}}} = 0, \quad H_{\max} = -\log_2 \frac{1}{n_{\text{ants}}}. \tag{3.3}$$

Tỷ lệ bay hơi pheromone ρ được điều chỉnh theo công thức 2.17 của thuật toán AACO-NC được trình bày ở phần 2.3. Tương tự, với số lượng kiến mỗi cho cá thể n_{indv} chúng tôi sử dụng một phiên bản sửa đổi của công thức 2.17:

$$n_{\text{indv}} = n_{\text{indv\_max}} - (n_{\text{indv\_max}} - n_{\text{indv\_min}}) \cdot \frac{H - H_{\min}}{H_{\max} - H_{\min}}. \tag{3.4}$$

Ở đây, số lượng kiến được gán cho mỗi cá thể, ký hiệu là n_{indv}, là cố định cho tất cả các cá thể. $n_{\text{indv\_max}}$ và $n_{\text{indv\_min}}$ là là tham số đại diện cho giới hạn lớn nhất và nhỏ nhất của số lượng kiến cho mỗi cá thể. Khác với tham số tốc độ bay hơi ρ, giá trị của n_{indv} tăng khi sự đa dạng lợi nhuận thấp để khuyến khích khám phá và giảm khi đa dạng cao để tăng cường khai phá không gian tìm kiếm. Sự tích hợp thông tin sự đa dạng lợi nhuận cải thiện chiến lược thích ứng, tạo ra sự cân bằng giữa khám phá và khai thác.

3.3 Cơ chế tự thích ứng sử dụng CMA-ES

Thuật toán ACO++ được tác giả điều chỉnh các siêu tham số riêng cho từng nhóm trường hợp để cải thiện tính ổn định của hiệu suất thuật toán. Chúng tôi đề xuất tích hợp một cơ chế tự thích ứng để điều chỉnh các tham số trong quá trình tìm kiếm lời giải của thuật toán. Một số tham số có thể tự thích ứng được liệt kê trong bảng 3.1. Cơ chế này cho phép các tham số thích nghi với đặc điểm riêng biệt của trường hợp và quá trình tìm kiếm.

Tham số	Cơ chế kiểm soát tham số	Khoảng giá trị
α	Tự thích ứng	[0, 1]
β	Tự thích ứng	[0, 1]
$\rho_{\min}$, $\rho_{\max}$	Tự thích ứng	[0, 1]
θ	Tự thích ứng	[0, 1]
δ	Tự thích ứng	[0, 1]
γ	Tự thích ứng	[0, 1]
n_{indv}	Thích ứng	$[n_{\text{indv\_max}}, n_{\text{indv\_min}}]$
ρ	Thích ứng	$[\rho_{\min}, \rho_{\max}]$

Bảng 3.1: Danh sách các tham số được kiểm soát bằng hai kỹ thuật kiểm soát tham số tự thích ứng và thích ứng

Tham số α và β là hai tham số kiểm soát tính quan trọng của nồng độ pheromone và thông tin heuristic trong công thức tính xác suất di chuyển (công thức 2.11). Tham số $\rho_{\min}$ và $\rho_{\max}$ là các tham số thể hiện cận trên là cận dưới cho tham số tốc độ bay hơi ρ, giúp cho cơ chế kiểm soát tham số thích ứng mà chúng tôi đã trình bày ở phần 3.2. Các tham số θ, δ, và γ, là các tham số điều chỉnh mức độ ảnh hưởng của các giá trị lợi nhuận, trọng lượng và khoảng cách trong công thức tính điểm của vật phẩm trong phương pháp thu thập heuristic (Công thức 2.12). Thuật toán CMA-ES sẽ kiểm soát các giá trị của tham số này qua một phân mối chuẩn đa biến. Phân phối chuẩn này được khởi tạo qua các giá trị trung tậm mặc định và giá trị bước nhảy mặc định cho từng tham số.

Trong phương pháp SAAS, chúng tôi sử dụng thuật toán CMA-ES để làm cơ chế tự thích ứng. Ở đầu mỗi lần lặp chính (Dòng 4 đến Dòng 30), một quần thể các bộ tham số được lấy mẫu từ CMA-ES. Sau đó, đối với mỗi cá thể trong quần thể này, một nhóm kiến con sẽ sử dụng các tham số đã được mã hóa trên cá thể đó để xây dựng các lời giải. Giá trị sinh tồn của mỗi cá thể được xác định bằng giá trị lợi nhuận cao nhất

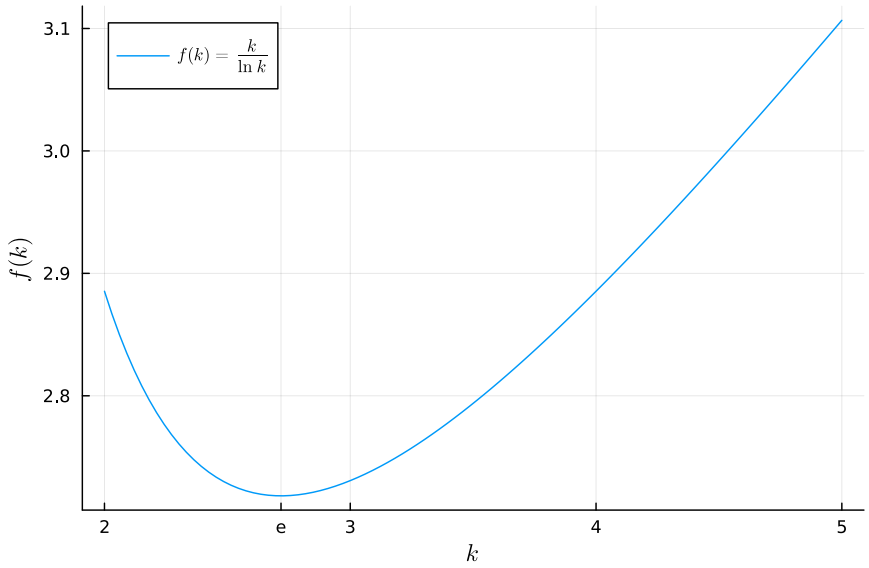
đạt được trong nhóm kiến con tương ứng của cá thể đó. Sau khi giá trị sinh tồn cho tất cả các cá thể đã được tính, thuật toán CMA-ES sẽ điều chỉnh phân phối của nó để cái thiện khả năng sinh ra các cá thể tốt hơn ở dựa theo các bộ tham số có khả năng đạt được giá trị lợi nhuận cao hơn.

Để tránh thuật toán bị kẹt trong cực trị địa phương hoặc quá trình tìm kiếm không hiệu quả, chúng tôi đã cài đặt cơ chế khởi động lại cho CMA-ES, cơ chế mà sẽ gán lại phân phối của CMA-ES. Cơ chế này được kích hoạt khi mà giá trị bước nhảy trở nên quá nhỏ (đã hội tụ vào một cực trị địa phương) hoặc trở nên quá lớn vượt quá giới hạn của không gian tìm kiếm (quá trình tìm kiếm không hiệu quả).

3.4 Kiến di chuyển trên cây cụm thứ bậc

Với giải thuật đàn kiến thông thường, khi kiến cân nhắc di chuyển đến thành phố tiếp theo, chúng ta cần xem xét n thành phố và tính xác suất dịch chuyển. Cách thức này có độ phức tạp về thời gian là $O(n)$. Nhận thấy kĩ thuật từ AACO-NC [21] giúp giảm chi ở quá trình này. Cụ thể như sau, nếu ta cluster n thành phố thành k cụm, khi chọn thành phố tiếp theo, ta sẽ chọn cụm trước, rồi mới cân nhắc n_{size} thành phố trong cụm đó. Kĩ thuật này đã giúp giảm độ phức tạp xuống còn $O(k + n_{\text{size}})$.

Khi được biết qua kĩ thuật trên, chúng tôi nhận thấy rằng nếu ta có thể giảm chi phí chọn thành phố bằng cách phân cụm chúng, vậy thì, tương tự, ta cũng có thể giảm chi phí chọn cụm bằng cách phân cụm chúng lần nữa và nhiều lần nữa. Chúng tôi quyết định xây dựng cây cụm thứ bậc, nơi mà các nút con là các cụm được phân cụm từ nút ba mẹ. Nút gốc đại diện cho một cụm chứa tất cả thành phố, các nút lá đại diện cho các cụm chứa đúng một thành phố. Bằng cách duyệt cây từ gốc đến lá, ở mỗi nút, kiến sẽ cân nhắc k cụm (nút) con, khi ấy chi phí chọn thành phố tiếp theo sẽ là $\Theta(k \cdot \log_k n)$.



Hình 3.2: Ảnh hưởng của k đến chi phí chọn thành phố tiếp theo.

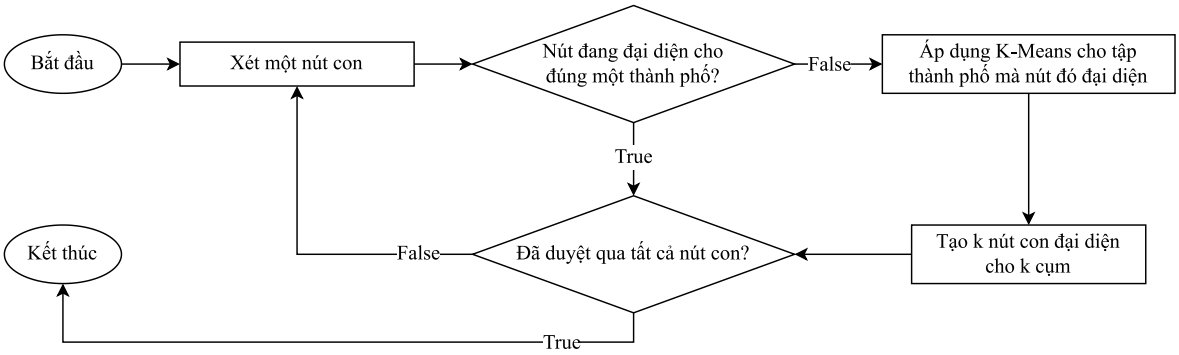
Để tìm ra giá trị k mang lại chi phí thấp nhất, chúng tôi gọi

$$\begin{aligned} f(k) &= k \cdot \log_k n \\ &= \ln n \cdot \frac{k}{\ln k} \end{aligned}$$

và tiến hành tìm cực trị của hàm f:

$$\begin{aligned} &\text{Xét } f'(k) = 0 \\ \Leftrightarrow &\ln n \cdot \frac{\ln k - 1}{\ln^2 k} = 0 \\ \Rightarrow &\ln k - 1 = 0 \\ \Leftrightarrow &k = \mathrm{e}. \end{aligned}$$

Nhìn hình 3.2, ta có thể thấy 2 và 3 là hai giá trị nguyên gần với e nhất và $f(3) < f(2)$. Do đó chọn k bằng 3 sẽ mang lại chi phí tốt nhất. Tuy vậy, trong quá trình thực nghiệm, chúng tôi đã chọn k bằng 2. Khi giá trị k được cố định, độ phức tạp về thời gian của



Hình 3.3: Phân cụm thứ bậc phân chia (divisive hierarchical clustering) với K-Means làm phương pháp tách.

việc chọn thành phố tiếp theo sẽ là $\Theta(\log n)$.

Chúng tôi xác định sáu đặc trưng cho mỗi thành phố để tiến hành phân cụm. Các đặc trưng bao gồm toạ độ x, tọa độ y của thành phố, khoảng tin cậy của trọng lượng và lợi nhuận của các vật phẩm thuộc thành phố. Thông tin về trọng lượng và lợi nhuận sẽ có khoảng tin cậy riêng và mỗi khoảng tin cậy tương ứng với 2 đặc trưng. Cụ thể, đầu tiên, ta tính giá trị trung bình mean và độ lệch chuẩn std từ các vật phẩm trong thành phố. Sau đó, ta chọn biên dưới $\text{mean} - 2 \cdot \text{std}$ và biên trên $\text{mean} + 2 \cdot \text{std}$ của khoảng tin cậy để làm đặc trưng. Việc thêm thông tin vật phẩm vào đặc trưng giúp ta phân cụm các thành phố có cùng phân bố vật phẩm. Từ đó giúp đàn kiến khám phá các thành phố lân cận hiệu quả hơn.

Thành phố bắt đầu là nơi chỉ có thể rời khỏi, không thể đi tới, do đó, nó không cần thêm vào cây cụm thứ bậc. Mặt khác, thành phố kết thúc, nơi mà không có vật phẩm nào, nghĩa là nó không thể có đặc trưng cho vật phẩm, do đó nó cần được phân cụm theo cách riêng. Cụ thể, ở lớp đầu tiên ngay dưới nút gốc, ta thủ công phân cụm thành phố kết thúc thành một cụm riêng và các thành phố còn lại thành một cụm. Đối với các thành phố khác, chúng tôi sử dụng phương pháp phân cụm thứ bậc phân chia (divisive hierarchical clustering) với K-Means làm phương pháp tách (Hình 3.3) để xây dựng cấu trúc cây.

Thuật toán phân cụm này là một cách hiệu quả để xây dựng cấu trúc cây cụm thứ bậc. Độ phức tạp về thời gian của phân cụm K-Means là $O(n \cdot t \cdot k \cdot d)$, trong đó n là số lượng thành phố (hay số điểm dữ liệu) cần phân cụm, t là số vòng lặp trong quá trình phân cụm, k thể hiện số cụm và d thể hiện số đặc trưng của thành phố (hay số chiều của điểm dữ liệu). Giữa lớp của cây, mặc dù có số lượng nút khác nhau, và mỗi nút đại diện cho một lượng thành phố khác nhau, nhưng trong một lớp, tổng số thành phố không thể vượt quá n. Do đó, việc áp dụng K-Means tại một lớp của cậy để tạo ra một lớp mới sẽ tốn tổng chi phí là $O(n \cdot t \cdot k \cdot d)$. Vì mỗi nút có k nút con, nên cây có trung bình $\log_k n$ lớp, chi phí của thuật toán sẽ là $\Theta(\log_k n \cdot n \cdot t \cdot k \cdot d)$. Do k và d sẽ được mang giá trị cố định và không có liên hệ đến input, độ phức tạp về thời gian của việc xây dựng cấu trúc cây là $\Theta(n \cdot \log n \cdot t)$.

Trong phương pháp của chúng tôi, kiến sẽ di chuyển qua cây cụm thứ bậc thay vì di chuyển trực tiếp từ thành phố này sang thành phố khác. Mỗi thành phố có cây cụm thứ bậc riêng thể hiện các cạnh đi đến n thành phố và có chung một cấu trúc. Việc có cấu trúc cây giống hệt nhau ở mọi thành phố cho phép thuật toán đánh dấu hiệu quả thành phố nào đã được được đi qua.

Ngược lại với AACO-NC, chúng tôi sử dụng pheromone và heuristic riêng biệt cho các cạnh của cây để tính toán xác suất chuyển đổi. Heuristic của một cạnh trên cây được định nghĩa là khoảng cách từ thành phố hiện tại đến tọa độ trọng tâm của cụm đích.

Trong các cây không cân bằng, lá ở độ sâu nông hơn thường có xác suất được chọn cao hơn. Để giải quyết vấn đề này, chúng tôi điều chỉnh trọng số chuyển đổi đến một nút con bằng cách nhân nó với số lượng lá dưới nút con đó. Điều này có nghĩa là trọng số chuyển đổi sang một nút, lúc này, mang tính đại diện cho tổng trọng số chuyển đổi đến các lá của nó.

3.5 Bay hơi lười biếng

Ở các giải thuật đàn kiến khác, việc bay hơi pheromone thường được thực hiện bằng cách duyệt qua từng cạnh trên bản đồ và nhân nồng độ pheromone của cạnh với $(1-\rho)$, mang độ phức tạp thời gian $O(n^2)$. Cách này tương đối lãng phí bởi vì phần lớn các cạnh ít khi được kiến đi qua, ta điều chỉnh dấu vết pheromone mà chưa thực sự dùng đến. Để khắc phục nhược điểm trên, chúng tôi đề xuất kỹ thuật Bay hơi lười biếng giúp thuật toán đạt được hiệu quả bay hơi tương tự với độ phức tạp thấp hơn. Bay hơi lười biếng giảm độ phức tạp bằng cách trì hoãn cập nhật các vết pheromone chưa dùng đến.

Nguyên tắc cốt lõi của bay hơi lười biếng là theo dõi trạng thái mong muốn và trạng thái lịch sử. Trạng thái mong muốn bao gồm hai biến DECount và DRCount. DECount theo dõi số lần bay hơi (mong muốn) kể từ lần cuối (mong muốn) khởi tạo lại pheromone. DRCount theo dõi số lần (mong muốn) khởi tạo lại pheromone kể từ lúc thuật toán bắt đầu. Ứng với mỗi cạnh là một trạng thái lịch sử riêng, bao gồm $\tau_{\text{historical}}$, HECount và HRCount. $\tau_{\text{historical}}$ là nồng độ pheromone lịch sử của cạnh tương ứng tại thời điểm HECount và HRCount. HRCount ghi nhận số lần $\tau_{\text{historical}}$ đã khởi tạo lại. HECount ghi nhận số lần $\tau_{\text{historical}}$ đã bay hơi kể từ lần khởi tạo lại pheromone thứ HRCount.

Bằng cách so sánh trạng thái lịch sử và trạng thái mong muốn, chúng ta có thể xác định nồng độ pheromone mong muốn τ_{desired} cho các cạnh cần dùng đến. Nếu HRCount nhỏ hơn DRCount, có nghĩa là cạnh chưa được cập nhật kể từ lần khởi tạo lại pheromone trước đó, vì vậy thuật toán sẽ đặt lại $\tau_{\text{historical}}$ bằng pheromone khởi tạo lại. Sau đó, $\tau_{\text{historical}}$ của cạnh được bay hơi (DECount - HECount) lần:

$$\tau_{\text{desired}} \leftarrow \tau_{\text{historical}} \cdot (1-\rho)^{\text{DECount}-\text{HECount}}. \tag{3.5}$$

Bay hơi lười biếng trì hoãn điều chỉnh nồng độ pheromone khi giải thuật đàn kiến có mong muốn thực hiện bay hơi hoặc khởi tạo lại, thay vào đó, ta cập nhật trạng thái mong muốn. Trước khi tính xác suất chuyển tiếp, Bay hơi lười biếng sẽ khôi phục τ_{desired} bằng cách quan sát trạng thái mong muốn và trạng thái lịch sử. Khi cần rải thêm pheromone, ta đồng bộ trạng thái lịch sử với trạng thái mong muốn và cộng thêm lượng pheromone mới (Công thức 3.6). Phương thức này giúp giảm độ phức tạp của quá trình bay hơi và khởi tạo lại pheromone xuống tương đương với chi phí chọn đường đi và rải thêm pheromone.

$$\tau_{\text{historical}} \leftarrow \tau_{\text{desired}} + \Delta\tau. \tag{3.6}$$

Tương tự, ở đầu giai đoạn tìm đường, ta cũng có thể dùng kỹ thuật lười biếng này để giảm chi phí khởi tạo lại trạng thái chưa đi qua của các thành phố.

Chương 4

THỰC NGHIỆM

4.1 Điều chỉnh siêu tham số

Các siêu tham số của SAAS bao gồm các giá trị tối thiểu và tối đa cho mỗi tham số thích ứng, và các giá trị trung bình ban đầu và kích thước bước cho mỗi tham số tự thích ứng. Khác với Chagas và Wagner [3], các siêu tham số của chúng tôi được điều chỉnh bằng thuật toán CMA-ES. Chúng tôi ngẫu nhiên chọn 21 trường hợp cho mỗi thế hệ và đánh giá các bộ siêu tham số trên chúng. Chúng tôi so sánh giá trị mục tiêu được đánh giá với giá trị mục tiêu của ACO++ trên cùng trường hợp và tính toán phần trăm cải thiện so với ACO++. Cuối cùng, chúng tôi lấy trung bình phần trăm cải thiện trên 21 trường hợp để làm fitness cho CMA-ES. Chúng tôi cập nhật quần thể bằng cách giữ lại hai cá thể tốt nhất từ hai thế hệ trước. Sau khi điều chỉnh, chúng tôi đánh giá hai cá thể tốt nhất từ hai thế hệ cuối cùng trên tất cả các trường hợp. Chúng tôi chọn cá thể tốt hơn làm bộ siêu tham số cuối cùng cho thuật toán.

Chúng tôi sử dụng các siêu tham số giống như Chagas và Wagner cho BRKGA, ACO và ACO++ [3], ngoại trừ ILS vì nó không có siêu tham số. Các siêu tham số này đã được tìm thấy thông qua quá trình điều chỉnh. Chagas và Wagner đã gom nhóm các trường hợp có cùng định dạng `XXX_YY_ZZZ` để phục vụ cho quá trình điều chỉnh siêu tham số. Tổng cộng, benchmark có 48 nhóm. Đối với mỗi nhóm trường hợp và mỗi thuật toán, Chagas và Wagner đã thực hiện 5.000 thí nghiệm [3], dẫn đến 240.000 thí

nghiệm cho mỗi thuật toán. Ngược lại, chúng tôi điều chỉnh SAAS chỉ với 45.000 thí nghiệm. Lưu ý rằng một thí nghiệm tương đương với việc thực thi thuật toán cho một trường hợp trong benchmark.

4.2 Bộ benchmark được sử dụng

ThOP benchmark [19] bao gồm 432 trường hợp, mỗi trường hợp có các đặc điểm riêng biệt về số lượng thành phố, số lượng vật phẩm, kích thước ba lô, thời gian di chuyển tối đa và sự tương quan giữa trọng lượng và lợi nhuận của các vật phẩm. Các trường hợp này được lấy từ TTP benchmark [18] bằng cách loại bỏ các vật phẩm liên quan đến các thành phố bắt đầu và kết thúc và thêm thời gian di chuyển tối đa. Các đặc điểm của các trường hợp:

- Số lượng thành phố: 51, 107, 280 hoặc 1000 (từ các trường hợp TSP: *eil51*, *pr107*, *a280*, *dsj1000*).
- Số lượng vật phẩm trên mỗi thành phố: *01*, *03*, *05* hoặc *10*.
- Sự tương quan giữa trọng lượng và lợi nhuận: Không tương quan (uncorrelated, *unc*), không tương quan với trọng lượng tương tự (uncorrelated with similar weights, *usw*) hoặc bị giới hạn và tương quan mạnh (bounded and strongly correlated, *bsc*).
- Kích thước ba lô: Kích thước ba lô có thể là *01*, *05* hoặc *10* lần kích thước ba lô nhỏ nhất.
- Thời gian di chuyển tối đa: Được xác định bởi các lớp *01*, *02* hoặc *03*, tương ứng với 50%, 75% và 100% thời gian tham chiếu cho mỗi trường hợp trong bài báo gốc của ThOP [19].

Mỗi trường hợp được biểu thị ở định dạng `XXX_YY_ZZZ_WW_TT`, trong đó `XXX` biểu thị loại trường hợp TSP, `YY` biểu thị số lượng vật phẩm trên mỗi thành phố, `ZZZ` biểu thị loại ba lô, `WW` biểu thị kích thước ba lô và `TT` biểu thị thời gian di chuyển tối đa. Ví dụ: `pr107_05_bsc_01_01` biểu thị: trường hợp có 107 thành phố pr107 của TSP benchmark; 5 vật phẩm trên mỗi thành phố có trọng lượng và lợi nhuận phụ thuộc lẫn nhau, bị giới hạn; kích thước ba lô và giới hạn thời gian nhỏ nhất như đã nêu.

4.3 Thiết lập thực nghiệm

Chúng tôi so sánh chất lượng của các giải pháp thu được bởi phương pháp đề xuất của chúng tôi SAAS và bởi các phương pháp trước đây. Các phương pháp trước đây được dùng trong thực nghiệm bao gồm ILS [19], BRKGA [19], ACO [4], và ACO++ [3] nhưng không bao gồm GA [9] do chúng tôi không thể tiếp cận mã nguồn của nó. Để có sự so sánh công bằng, chúng tôi đã chạy thực nghiệm cho tất cả các phương pháp trên cùng một máy tính với Intel(R) Core(TM) i7-8750H @ 2.20GHz.

Vì tất cả các thuật toán đều có yếu tố ngẫu nhiên, chúng tôi thực hiện 30 lần chạy độc lập trên mỗi trường hợp trong benchmark. Đối với SAAS, chúng tôi sử dụng bộ tham số tốt nhất được tìm thấy bởi quá trình điều chỉnh của chúng tôi cho tất cả các trường hợp trong ThOP benchmark. Đối với BRKGA, ACO và ACO++, chúng tôi sử dụng các bộ siêu tham số tốt nhất tương ứng với các nhóm trường hợp được tìm thấy trong bài báo ACO++ [3]. Xin lưu ý rằng chúng tôi giới hạn thời gian thực thi của mỗi trường hợp trong ThOP benchmark là $\lceil 0.1m \rceil$ giây với m là số lượng vật phẩm trong trường hợp tương ứng [19].

Bên cạnh so sánh giá trị mục tiêu của lời giải, chúng tôi có tiến hành kiểm định Wilcoxon signed-rank cho 2 phương pháp tốt nhất để kiểm tra độ tin cậy của so sánh.

4.4 Kết quả thực nghiệm

4.4.1 Kết quả thực nghiệm so sánh lời giải của các thuật toán

Đầu tiên chúng tôi tiến hành thực nghiệm so sánh kết quả lời giải của các thuật toán bằng cách so sánh tỉ lệ giá trị mục tiêu ở mỗi trường hợp với lời giải tốt nhất được tìm thấy. Ở mỗi thuật toán, chúng tôi tính giá trị tỉ lệ này bằng việc tính trung bình giá trị mục tiêu của 30 lần chạy độc lập cho một trường hợp. Sau đó tính tỉ lệ so với giá trị mục tiêu của lời giải tốt nhất được tìm thấy cho trường hợp đó qua thực nghiệm của chúng tôi. Giá trị tỉ lệ này càng cao thì hiệu suất của thuật toán hiện tại càng tốt trong trường hợp đó. Kết quả của thực nghiệm này được thể hiện qua Hình 4.1.

Qua kết quả thực nghiệm cho thấy được rằng ILS và BRKGA thể hiện hiệu suất thấp hơn hẳn so với 3 thuật toán dựa trên giải thuật tối ưu hóa đàn kiến ACO, ACO++, SAAS với 2 biểu đồ nhiệt có bảng màu nhạt hơn. Ta có thể thấy rằng ILS và BRKGA thể hiện tốt với cái trường hợp có kích thước nhỏ ở số lượng thành phố hoặc số lượng vật phẩm ở mối thành phố. Điều này thể hiện qua biểu đồ nhiệt của 2 thuật toán này đậm hơn ở 2 viền: viền trái và viền dưới của biểu đồ.

Các thuật toán dựa trên giải thuật đàn kiến (ACO, ACO++, SAAS) thể hiện hiệu suất ổn định trên toàn bộ thang đo với các biểu đồ nhiệt có bảng màu đậm ở hầu hết các trường hợp. ACO++ và SAAS là 2 thuật có biểu đồ nhiệt đậm nhất trong 5 thuật toán chúng tôi đã thử nghiệm. ACO++ thể hiện rất tốt với biểu đồ nhiệt đậm và có nhiều ký hiệu hình thoi (tìm ra được nhiều lời giải tốt nhất), tuy nhiên thuật toán SAAS của chúng tôi có biểu đồ nhiều đậm hơn và dày đặc các ký hiệu hình thoi hơn (tìm ra được nhiều lời giải tốt nhất hơn). SAAS thành công tìm ra được lời giải tốt nhất cho 330 trường hợp trên 432 trường hợp của benchmark trong khi đó ACO++ chỉ tìm được lời giải tốt nhất trên 180 trường hợp.

Với biểu đồ nhiệt của thuật toán SAAS của chúng tôi, ta có thể thấy các ký hiệu

hình thoi dày đặt ở phía bên trái biểu đồ và thưa dần về bên phải bắt đầu từ các trường hợp có tiền tố `a280_01` trở đi. Điều này xảy ra là bởi vì thuật toán của chúng tôi không có đủ thời gian để tìm ra lời giải tốt nhất. Các trường hợp ở bên phải các trường hợp có tiền tố `a280_01` là các trường hợp của số lượng thành phố lớn (280 và 1000 thành phố) nên để tìm được lời giải tốt thuật toán của chúng tôi cần nhiều thời gian hơn bởi vì việc tìm kiếm các giá trị tham số thực thi trong quá trình chạy.

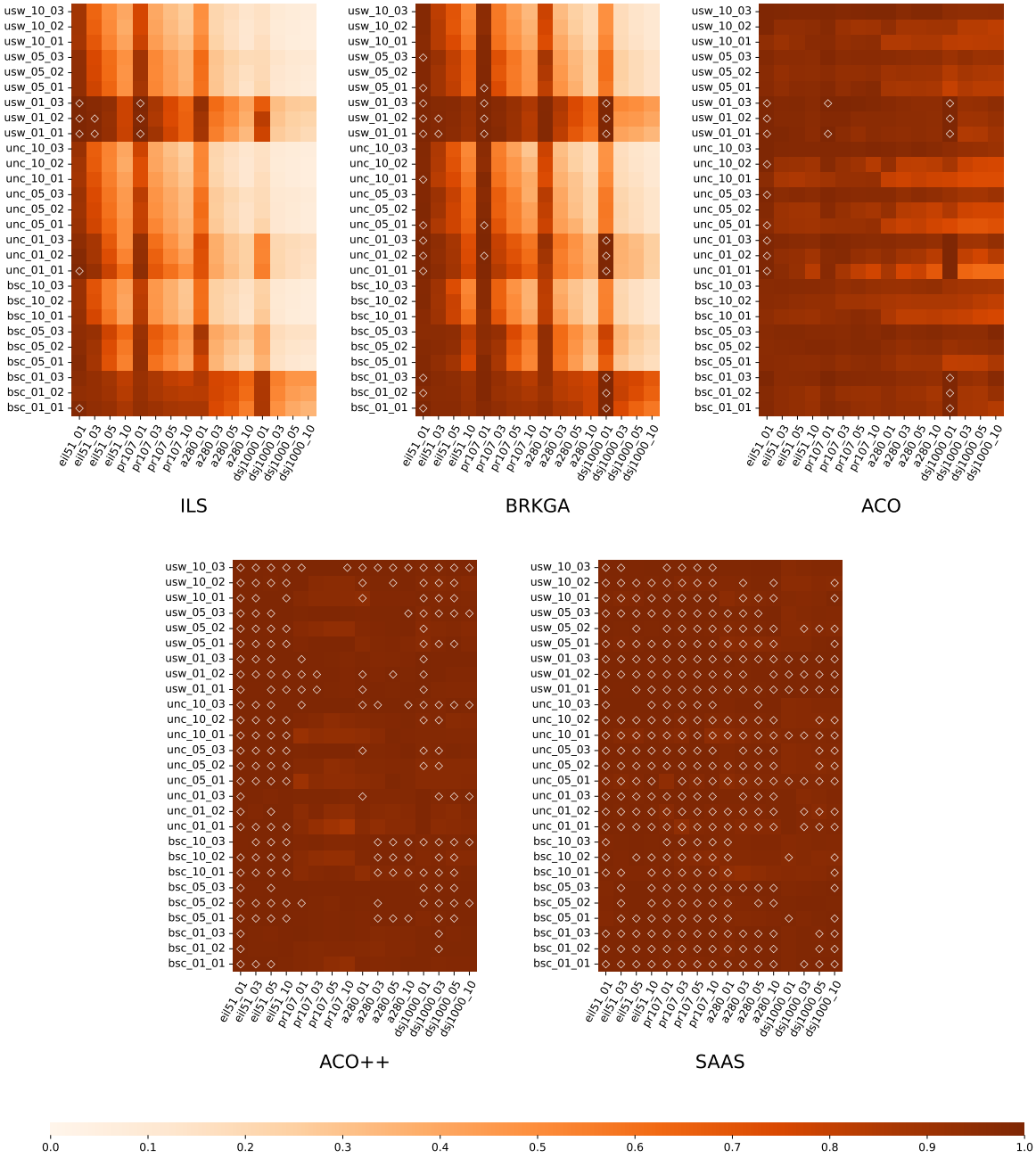
4.4.2 Kết quả thực nghiệm so sánh hiệu suất giữa các thuật toán

Để so sánh các cặp thuật toán với nhau về mặt số lượng lời giải tốt, chúng tôi đã xây dựng bảng 4.1, bảng thống kê phần trăm các lời giải của mỗi thuật toán tìm ra lời giải có chất lượng tốt hơn hoặc ngang bằng so với thuật toán khác. Ở mỗi cặp thuật toán, chúng tôi tính giá trị phần trăm số lượng dựa trên lời giải tốt nhất ở hai thuật toán của 30 lần chạy độc lập ở mỗi trường hợp.

i ↓ j →	ILS	BRKGA	ACO	ACO++	SAAS
ILS	-	2.55%	4.40%	2.55%	2.31%
BRKGA	100.00%	-	16.20%	8.80%	7.18%
ACO	97.22%	87.27%	-	5.79%	4.86%
ACO++	99.54%	95.83%	97.69%	-	41.90%
SAAS	99.77%	97.92%	98.61%	78.24%	-

Bảng 4.1: Bảng thống kê phần trăm chất lượng các lời giải của thuật toán i tốt hơn hoặc ngang bằng so với lời giải của thuật toán j.

Kết quả được thể hiện trong bảng này chứng thực cho những gì thể hiện trên hình 4.1. BRKGA hoàn toàn áp đảo ILS với tỉ lệ là 100% và các thuật toán dựa trên thuật toán đàn kiến thể hiện hiệu suất tốt hơn với hai thuật ILS và BRKGA. Thuật toán của chúng tôi SAAS và ACO++ đều thiện hiện hiệu suất vượt trội so với các thuật toán



Hình 4.1: Biểu đồ nhiệt của mỗi thuật toán thể hiện tỉ lệ kết quả lời giải trên từng trường hợp của thuật toán đó, với lời giải tốt nhất được tìm thấy cho trường hợp đó trong tất cả thực nghiệm. Mỗi ô trên mỗi biểu đồ nhiệt thể hiện một trường hợp. Độ đậm nhạt ở mỗi ô thể hiện tỉ lệ lời giải, màu sắc càng đậm thể hiện lời giải có kết quả gần bằng với lời giải tốt nhất được tìm thất. Ký hiệu hình thoi ở một số ô thể hiện thuật toán đó tìm ra lời giải tốt nhất được tìm thấy cho trường hợp tại vị trí đó.

khác với hơn 95% số lượng các trường hợp trên benchmark. SAAS có số lượng lời giải tốt hơn 78.24% ACO++.

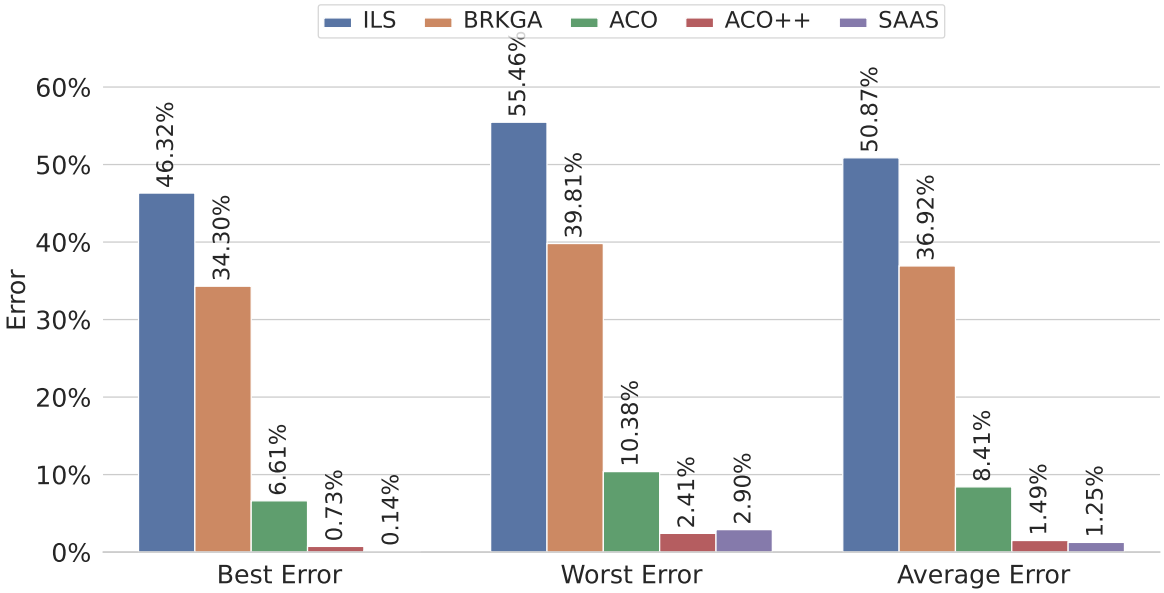
4.4.3 So sánh thống kê Wilcoxon signed-rank

Với hiệu suất tương đối và dẫn đầu của cả hai thuật toán SAAS và ACO++, chúng tôi tiến hành thử nghiệm Wilcoxon signed-rank để xác minh sự khác biệt giữa chúng. Sử dụng mức ý nghĩa là 5%, kết quả cho thấy rằng hiệu suất của SAAS thấp hơn thống kê so với ACO++ ở 170 trường hợp. Trong 86 trường hợp, không có sự chênh lệch đáng kể nào được quan sát giữa hiệu suất của hai thuật toán. Ngược lại, trong 176 trường hợp, SAAS thể hiện hiệu suất ưu việt so với ACO++.

Lưu ý quan trọng rằng trong khi thuật toán SAAS có kết quả cạnh tranh với thuật toán ACO++, nó sử dụng một bộ siêu tham số duy nhất trên tất cả 432 trường hợp trong quá trình thực nghiệm trên thang đo ThOP. Ngược lại, ACO++ sử dụng 48 bộ siêu tham số riêng biệt tương ứng với 48 nhóm trường hợp trong thang đo ThOP.

4.4.4 Kết quả thực nghiệm so sánh độ chênh lệch của lời giải của các thuật toán

Hình 4.2 mô tả độ chênh lệch trung bình cho các lời giải tốt nhất, tệ nhất và trung bình của các thuật toán trên tất cả các trường hợp trong thang đo ThOP. Cách chúng tôi tính các độ chênh lệch cho mỗi thuật được mô tả ở công thức 4.1. Với ”ThOP benchmark” là tập các trường hợp có trong thang đo ThOP và P_n^i là giá trị tổng lợi nhuận của lời giải của lần chạy độc lập thứ n của trường hợp i.



Hình 4.2: Biểu đồ cột thể hiện độ chênh lệch của lời giải của mỗi thuật toán so với lời giải tốt nhất tìm được. Biểu đồ thể hiện 3 cách tính độ chênh lệch: độ chênh lệch tốt nhất, độ chênh lệch tệ nhất, và độ chênh lệch trung bình.

$$
\begin{aligned}
\text{Best Error} &= \frac{1}{|\text{ThOP benchmark}|} \sum_{i \in \text{ThOP benchmark}} \max(P_1^i, P_2^i, \ldots, P_{30}^i) \\
\text{Worst Error} &= \frac{1}{|\text{ThOP benchmark}|} \sum_{i \in \text{ThOP benchmark}} \min(P_1^i, P_2^i, \ldots, P_{30}^i) \\
\text{Average Error} &= \frac{1}{|\text{ThOP benchmark}|} \sum_{i \in \text{ThOP benchmark}} \text{mean}(P_1^i, P_2^i, \ldots, P_{30}^i)
\end{aligned}
\tag{4.1}
$$

Thuật toán SAAS thể hiện hiệu suất đáng kể, với tỷ lệ chênh lệch thấp nhất là 0.14% và 1.25% ở các mục độ chênh lệch tốt nhất và độ chênh lệch trung bình. Mặc dù vậy độ chênh lệch trong trường hợp tồi nhất của thuật toán của chúng tôi cao hơn ACO++ khoảng 0.5%. Mặc dù không sử dụng các bộ tham số cụ thể cho mỗi nhóm trường hợp như ACO++ đã làm, thuật toán của chúng tôi, SAAS, liên tục mang lại hiệu suất ổn định với tỷ lệ chênh lệch thấp trên tất cả ba mục.

Chương 5

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Chúng tôi đã cải tiến thuật toán ACO++ bằng cách tích hợp hai cơ chế kiểm soát tham số, kỹ thuật Bay hơi lười biếng và kỹ thuật Phân cụm thứ bậc để tăng cường tính linh hoạt và hiệu suất. Hầu hết các tham số của chúng tôi được điều chỉnh tự động, một số trở nên tự thích ứng nhờ vào ES và một số khác trở nên thích ứng thông qua cơ chế thích ứng lấy cảm hứng từ AACO-NC. Kiến di chuyển trên các cạnh của cây cụm thứ bậc thay vì di chuyển từ thành phố này sang thành phố khác. Phân cụm thứ bậc đã giúp làm giảm chi phí tìm kiếm tuyến đường. Bay hơi lười biếng giúp giảm thiểu độ phức tạp thời gian của giai đoạn bay hơi bằng cách theo dõi trạng thái mong muốn và lịch sử, và chỉ bay hơi pheromone trên các cạnh được sử dụng. Trong khi ACO++ yêu cầu điều chỉnh riêng cấu hình siêu tham số cho mỗi nhóm trường hợp, SAAS mang lại kết quả xuất sắc với chỉ một cấu hình chạy trên tất cả 432 trường hợp đánh giá.

Trong công trình này, chúng tôi chỉ mới tập trung cải tiến ở giai đoạn tìm đường và chất lượng của tham số. Ở bài toán Điều Hướng Thu Thập và các thuật toán trước đó, tồn tại hai giai đoạn quan trọng khác mà chúng tôi chưa chạm đến. Đó là xây dựng chiến lược thu thập và tìm kiếm cục bộ. Các thuật toán cho bài toán Người Thu Thập Du Lịch (TTP) sở hữu các phép tìm kiếm cục bộ rất truyền cảm hứng [16]. Ta có thể điều chỉnh chúng để áp dụng được cho ThOP. Chiến lược thu thập của SAAS, được kế thừa từ ACO++, có độ phức tạp cao nhất trong tất cả giai đoạn, chứa nhiều khả năng có thể cải tiến.

Bộ trường hợp đánh giá của ThOP rất đa dạng và bao phủ rộng các khả năng. Tuy vậy, nó vẫn tồn tại vài thiếu sót không giống với thực tế. Ví dụ như mỗi thành phố đều có số lượng vật phẩm như nhau và bản đồ mang tính đối xứng. Bản đồ đối xứng nghĩa là quãng được từ A đến B và từ B đến A là như nhau. Điều này có thể dẫn đến đánh giá sai tiềm năng của thuật toán trong bối cảnh thực tế. Các nghiên cứu tương lai có thể hướng đến cải thiện yếu điểm này.

Các kĩ thuật chúng tôi đề xuất có khả năng tích hợp vào các phương pháp tân tiến trên nhiều bài toán khác, từ đó giúp cải thiện chất lượng lời giải. Ta cũng có thể tích hợp các yếu tố cửa sổ thời gian (time window), nhiều điểm kho chứa (multiple depot) hoặc đa mục tiêu vào bài toán Điều Hướng Thu Thập để thúc đẩy bài toán nghiên cứu gần hơn với vấn đề thực tế.

TÀI LIỆU THAM KHẢO

[1] Mohammad Reza Bonyadi, Zbigniew Michalewicz, and Luigi Barone. “The travelling thief problem: The first step in the transition from theoretical problems to realistic problems”. In: *2013 IEEE Congress on Evolutionary Computation*. 2013, pp. 1037–1044. doi: 10.1109/CEC.2013.6557681.

[2] Mohammad Reza Bonyadi et al. “Evolutionary Computation for Multicomponent Problems: Opportunities and Future Directions”. In: *Optimization in Industry: Present Practices and Future Scopes*. Ed. by Shubhabrata Datta and J. Paulo Davim. Cham: Springer International Publishing, 2019, pp. 13–30. isbn: 978-3-030-01641-8. doi: 10.1007/978-3-030-01641-8_2. url: https://doi.org/10.1007/978-3-030-01641-8_2.

[3] Jonatas B. C. Chagas and Markus Wagner. “Efficiently solving the thief orienteering problem with a max–min ant colony optimization approach”. In: *Optimization Letters* 16.8 (Nov. 2021), pp. 2313–2331. doi: 10.1007/s11590-021-01824-y. url: https://doi.org/10.1007/s11590-021-01824-y.

[4] Jonatas B.C. Chagas and Markus Wagner. “Ants can orienteer a thief in their robbery”. In: *Operations Research Letters* 48.6 (2020), pp. 708–714. issn: 0167-6377. doi: https://doi.org/10.1016/j.orl.2020.08.011. url: https://www.sciencedirect.com/science/article/pii/S0167637720301255.

[5] Y. Crama, A. W. J. Kolen, and E. J. Pesch. “Local search in combinatorial optimization”. In: *Artificial Neural Networks: An Introduction to ANN Theory and*

Practice. Ed. by P. J. Braspenning, F. Thuijsman, and A. J. M. M. Weijters. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 157–174. isbn: 978-3-540-49283-2. doi: 10.1007/BFb0027029. url: https://doi.org/10.1007/BFb0027029.

[6] Marco Dorigo and Christian Blum. "Ant colony optimization theory: A survey". In: *Theoretical Computer Science* 344.2 (2005), pp. 243–278. issn: 0304-3975. doi: https://doi.org/10.1016/j.tcs.2005.05.020. url: https://www.sciencedirect.com/science/article/pii/S0304397505003798.

[7] A.E. Eiben, R. Hinterding, and Z. Michalewicz. "Parameter control in evolutionary algorithms". In: *IEEE Transactions on Evolutionary Computation* 3.2 (1999), pp. 124–141. doi: 10.1109/4235.771166.

[8] Hayden Faulkner et al. "Approximate Approaches to the Traveling Thief Problem". In: *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. GECCO '15. Madrid, Spain: Association for Computing Machinery, 2015, pp. 385–392. isbn: 9781450334723. doi: 10.1145/2739480.2754716. url: https://doi.org/10.1145/2739480.2754716.

[9] Leonardo M. Faêda and André G. Santos. "A Genetic Algorithm for the Thief Orienteering Problem". In: *2020 IEEE Congress on Evolutionary Computation (CEC)*. 2020, pp. 1–8. doi: 10.1109/CEC48606.2020.9185848.

[10] Bruce L. Golden, Larry Levy, and Rakesh Vohra. "The orienteering problem". In: *Naval Research Logistics (NRL)* 34.3 (1987), pp. 307–318. doi: https://doi.org/10.1002/1520-6750(198706)34:3<307::AID-NAV3220340302>3.0.CO;2-D.

[11] N. Hansen, S.D. Muller, and P. Koumoutsakos. "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)." In: *Evolutionary Computation* 11.1 (2003), pp. 1–18.

[12] Nikolaus Hansen. *The CMA Evolution Strategy: A Tutorial*. 2023. arXiv: 1604.00772 [cs.LG].

[13] Manuel Iori, Juan-José Salazar-González, and Daniele Vigo. "An Exact Approach for the Vehicle Routing Problem with Two-Dimensional Loading Constraints". In: *Transportation Science* 41.2 (2007), pp. 253–264. doi: 10.1287/trsc.1060.0165. eprint: https://doi.org/10.1287/trsc.1060.0165. url: https://doi.org/10.1287/trsc.1060.0165.

[14] Fionn Murtagh and Pedro Contreras. "Algorithms for hierarchical clustering: an overview". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 2.1 (2012), pp. 86–97.

[15] Fionn Murtagh and Pedro Contreras. "Algorithms for hierarchical clustering: an overview, II". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 7.6 (2017), e1219.

[16] Majid Namazi et al. "Solving travelling thief problems using coordination based methods". In: *Journal of Heuristics* 29 (2023), pp. 487 –544. url: https://api.semanticscholar.org/CorpusID:263807019.

[17] Sergey Polyakovskiy and Rym M'Hallah. "An Intelligent Framework to Online Bin Packing in a Just-In-Time Environment". In: *Modern Approaches in Applied Intelligence*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 226–236. isbn: 978-3-642-21827-9.

[18] Sergey Polyakovskiy et al. “A Comprehensive Benchmark Set and Heuristics for the Traveling Thief Problem”. In: *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*. GECCO ’14. Vancouver, BC, Canada: Association for Computing Machinery, 2014, pp. 477–484. isbn: 9781450326629. doi: 10.1145/2576768.2598249. url: https://doi.org/10.1145/2576768.2598249.

[19] André G. Santos and Jonatas B.C. Chagas. “The Thief Orienteering Problem: Formulation and Heuristic Approaches”. In: *2018 IEEE Congress on Evolutionary Computation (CEC)*. 2018, pp. 1–9. doi: 10.1109/CEC.2018.8477853.

[20] Pranav Shetty and Suraj Singh. “Hierarchical clustering: a survey”. In: *International Journal of Applied Research* 7.4 (2021), pp. 178–181.

[21] Petr Stodola, Pavel Otřísal, and Kamila Hasilová. “Adaptive Ant Colony Optimization with node clustering applied to the Travelling Salesman Problem”. In: *Swarm and Evolutionary Computation* 70 (2022), p. 101056. issn: 2210-6502. doi: https://doi.org/10.1016/j.swevo.2022.101056. url: https://www.sciencedirect.com/science/article/pii/S2210650222000281.

[22] Thomas Stützle and Holger H. Hoos. “MAX–MIN Ant System”. In: *Future Generation Computer Systems* 16.8 (2000), pp. 889–914. issn: 0167-739X. doi: https://doi.org/10.1016/S0167-739X(00)00043-1. url: https://www.sciencedirect.com/science/article/pii/S0167739X00000431.