UỶ BAN NHÂN DÂN

THÀNH PHỐ HỒ CHÍ MINH

**TRƯỜNG ĐẠI HỌC SÀI GÒN**



**BÁO CÁO PHÂN TÍCH  
MÔ PHỎNG VÀ ĐÁNH GIÁ TÁC NHÂN THÔNG MINH,  
SO SÁNH HIỆU SUẤT CÁC THUẬT TOÁN TÌM KIẾM TRONG MÊ CUNG**

Học phần: Trí tuệ nhân tạo nâng cao

Giảng viên hướng dẫn: TS. Đỗ Như Tài

Sinh viên tham gia:

|  |  |
| --- | --- |
| Cao Tấn Đạt | 3122410069 |
| Lê Nguyễn Anh Đức | 3122410087 |
| Nguyễn Trọng Tấn Sang | 3121560076 |
|  |  |

**Thành phố Hồ Chí Minh, 10/2025**

**Mục lục**

[**I.** **N-QUEENS PROBLEM** 1](#_Toc211191315)

[**1.Mô tả bài toán** 1](#_Toc211191316)

[**1.1.** **Bài toán n-Queens** 1](#_Toc211191317)

[**1.2.** **Phương pháp tiếp cận** 1](#_Toc211191318)

[**2.Các thuật toán được triển khai** 1](#_Toc211191319)

[**2.1.** **Steepest\_Ascend Hill Climbing** 1](#_Toc211191320)

[**2.2.** **Stochastic Hill Climbing 1** 1](#_Toc211191321)

[**2.3.** **Stochastic Hill Climbing 2 (First-Choice)** 2](#_Toc211191322)

[**2.4.** **Random Restarts** 2](#_Toc211191323)

[**2.5.** **Simulated Annealing** 2](#_Toc211191324)

[**3.Kết quả thực nghiệm** 2](#_Toc211191325)

[**3.1.** **So sánh hiệu suất** 2](#_Toc211191326)

[**3.2.** **So sánh hiệu suất** 3](#_Toc211191327)

[**3.3.** **Phân tích khả năng mở rộng** 4](#_Toc211191328)

[**II.** **TSP PROBLEM** 4](#_Toc211191329)

[**1. GIỚI THIỆU** 4](#_Toc211191330)

[**1.1. Bài toán Người du lịch (TSP)** 4](#_Toc211191331)

[**1.2. Mục tiêu của Báo cáo** 5](#_Toc211191332)

[**2. CÔNG NGHỆ VÀ MÔI TRƯỜNG** 5](#_Toc211191333)

[**2.1. Ngôn ngữ lập trình** 5](#_Toc211191334)

[**2.2. Các thư viện chính** 5](#_Toc211191335)

[**3. CHI TIẾT VỀ BÀI TOÁN NGƯỜI DU LỊCH (TSP)** 5](#_Toc211191336)

[**3.1. Định nghĩa** 6](#_Toc211191337)

[**3.2. Không gian trạng thái và Hàm mục tiêu** 6](#_Toc211191338)

[**3.3. Phép dịch chuyển cục bộ (Local Moves)** 6](#_Toc211191339)

[**4. CÁC PHƯƠNG PHÁP GIẢI QUYẾT ĐƯỢC TRIỂN KHAI** 6](#_Toc211191340)

[**4.1. Leo đồi Ưu tiên Lựa chọn Đầu tiên (First-Choice Hill Climbing)** 6](#_Toc211191341)

[**4.2. Giải thuật Di truyền (Genetic Algorithm - GA)** 7](#_Toc211191342)

[**4.3. Sử dụng Gói TSP trong R (Làm cơ sở tham chiếu)** 8](#_Toc211191343)

[**5. PHÂN TÍCH VÀ SO SÁNH HIỆU NĂNG** 8](#_Toc211191344)

[**5.1. Phương pháp luận** 8](#_Toc211191345)

[**5.2. Bảng kết quả** 8](#_Toc211191346)

[**5.3. Phân tích chi tiết** 9](#_Toc211191347)

[**III.** **KẾT LUẬN** 9](#_Toc211191348)

# Bảng phân công nhiệm vụ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Tuần | Tên bài tập | Thành viên tham gia | % đóng góp (nếu trên 1 người) |
| 1 | 5 | Làm word và slide báo cáo kết quả | Đức | 33.33% |
| 2 | 5 | Làm phần bài tập trong file n\_qeens.ipynb | Đạt | 33.33% |
| 3 | 5 | Làm phần bài tập trong file traveling\_salesman-problem.ipynb | Sang | 33.33% |

1. **N-QUEENS PROBLEM**
2. **Mô tả bài toán**
   1. **Bài toán n-Queens**

Mục tiêu: Tìm cách sắp xếp n quân hậu trên bàn cờ n×n sao cho không có quân hậu nào nằm trên cùng hàng, cột hoặc đường chéo với bất kỳ quân hậu nào khác.

Không gian trạng thái: Một cách sắp xếp các quân hậu trên bàn cờ, giới hạn một quân hậu mỗi cột. Chúng ta biểu diễn trạng thái dưới dạng vector số nguyên q = {q₁, q₂, ..., qₙ}, trong đó mỗi số đại diện cho vị trí hàng của quân hậu từ trái sang phải.

Hàm mục tiêu: Số lượng xung đột theo cặp (các quân hậu đe dọa lẫn nhau).

Bài toán tối ưu: Minimize conflicts(q) với điều kiện q chỉ chứa một quân hậu mỗi cột.

Bước di chuyển cục bộ: Di chuyển một quân hậu sang hàng khác trong cùng cột của nó.

Điều kiện dừng: Luôn tồn tại nghiệm tối ưu với conflicts(q\*) = 0, nhưng các thuật toán tìm kiếm cục bộ có thể bị kẹt ở cực tiểu địa phương.

* 1. **Phương pháp tiếp cận**

Chúng tôi tiếp cận bài toán này bằng các thuật toán tìm kiếm cục bộ, lặp đi lặp lại cải thiện nghiệm ứng viên bằng cách thực hiện các sửa đổi nhỏ. Thách thức chính là thoát khỏi cực tiểu địa phương - các cấu hình mà không có bước di chuyển một quân hậu nào cải thiện nghiệm, nhưng chưa đạt được cực tiểu toàn cục.

1. **Các thuật toán được triển khai**
   1. **Steepest\_Ascend Hill Climbing**

Thuật toán này đánh giá TẤT CẢ các bước di chuyển cục bộ có thể (n × (n-1) bước) ở mỗi lần lặp và luôn chọn bước di chuyển tạo ra sự giảm xung đột lớn nhất.  
Đặc điểm:

* Xác định - luôn chọn bước di chuyển tốt nhất
* Toàn diện - đánh giá tất cả các bước di chuyển có thể
* Độ phức tạp thời gian: O(n²) đánh giá mỗi lần lặp
* Đảm bảo giảm đơn điệu cho đến khi đạt cực tiểu địa phương
  1. **Stochastic Hill Climbing 1**

Biến thể này đánh giá tất cả các bước di chuyển có thể nhưng chọn ngẫu nhiên từ tất cả các bước di chuyển cải thiện (bước đi lên) thay vì luôn chọn bước tốt nhất.  
Đặc điểm:

* Ngẫu nhiên - đưa tính ngẫu nhiên vào việc lựa chọn bước di chuyển
* Chi phí tính toán giống Steepest-Ascend (đánh giá tất cả bước di chuyển)
* Có thể khám phá các đường dẫn nghiệm khác nhau trong các lần chạy khác nhau
  1. **Stochastic Hill Climbing 2 (First-Choice)**

Còn được gọi là First-Choice Hill Climbing, thuật toán này chỉ tạo ra MỘT láng giềng ngẫu nhiên tại một thời điểm và chấp nhận nó nếu nó cải thiện hàm mục tiêu.  
  
Đặc điểm:

* Rất hiệu quả - chỉ một đánh giá cho mỗi bước di chuyển được chấp nhận
* Thực thi nhanh cho không gian trạng thái lớn
* Sử dụng ngưỡng (max\_tries = 100) để phát hiện cực tiểu địa phương
* Có thể bỏ lỡ bước di chuyển cải thiện tốt nhất nhưng được tốc độ đáng kể
  1. **Random Restarts**

Một chiến lược meta khởi động lại bất kỳ thuật toán leo đồi nào với trạng thái ban đầu ngẫu nhiên mới khi bị kẹt ở cực tiểu địa phương.  
Đặc điểm:

* Hoạt động với bất kỳ thuật toán leo đồi cơ sở nào
* Cải thiện đáng kể tỉ lệ thành công
* Chạy tối đa 100 lần khởi động lại hoặc cho đến khi tìm được nghiệm tối ưu
* Đánh đổi: tăng thời gian chạy để có chất lượng nghiệm tốt hơn
  1. **Simulated Annealing**

Lấy cảm hứng từ quá trình ủ luyện kim loại, thuật toán này chấp nhận các bước di chuyển tồi hơn với xác suất giảm dần theo thời gian theo lịch trình nhiệt độ.  
Xác suất chấp nhận: P(chấp nhận) = e^(-Δ/T) cho các bước tăng xung đột thêm Δ  
Lịch trình nhiệt độ: Làm lạnh hình học T(i+1) = α × T(i) với α = 0.95  
Tham số:

* Nhiệt độ ban đầu: 100
* Tỉ lệ làm lạnh: 0.95
* Nhiệt độ tối thiểu: 0.01

Điều này cho phép thuật toán thoát khỏi cực tiểu địa phương bằng cách thỉnh thoảng chấp nhận các bước di chuyển tồi hơn, với việc khám phá giảm dần khi nhiệt độ nguội đi.

1. **Kết quả thực nghiệm**
   1. **So sánh hiệu suất**

Mỗi thuật toán được kiểm tra 100 lần trên các bàn cờ được tạo ngẫu nhiên có kích thước 4×4 và 8×8. Bảng sau tóm tắt kết quả:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Thuật toán | Kích thước | T.gian TB (s) | Xung đột TB | Tỉ lệ T.công |
| Steepest asc. HC | 4×4 | 0.000186 | 0.73 | 38.0% |
| Stochastic HC 1 | 4×4 | 0.000166 | 0.76 | 35.0% |
| Stochastic HC 2 | 4×4 | 0.000889 | 0.81 | 32.0% |
| Simulated Annealing | 4×4 | 0.001153 | 0.07 | 93.0% |
| Steepest asc. HC | 8×8 | 0.002120 | 1.22 | 12.0% |
| Stochastic HC 1 | 8×8 | 0.003021 | 1.44 | 7.0% |
| Stochastic HC 2 | 8×8 | 0.002150 | 1.42 | 13.0% |

Nhận xét chính:

* Với bàn cờ 4×4: Simulated Annealing đạt tỉ lệ thành công 93%, vượt trội hơn đáng kể so với các biến thể leo đồi (32-38%).
* Với bàn cờ 8×8: Tất cả các thuật toán đều gặp khó khăn hơn, với tỉ lệ thành công giảm xuống còn 7-13%.
* Thời gian chạy: Simulated Annealing chậm hơn một chút nhưng tìm được nghiệm tốt hơn đáng kể.
* Xung đột trung bình: Simulated Annealing liên tục đạt được xung đột trung bình thấp hơn trên cả hai kích thước bàn cờ.
* Tác động của Random Restarts: Khi kết hợp với random restarts (tối đa 100 lần), tỉ lệ thành công tiến gần 100% cho tất cả các thuật toán.
  1. **So sánh hiệu suất**

Phân tích các mẫu hội tụ cho thấy các đặc điểm khác biệt cho từng thuật toán:

Steepest-Ascend Hill Climbing:

* Giảm đơn điệu (luôn cải thiện hoặc dừng lại)
* Cải thiện nhanh ban đầu sau đó đạt ngưỡng ở cực tiểu địa phương
* Trung bình 92% số lần chạy bị kẹt ở cực tiểu địa phương với bàn cờ 8×8

Các biến thể Stochastic Hill Climbing:

* Mẫu giảm đơn điệu tương tự
* Lựa chọn ngẫu nhiên tạo ra sự khác biệt trong các đường dẫn hội tụ
* Stochastic HC 2 cho thấy ít bước cải thiện hơn nhưng lớn hơn
* Tần suất cực tiểu địa phương: 76-90% với bàn cờ 8×8

Simulated Annealing:

* Hội tụ không đơn điệu (có thể tạm thời tăng xung đột)
* Khám phá rộng ở giai đoạn đầu, hội tụ khi nhiệt độ giảm
* Mạnh mẽ hơn trong việc tránh cực tiểu địa phương
* Đặc điểm hội tụ tổng thể tốt nhất
  1. **Phân tích khả năng mở rộng**

Kiểm tra khả năng mở rộng trên các bàn cờ kích thước n = 4, 8, 12, 16, 20 cho thấy:  
Độ phức tạp thời gian thực nghiệm:

* Steepest-Ascend HC: O(n^4.18) - Bậc ba hoặc tệ hơn
* Simulated Annealing: O(n^1.01) - Dưới bậc hai (gần như tuyến tính!)

Kết luận về khả năng mở rộng:

* Simulated Annealing mở rộng tốt hơn đáng kể cho các bài toán lớn
* Tính toán đơn giản hơn mỗi lần lặp mặc dù có nhiều lần lặp hơn
* Tốt hơn trong việc tìm nghiệm mà không cần khởi động lại
* Steepest-Ascend HC có chi phí tính toán cao
  + Đánh giá tất cả O(n²) bước di chuyển có thể mỗi lần lặp
  + Nhiều lần lặp làm tăng thêm chi phí
* Khuyến nghị cho bài toán n-Queens lớn (n > 20):
  + Simulated Annealing: Hiệu suất một lần chạy tốt nhất
  + Stochastic HC 2 với Restarts: Lựa chọn thay thế nhanh nhất
  + Tránh Steepest-Ascend HC cho các bàn cờ rất lớn

1. **TSP PROBLEM**

**1. GIỚI THIỆU**

**1.1. Bài toán Người du lịch (TSP)**

Bài toán Người du lịch là một trong những bài toán tối ưu hóa tổ hợp nổi tiếng và quan trọng nhất trong khoa học máy tính. Mục tiêu là tìm ra một chu trình ngắn nhất đi qua một tập hợp các thành phố cho trước, mỗi thành phố chỉ được thăm đúng một lần và quay trở về điểm xuất phát. Mặc dù phát biểu đơn giản, TSP là một bài toán NP-hard, nghĩa là không có thuật toán hiệu quả nào (chạy trong thời gian đa thức) có thể tìm ra lời giải tối ưu cho mọi trường hợp khi số lượng thành phố tăng lên.

**1.2. Mục tiêu của Báo cáo**

Phân tích cách cài đặt hai thuật toán tìm kiếm phổ biến: First-Choice Hill Climbing và Genetic Algorithm.

Giải thích các khái niệm, kiến thức khoa học máy tính đằng sau mỗi thuật toán. Đánh giá và so sánh hiệu năng của hai thuật toán dựa trên các tiêu chí: chất lượng lời giải (độ dài tour) và thời gian thực thi.

Mô tả các công nghệ và thư viện đã được sử dụng để thực hiện phân tích.

**2. CÔNG NGHỆ VÀ MÔI TRƯỜNG**

**2.1. Ngôn ngữ lập trình**

Python (phiên bản 3.10): Ngôn ngữ chính được sử dụng để cài đặt các thuật toán và tiến hành so sánh.

R: Được sử dụng thông qua thư viện rpy2 để tận dụng gói TSP có sẵn, cung cấp một lời giải tham chiếu chất lượng cao.

**2.2. Các thư viện chính**

Python:

* numpy, pandas: Xử lý dữ liệu số và cấu trúc dữ liệu (tọa độ thành phố, ma trận khoảng cách).
* matplotlib: Trực quan hóa bản đồ các thành phố và lộ trình tìm được.
* scipy: Tính toán ma trận khoảng cách Euclid giữa các thành phố một cách hiệu quả.
* random: Cung cấp các hàm sinh số ngẫu nhiên, cốt lõi cho việc khởi tạo và các thao tác trong thuật toán.
* rpy2: Cầu nối cho phép thực thi mã R ngay trong môi trường Python.

R:

* TSP: Một gói chuyên dụng cung cấp các cấu trúc dữ liệu và thuật toán mạnh để giải quyết TSP.
* microbenchmark: Đo lường hiệu năng của các đoạn mã R.

**3. CHI TIẾT VỀ BÀI TOÁN NGƯỜI DU LỊCH (TSP)**

**3.1. Định nghĩa**

Mục tiêu: Tìm tour ngắn nhất đi qua N thành phố, mỗi thành phố một lần và quay về điểm bắt đầu.

Đầu vào: Một ma trận khoảng cách đối xứng di,j giữa thành phố i và j.

**3.2. Không gian trạng thái và Hàm mục tiêu**

Không gian trạng thái: Tập hợp tất cả các hoán vị có thể có của N thành phố. Một trạng thái là một tour, biểu diễn bằng một vector hoán vị π.

A black background with white text

AI-generated content may be incorrect.Hàm mục tiêu: Tối thiểu hóa tổng độ dài của tour, được tính bằng công thức:

**3.3. Phép dịch chuyển cục bộ (Local Moves)**

Để di chuyển từ trạng thái này sang trạng thái khác trong không gian tìm kiếm, notebook sử dụng phép dịch chuyển 2-opt swap: hoán đổi vị trí của hai thành phố bất kỳ trong một tour. Đây là một phép dịch chuyển đơn giản nhưng hiệu quả để tạo ra các "hàng xóm" (neighbor) của một tour hiện tại.

**4. CÁC PHƯƠNG PHÁP GIẢI QUYẾT ĐƯỢC TRIỂN KHAI**

**4.1. Leo đồi Ưu tiên Lựa chọn Đầu tiên (First-Choice Hill Climbing)**

Đây là một biến thể của thuật toán tìm kiếm cục bộ Leo đồi (Hill Climbing).

Kiến thức & Nguyên lý:

* Thuật toán bắt đầu từ một lời giải ngẫu nhiên.
* Ở mỗi bước, nó không tìm kiếm tất cả các "hàng xóm" để chọn ra cái tốt nhất (như Steepest-Ascent Hill Climbing), mà chỉ tạo ra một hàng xóm ngẫu nhiên (bằng cách hoán vị 2 thành phố).
* Nếu hàng xóm này tốt hơn (tour ngắn hơn) so với trạng thái hiện tại, thuật toán sẽ di chuyển đến trạng thái đó. Nếu không, nó sẽ giữ nguyên và thử tạo một hàng xóm ngẫu nhiên khác.
* Quá trình lặp lại cho đến khi đạt đủ số lần lặp (max\_iter).

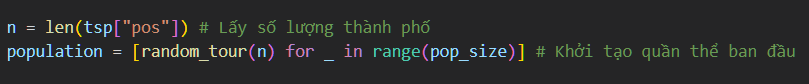
Ưu điểm: Nhanh và đơn giản để cài đặt.

Nhược điểm: Rất dễ bị "mắc kẹt" tại các điểm tối ưu cục bộ (local optima) – những lời giải tốt hơn các hàng xóm của nó nhưng không phải là lời giải tốt nhất toàn cục.

**4.2. Giải thuật Di truyền (Genetic Algorithm - GA)**

Đây là một phương pháp tìm kiếm metaheuristic lấy cảm hứng từ quá trình tiến hóa tự nhiên.

Kiến thức & Nguyên lý:

* Quần thể (Population): Thuật toán duy trì một tập hợp gồm nhiều lời giải (tours) khác nhau, gọi là quần thể.

* A black background with white text

  AI-generated content may be incorrect.Độ thích nghi (Fitness): Mỗi lời giải được đánh giá bằng một hàm "fitness". Trong bài toán này, fitness chính là nghịch đảo của độ dài tour (tour càng ngắn, fitness càng cao).

* A black background with white text

  AI-generated content may be incorrect.Chọn lọc (Selection): Các lời giải tốt hơn có xác suất được chọn để "sinh sản" cao hơn. Trong notebook này, phương pháp chọn lọc là Elitism, tức là giữ lại 2 cá thể tốt nhất của thế hệ cũ và đưa thẳng vào thế hệ mới.

* A computer screen shot of text

  AI-generated content may be incorrect.Lai ghép (Crossover): Hai lời giải "cha mẹ" được kết hợp để tạo ra lời giải "con". Phương pháp được sử dụng là Ordered Crossover (OX1): một đoạn gen (chuỗi thành phố) từ cha mẹ 1 được sao chép vào con, sau đó các vị trí còn lại được điền bằng các gen từ cha mẹ 2 theo đúng thứ tự và không trùng lặp.

* Đột biến (Mutation): Một thay đổi nhỏ, ngẫu nhiên được áp dụng cho lời giải con (ví dụ: hoán vị 2 thành phố) để duy trì sự đa dạng trong quần thể và tránh tối ưu cục bộ.

A screen shot of a computer program

AI-generated content may be incorrect.

Ưu điểm: Khả năng thoát khỏi các điểm tối ưu cục bộ tốt hơn so với Leo đồi.

Nhược điểm: Phức tạp hơn, nhiều tham số cần tinh chỉnh (kích thước quần thể, tỷ lệ đột biến,...), và thường tốn nhiều thời gian tính toán hơn.

**4.3. Sử dụng Gói TSP trong R (Làm cơ sở tham chiếu)**

Notebook sử dụng hàm solve\_TSP (tsp, rep = 100) từ gói TSP của R. Tham số rep = 100 có nghĩa là thuật toán sẽ chạy 100 lần với các điểm bắt đầu ngẫu nhiên khác nhau và trả về kết quả tốt nhất. Điều này làm tăng đáng kể khả năng tìm được lời giải gần tối ưu.

**5. PHÂN TÍCH VÀ SO SÁNH HIỆU NĂNG**

**5.1. Phương pháp luận**

Hai thuật toán Hill Climbing và Genetic Algorithm được so sánh trên các bài toán TSP với số lượng thành phố khác nhau: 10, 20, 30 và 100. Các tiêu chí so sánh bao gồm:

Chất lượng lời giải: Độ dài tour ngắn nhất tìm được.

Thời gian thực thi: Thời gian (tính bằng mili giây) để thuật toán chạy xong.

**5.2. Bảng kết quả**

A screen shot of a computer

AI-generated content may be incorrect.(Bảng được trích xuất từ output của notebook)

**5.3. Phân tích chi tiết**

Chất lượng lời giải (Tour Length):

* Với N=10 và N=20, cả hai thuật toán đều cho ra kết quả khá tương đồng.
* Với N=30, có một sự khác biệt lớn: Hill Climbing tìm được lời giải tốt hơn đáng kể (6.52 so với 10.57). Điều này cho thấy tính ngẫu nhiên của các thuật toán heuristic. Trong lần chạy cụ thể này, Genetic Algorithm đã hội tụ sớm về một vùng tối ưu cục bộ không tốt, trong khi Hill Climbing may mắn tìm được một vùng tốt hơn. Để có kết luận chính xác hơn, cần phải chạy mỗi thuật toán nhiều lần và lấy kết quả trung bình.

Thời gian thực thi (Runtime):

* Hill Climbing có thời gian chạy khá ổn định và nhanh chóng qua các kích thước bài toán khác nhau.
* Genetic Algorithm có thời gian chạy tăng rõ rệt khi số lượng thành phố tăng lên. Điều này là dễ hiểu vì ở mỗi thế hệ, GA phải quản lý và đánh giá cả một quần thể lớn, thực hiện các phép lai ghép và đột biến phức tạp hơn.

Khả năng mở rộng (Scalability):

* Hill Climbing cho thấy khả năng mở rộng về thời gian tốt hơn trong phạm vi thử nghiệm.
* Genetic Algorithm sẽ tốn nhiều tài nguyên hơn khi bài toán lớn hơn, nhưng bù lại nó có tiềm năng tìm kiếm trong không gian lời giải rộng hơn.

1. **KẾT LUẬN**

Qua quá trình nghiên cứu và thực nghiệm trên hai bài toán tối ưu điển hình n-Queens và TSP-like, có thể rút ra những nhận định quan trọng về hiệu quả và đặc trưng của các thuật toán tìm kiếm cục bộ. Kết quả cho thấy rằng không tồn tại một thuật toán nào tối ưu tuyệt đối cho mọi trường hợp; thay vào đó, hiệu năng phụ thuộc sâu sắc vào đặc tính của bài toán, chiến lược khám phá khai thác, và nguồn tài nguyên tính toán sẵn có.

Đối với bài toán n-Queens, các thuật toán như Hill Climbing và Simulated Annealing thể hiện khả năng giải quyết hiệu quả không gian nghiệm có mật độ cực tiểu địa phương cao. Simulated Annealing đặc biệt nổi bật nhờ khả năng thoát khỏi bẫy cực tiểu nhờ cơ chế chấp nhận bước di chuyển “kém hơn” có xác suất, giúp duy trì khả năng tìm kiếm toàn cục. Khi kết hợp random restarts, ngay cả các thuật toán đơn giản như Hill Climbing cũng đạt tỷ lệ thành công gần tuyệt đối, khẳng định vai trò của tính ngẫu nhiên trong việc tăng cường độ tin cậy của các phương pháp cục bộ.

Trong khi đó, với bài toán tối ưu đường đi (TSP), sự khác biệt giữa First-Choice Hill Climbing và Genetic Algorithm thể hiện rõ rệt hơn. Hill Climbing hoạt động nhanh, ổn định và phù hợp cho các bài toán nhỏ hoặc yêu cầu thời gian gấp. Ngược lại, Genetic Algorithm tỏ ra ưu thế ở những bài toán có không gian nghiệm lớn và phức tạp, nơi cần khả năng khám phá đa điểm và duy trì tính đa dạng quần thể. Tuy nhiên, chi phí tính toán của GA tăng nhanh theo quy mô bài toán, đòi hỏi sự cân nhắc giữa chất lượng lời giải và thời gian xử lý.

Tóm lại, nghiên cứu này khẳng định rằng hiểu rõ bản chất và đặc trưng của bài toán là điều kiện tiên quyết để lựa chọn và tinh chỉnh thuật toán tối ưu phù hợp, từ đó đạt được sự cân bằng tốt nhất giữa tốc độ, độ tin cậy và chất lượng lời giải.