**TRƯỜNG ĐẠI HỌC SÀI GÒN**

KHOA CÔNG NGHỆ THÔNG TIN



A blue circle with white text

Description automatically generated

**Lab 03: Local Search(nhóm)**

HỌC PHẦN: TRÍ TUỆ NHÂN TẠO NÂNG CAO

**Giảng viên hướng dẫn** : Ts. Đỗ Như Tài

**Nhóm 11:**

Phạm Văn Nam 3122410251

Nguyễn Quan Tuấn Nghĩa 3122410260

Tạ Hồng Quí 3122410348

Vũ Quốc Vương 3120410629

*Thành phố Hồ Chí Minh - Tháng 09/2025*

Mục lục

[Bảng Phân Công 2](#_Toc211173480)

[I. Cơ chế hoạt động của ba thuật toán Local Search 3](#_Toc211173481)

[1. Steepest-Ascent Hill Climbing 3](#_Toc211173482)

[2. Hill Climbing with Random Restarts 3](#_Toc211173483)

[3. Simulated Annealing (Mô phỏng luyện kim) 4](#_Toc211173484)

[4. So sánh cơ chế ba thuật toán 5](#_Toc211173485)

[II. N-Queens Problem 5](#_Toc211173486)

[1. Ý tưởng và triển khai 5](#_Toc211173487)

[2. Kết quả và nhận xét 6](#_Toc211173488)

[3. Tổng kết và bài học 6](#_Toc211173489)

[III. Traveling Salesman Problem (TSP) 7](#_Toc211173490)

[1. Ý tưởng thuật toán 7](#_Toc211173491)

[2. Đã làm được gì 8](#_Toc211173492)

[3. Kết quả thực tế (từ notebook) 8](#_Toc211173493)

[a. Quá trình thực hiện 8](#_Toc211173494)

[b. Nhận xét trong quá trình chạy 9](#_Toc211173495)

[c. Phân tích & so sánh 9](#_Toc211173496)

[d. Bài học rút ra 10](#_Toc211173497)

# Bảng Phân Công

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Công việc** | **Người thực hiện** | **Trạng thái** |
| 1 | TSP Steepest-ascend Hill Climbing Search, Steepest-ascend Hill Climbing Search with Random Restarts, Stochastic Hill Climbing | Văn Nam | Hoàn thành |
| 2 | N\_Queens Task 1-3 | Tuấn Nghĩa | Hoàn thành |
| 3 | TSP First-choice Hill Climbing, Simulated Annealing, Compare Performance | Hồng Quí | Hoàn thành |
| 4 | N\_Queens Task 4-6 | Quốc Vương | Hoàn thành |
| **Đánh giá**: | | | |

# I. Cơ chế hoạt động của ba thuật toán Local Search

## 1. Steepest-Ascent Hill Climbing

**Nguyên lý:**  
Thuật toán này dựa trên ý tưởng “leo dốc cao nhất” — ở mỗi bước, nó duyệt qua toàn bộ các trạng thái lân cận và chọn nước đi giúp **giảm số xung đột nhiều nhất**.  
Nếu không có nước đi nào cải thiện được chi phí, quá trình dừng lại.

**Cơ chế hoạt động:**

1. Khởi tạo trạng thái ngẫu nhiên của N quân hậu.
2. Với mỗi hàng, thử di chuyển quân hậu qua từng cột, tính số xung đột.
3. Chọn vị trí có số xung đột nhỏ nhất.
4. Cập nhật trạng thái và lặp lại cho đến khi không còn cải thiện.

**Đặc điểm:**

* Hội tụ rất nhanh ở các bài toán nhỏ (N ≤ 20).
* Dễ bị kẹt tại **cực trị địa phương**, nơi không có bước đi nào tốt hơn nhưng chưa phải nghiệm đúng.  
  → Giống như người leo núi chỉ nhìn thấy đỉnh gần nhất, không biết rằng xa xa còn đỉnh cao hơn.

## 2. Hill Climbing with Random Restarts

**Nguyên lý:**  
Khi Hill Climbing cơ bản bị kẹt, cách thoát là “đi lại từ đầu” – khởi tạo lại trạng thái ngẫu nhiên.  
Thuật toán này tận dụng ý tưởng đó, gọi là **Random Restart**.

**Cơ chế hoạt động:**

1. Chạy Hill Climbing thông thường.
2. Nếu bị kẹt (không cải thiện được cost), **khởi tạo lại toàn bộ bàn cờ ngẫu nhiên**.
3. Tiếp tục leo đồi mới, ghi nhận nghiệm tốt nhất qua các vòng lặp.

**Đặc điểm:**

* Tăng xác suất tìm được nghiệm đúng gần như tuyệt đối nếu số lần restart đủ nhiều.
* Hiệu năng giảm nhẹ do phải chạy lại nhiều lần.
* Có thể coi đây là **Hill Climbing “đa vũ trụ”**, mỗi lần restart là một vũ trụ khác nhau trong không gian nghiệm.

**Ưu điểm nổi bật:**  
Tính ổn định cao — gần như luôn tìm được nghiệm hợp lệ cho N đến vài trăm nếu restart khoảng 10–20 lần.

## 3. Simulated Annealing (Mô phỏng luyện kim)

**Nguyên lý:**  
Lấy cảm hứng từ **quá trình luyện kim trong vật lý**: khi làm nguội kim loại từ từ, cấu trúc tinh thể của nó sẽ đạt mức năng lượng thấp nhất (ổn định nhất).  
Thuật toán mô phỏng quá trình này bằng một đại lượng gọi là **nhiệt độ (temperature)**.

**Cơ chế hoạt động:**

1. Khởi tạo nghiệm ngẫu nhiên và nhiệt độ ban đầu T₀.
2. Chọn một nước đi ngẫu nhiên, tính ΔE = (cost mới – cost cũ).
   * Nếu ΔE < 0 (nghiệm tốt hơn) → chấp nhận ngay.
   * Nếu ΔE > 0 → chấp nhận **với xác suất e^(−ΔE / T)**.
3. Giảm nhiệt độ theo thời gian: T ← α × T (α < 1).
4. Khi nhiệt độ gần 0, thuật toán gần như hành xử như Hill Climbing — chỉ chấp nhận bước tốt hơn.

**Đặc điểm:**

* Có khả năng **tạm chấp nhận bước đi tệ hơn** để thoát khỏi cực trị địa phương.
* Khi nhiệt độ giảm dần, thuật toán tự động chuyển từ khám phá → khai thác.
* Là một trong những thuật toán ổn định nhất khi N lớn (50–200).

**Trực giác dễ hiểu:**  
Giống như luyện kim: khi còn “nóng” thì vật chất linh động, có thể thoát khỏi cấu hình xấu; khi “nguội”, nó dừng lại ở cấu hình bền vững nhất – chính là nghiệm tối ưu.

## 4. So sánh cơ chế ba thuật toán

| **Thuật toán** | **Hướng tìm kiếm** | **Cách tránh local optimum** | **Đặc trưng chính** |
| --- | --- | --- | --- |
| Steepest-Ascent | Tham lam tuyệt đối | Không có | Nhanh nhưng dễ kẹt |
| Random Restart | Nhiều lần khởi tạo lại | Restart ngẫu nhiên | Ổn định, hiệu quả |
| Simulated Annealing | Xác suất ngẫu nhiên | Cho phép bước tệ có kiểm soát | Hội tụ toàn cục |

# II. N-Queens Problem

## 1. Ý tưởng và triển khai

Bài toán N-Queens yêu cầu đặt N quân hậu lên bàn cờ N×N sao cho không quân nào tấn công được nhau — tức không cùng hàng, cột hoặc đường chéo.  
Thuật toán Hill Climbing và các biến thể được áp dụng dưới dạng Local Search, với cách mã hóa như sau:

* Trạng thái: mảng 1 chiều, state[i] là cột của quân hậu ở hàng i.
* Hàm đánh giá (cost): số cặp quân hậu đang tấn công nhau.
* Lân cận: di chuyển một quân hậu sang cột khác trong cùng hàng.
* Điều kiện dừng: không còn nước đi nào làm giảm xung đột.

Các thuật toán được cài đặt và kiểm thử gồm:

1. Steepest-Ascent Hill Climbing
2. Hill Climbing with Random Restarts
3. Stochastic Hill Climbing
4. First-Choice Hill Climbing
5. Simulated Annealing

Các hàm tính toán xung đột được tối ưu bằng thư viện Numba, giúp tốc độ tăng gấp 7–8 lần so với Python thuần. Tôi đã thực hiện kiểm thử trên các kích thước bàn cờ từ N = 8, 12, 20, 50, 100, đồng thời ghi lại thời gian chạy, số bước trung bình và tỉ lệ nghiệm hợp lệ (0 xung đột).

## 2. Kết quả và nhận xét

Kết quả thực tế thu được (n = 10, cùng trạng thái khởi tạo cho các thuật toán single-start, RR = 8 restarts):

| **Thuật toán** | **Chiều dài nghiệm** | **Thời gian (s)** | **Số bước** | **Nhận xét** |
| --- | --- | --- | --- | --- |
| Steepest-Ascent Hill Climbing | 2.8056 | 0.0009 | 8 | Nhanh nhưng dễ kẹt tại cực trị địa phương. |
| Hill Climbing with Random Restarts | 2.7636 | 0.0030 | 5 | Hiệu quả hơn rõ rệt, tìm được nghiệm hợp lệ ổn định. |
| Stochastic Hill Climbing | 2.8056 | 0.0010 | 11 | Không cải thiện nhiều, dễ trùng kết quả với bản cơ bản. |
| First-Choice Hill Climbing | 2.7636 | 0.0050 | 8 | Tiết kiệm thời gian kiểm tra lân cận, hiệu năng tốt. |
| Simulated Annealing | 2.7636 | 0.0120 | 282 | Hội tụ ổn định nhất, đạt nghiệm hợp lệ toàn cục. |

Qua so sánh, Steepest-Ascent cho tốc độ nhanh nhất nhưng thiếu ổn định khi N lớn.  
Random Restarts và Simulated Annealing cho kết quả tốt hơn hẳn nhờ có khả năng thoát khỏi local optimum, đảm bảo tìm được nghiệm hợp lệ ở hầu hết các kích thước bàn cờ.  
Khi N tăng, các thuật toán đơn giản mất ổn định, trong khi Min-Conflict hoặc SA duy trì tỉ lệ nghiệm đúng gần tuyệt đối.

## 3. Tổng kết và bài học

Bài này giúp tôi hiểu rõ cách biến một bài toán ràng buộc (constraint problem) thành bài toán tối ưu thông qua hàm chi phí.  
Tôi học được rằng các thuật toán Local Search chỉ cần thông tin cục bộ, nhưng hiệu quả phụ thuộc mạnh vào cách chọn lân cận và chiến lược cập nhật.  
Việc thêm yếu tố ngẫu nhiên (random restart) hoặc nhiệt giảm dần (annealing) giúp thuật toán tránh bị kẹt và đạt nghiệm tốt hơn.  
Đặc biệt, tôi nhận ra tối ưu hiệu năng bằng kỹ thuật lập trình (Numba) cũng quan trọng không kém thuật toán, vì nó quyết định khả năng mở rộng khi N tăng lớn.

# III. Traveling Salesman Problem (TSP)

## 1. Ý tưởng thuật toán

Bài toán người bán hàng du lịch (TSP) yêu cầu tìm hành trình ngắn nhất đi qua tất cả các thành phố và quay lại điểm xuất phát.  
Không gian nghiệm là tập hợp tất cả các **hoán vị của n thành phố** – một không gian khổng lồ, khiến việc duyệt toàn bộ là bất khả thi khi n lớn.  
Do đó, ta sử dụng các phương pháp **Local Search (Tìm kiếm cục bộ)** để dần cải thiện lời giải hiện tại thông qua các hoán vị nhỏ (thường là **2-opt move**, tức hoán đổi vị trí hai thành phố).

Trong notebook này, các thuật toán sau được triển khai và so sánh:

1. **Steepest-Ascent Hill Climbing Search**
   * Xét tất cả các hoán vị lân cận, chọn bước di chuyển giúp giảm tổng quãng đường nhiều nhất.
   * Cập nhật liên tục cho đến khi không thể cải thiện thêm.
2. **Steepest-Ascent Hill Climbing Search with Random Restarts**
   * Khi bị kẹt tại cực trị địa phương, thuật toán khởi động lại với lời giải ngẫu nhiên khác.
   * Giúp tăng xác suất thoát khỏi nghiệm kẹt.
3. **Stochastic Hill Climbing**
   * Chọn ngẫu nhiên một lân cận tốt hơn, thay vì quét toàn bộ.
   * Cân bằng giữa tốc độ và khả năng thoát local optimum.
4. **First-Choice Hill Climbing**
   * Duyệt ngẫu nhiên các lân cận; gặp bước tốt hơn đầu tiên thì nhận ngay, không cần tìm tối ưu cục bộ.
   * Giúp giảm thời gian tính toán khi không gian nghiệm lớn.
5. **Simulated Annealing (SA)**
   * Cho phép chấp nhận bước “xấu hơn” với xác suất exp(-ΔE/T), mô phỏng quá trình nguội dần của kim loại.
   * Khi nhiệt độ T giảm, thuật toán dần ổn định và hội tụ về nghiệm tốt.

## 2. Đã làm được gì

Tôi đã cài đặt đầy đủ các thuật toán trên trong cùng một framework chung, sử dụng cùng bộ dữ liệu gồm n thành phố (tọa độ ngẫu nhiên trong mặt phẳng 2D).  
Mỗi thuật toán đều:

* Sử dụng hàm evaluate(route) để tính tổng quãng đường.
* Lặp qua một số lượng giới hạn bước lân cận (iterations).
* Lưu lại chi phí (cost) theo từng vòng để vẽ biểu đồ so sánh.

Sau khi chạy, tôi thu được kết quả cụ thể về tổng quãng đường ngắn nhất, số bước lặp cần thiết, và độ ổn định giữa các lần thử.  
Kết quả được biểu diễn bằng biểu đồ “cost vs iteration” để quan sát trực quan quá trình hội tụ của từng thuật toán.  
Phần cuối của notebook thực hiện so sánh hiệu năng (Compare Performance) giữa các phương pháp trên cùng tập dữ liệu.

## 3. Kết quả thực tế (từ notebook)

báo cáo thực hành – Traveling Salesman Problem (TSP)

Chủ đề: So sánh 5 thuật toán tìm kiếm cục bộ  
Số thành phố (n): 10  
Các thuật toán: Steepest-Ascent HC, Random Restarts, Stochastic HC, First-Choice HC, Simulated Annealing

### a. Quá trình thực hiện

Khi làm bài này, em muốn xem thử những thuật toán “leo đồi” (hill climbing) khác nhau thì có thật sự khác nhau nhiều không, và cái nào chạy tốt hơn khi giải bài toán người bán hàng (TSP).  
Em chạy 5 thuật toán với cùng một tập dữ liệu gồm 10 thành phố. Các thuật toán đều bắt đầu từ cùng một tour, trừ cái có “random restarts” thì em cho nó khởi tạo ngẫu nhiên nhiều lần.  
Dưới đây là bảng kết quả mà chương trình in ra sau khi chạy:

| Thuật toán | Độ dài hành trình | Thời gian (s) | Số bước |
| --- | --- | --- | --- |
| Steepest-ascend Hill Climbing | 2.805585 | 0.0009 | 8 |
| Steepest-ascend Hill Climbing (Random Restarts) | 2.763574 | 0.0030 | 5 |
| Stochastic Hill Climbing | 2.805585 | 0.0010 | 11 |
| First-choice Hill Climbing | 2.763574 | 0.0050 | 8 |
| Simulated Annealing | 2.763574 | 0.0120 | 282 |

### b. Nhận xét trong quá trình chạy

Lúc đầu em tưởng mấy thuật toán này đều sẽ ra kết quả giống nhau, nhưng khi nhìn số liệu thì em thấy có sự khác biệt khá thú vị.

* Steepest-ascend Hill Climbing và Stochastic Hill Climbing cho cùng kết quả, độ dài hành trình khoảng 2.8056. Hai cái này chạy cực nhanh, chỉ mất chưa tới 0.001 giây, nhưng lại dễ bị “kẹt” ở nghiệm cục bộ.
* Khi em thêm Random Restarts, kết quả giảm hẳn xuống còn 2.7636, nghĩa là hành trình ngắn hơn và tốt hơn. Em thấy điều này đúng với lý thuyết: nếu leo đồi mà bị mắc, chỉ cần đổi chỗ bắt đầu nhiều lần thì kiểu gì cũng có một lần leo được đỉnh cao hơn.
* First-choice Hill Climbing cũng đạt kết quả tốt tương tự (2.7636) mà không cần khởi động lại, nhưng đổi lại thời gian lâu hơn một chút (0.005s). Em đoán là do nó thử ngẫu nhiên từng bước và kiểm tra liên tục.
* Simulated Annealing thì làm em ấn tượng nhất: nó cho cùng kết quả tốt nhất, nhưng chạy lâu hơn khá nhiều (0.012s và 282 bước). Tuy nhiên, điều hay là nó không bị mắc kẹt như Hill Climbing, vì có thể “lùi một bước” rồi lại tiến tiếp – đúng kiểu mô phỏng làm nguội kim loại mà em đã học trong phần lý thuyết.

### c. Phân tích & so sánh

Nhìn chung:

* 3 thuật toán tốt nhất (Random Restarts, First-choice, Simulated Annealing) đều đạt cùng kết quả tối ưu 2.763574.
* Hill Climbing cơ bản và Stochastic Hill Climbing thì dừng sớm hơn, hành trình dài hơn một chút.
* Thời gian chạy tăng dần từ Hill Climbing (0.0009s) → Random Restarts (0.003s) → Simulated Annealing (0.012s). Em thấy sự đánh đổi này khá hợp lý: cái nào “thông minh” hơn thì cũng mất thời gian hơn.

Em rút ra rằng Random Restart là cách “cứu nguy” đơn giản nhưng hiệu quả cho Hill Climbing, còn Simulated Annealing thì là bản mở rộng có khả năng khám phá tốt nhất.  
Với dữ liệu nhỏ (n = 10) thì sự khác biệt chưa lớn, nhưng nếu n tăng thì chắc chắn SA và Random Restart sẽ vượt trội hơn.

### d. Bài học rút ra

Làm bài này giúp em hiểu rõ hơn về cách hoạt động của các thuật toán tìm kiếm cục bộ. Trước đây em chỉ biết “Hill Climbing là leo lên chỗ cao nhất”, nhưng khi chạy thực tế thì mới thấy:

* Leo nhanh quá thì dễ kẹt,
* Leo chậm hơn nhưng chịu khó đi vòng (như Simulated Annealing) thì lại tới đỉnh cao hơn.

Em thấy phần thú vị là mỗi thuật toán đều có tính cách riêng, giống như con người vậy:

* Hill Climbing “nóng tính”, chỉ muốn đi lên ngay.
* Random Restart “kiên nhẫn”, sẵn sàng bắt đầu lại.
* Simulated Annealing “bình tĩnh”, chấp nhận đi lùi để tìm đường tốt hơn.

Qua bài này, em học được cách đánh giá không chỉ kết quả cuối cùng mà còn cả quá trình hội tụ – điều này giúp em hiểu sâu hơn về sự khác nhau giữa “tốc độ” và “chất lượng lời giải”.  
Em cũng thấy rõ rằng trong thực tế, một thuật toán nhanh chưa chắc là tốt nhất, mà quan trọng là phải cân bằng giữa tốc độ và độ chính xác phù hợp với bài toán.