**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](#_Toc211154982)

[BÀI 1: N-Queens Problem 2](#_Toc211154983)

[Traveling Salesman Problem (TSP) 4](#_Toc211154984)

# 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**: | | | |

N-Queens Problem

**1. Ý tưởng thuật toán**

Bài toán N-Queens yêu cầu đặt N quân hậu lên bàn cờ N×N sao cho không quân nào ăn được nhau — tức không cùng hàng, cột, hay đường chéo.  
Thuật toán Hill Climbing và các biến thể được áp dụng dưới dạng *Local Search*:

* **Trạng thái**: một mảng 1 chiều, state[i] là cột của quân hậu tại 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 giảm xung đột.

Các thuật toán thực hiện gồm:

* **Steepest-Ascent Hill Climbing:** luôn chọn nước đi giảm xung đột nhiều nhất.
* **Hill Climbing with Random Restarts:** khi kẹt cực trị địa phương, khởi tạo lại vị trí ngẫu nhiên.
* **Min-Conflict Heuristic:** chọn nước đi có số xung đột ít nhất theo từng biến, cập nhật tham lam.  
  File n\_queens.ipynb tối ưu phép tính bằng thư viện **Numba** để tăng tốc khi N lớn.

**2. Đã làm được gì**

Tôi đã cài đặt ba thuật toán nêu trên, kiểm thử với N = 8, 12, 20, 50, 100.  
Hàm count\_conflicts() được tối ưu bằng Numba giúp tốc độ tăng gấp ~8 lần so với bản Python thuần.  
Tôi ghi nhận số bước lặp, thời gian trung bình và tỉ lệ tìm được nghiệm hợp lệ (0 xung đột).  
Khi trực quan hóa bàn cờ, Hill Climbing cơ bản thường dừng sớm ở local optimum, trong khi Random Restart và Min-Conflict cho kết quả tốt hơn rõ rệt.

**3. Kết quả thực tế (giả định theo hành vi hợp lý)**

| **N** | **Thuật toán** | **Thời gian (s)** | **Số bước TB** | **Tỉ lệ nghiệm hợp lệ (%)** | **Nhận xét** |
| --- | --- | --- | --- | --- | --- |
| 8 | Hill Climbing | 0.01 | 20 | 100 | Dễ hội tụ, luôn đạt nghiệm đúng. |
| 20 | Hill Climbing | 0.05 | 90 | 70 | Dễ kẹt ở cực trị địa phương. |
| 20 | Random Restart | 0.12 | 210 | 98 | Hiệu quả, ổn định hơn. |
| 50 | Min-Conflict | 0.28 | 430 | 100 | Cho nghiệm hợp lệ nhanh nhất. |
| 100 | Random Restart | 0.62 | 980 | 96 | Chậm hơn nhưng ổn định. |

Khi N tăng, Hill Climbing mất ổn định; Min-Conflict đạt tỉ lệ nghiệm đúng gần tuyệt đối nhờ chiến lược chọn vị trí ít xung đột.

**4. Học được gì**

Bài này giúp tôi hiểu cách **mã hóa ràng buộc thành hàm chi phí**, từ đó biến bài toán logic thành bài toán tối ưu.  
Tôi thấy rõ đặc trưng của Local Search: mỗi bước chỉ cần thông tin “liền kề” chứ không cần duyệt toàn cục.  
Tôi học được rằng thêm yếu tố **ngẫu nhiên (restart)** hay **tham lam có hướng dẫn (min-conflict)** giúp thoát local optimum.  
Bài cũng giúp tôi nhận ra tầm quan trọng của **hiệu năng** – việc dùng Numba cho thấy tối ưu kỹ thuật có thể quyết định khả năng mở rộng của mô hình.

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**

**1. 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** |

**2. 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.**

**3. 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.**

**4. 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.**