**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

Logo, company name

Description automatically generated

**BÀI TẬP LỚN**

**TÊN HỌC PHẦN: TRÍ TUỆ NHÂN TẠO**

**ĐỀ TÀI: ỨNG DỤNG GIẢI THUẬT DI TRUYỀN TRONG TỐI ƯU SẮP XẾP CHỖ NGỒI TIỆC CƯỚI**

**Giáo viên hướng dẫn: ThS: Trần Đăng Công**

**Sinh viên thực hiện:**

|  |  |  |  |
| --- | --- | --- | --- |
| STT | **Mã sv** | **Họ và tên** | **Lớp** |
| **1** | **1771020169** | Phùng Mạnh Đức | **CNTT 17-14** |
| **2** | 1771020740 | Hồ Thị Ngọc Vi | **CNTT 17-14** |
| **3** | **1771020558** | Phạm Ngọc Quân | **CNTT 17-14** |

**Hà Nội, năm 2025**

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

Logo, company name

Description automatically generated

**BÀI TẬP LỚN**

**TÊN HỌC PHẦN: TRÍ TUỆ NHÂN TẠO**

**ĐỀ TÀI:** **ỨNG DỤNG GIẢI THUẬT DI TRUYỀN TRONG TỐI ƯU SẮP XẾP CHỖ NGỒI TIỆC CƯỚI**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Mã Sinh Viên | Họ và Tên | Ngày Sinh | Điểm | |
| Bằng Số | Bằng Chữ |
| 1 | **1771020169** | Phùng Mạnh Đức | 24/08/2004 |  |  |
| 2 | 1771020740 | Hồ Thị Ngọc Vi | 28/07/2005 |  |  |
| 3 | **1771020558** | Phạm Ngọc Quân | 01/06/2002 |  |  |

### CÁN BỘ CHẤM THI

**ThS: Trần Đăng Công**

**Hà Nội, năm 2025**

**LỜI NÓI ĐẦU**

Trí tuệ nhân tạo (Artificial Intelligence - AI) là một trong những lĩnh vực khoa học và công nghệ phát triển mạnh mẽ nhất trong thời đại hiện nay. Với khả năng mô phỏng tư duy con người, AI không chỉ giúp tự động hóa các quy trình phức tạp mà còn mở ra nhiều cơ hội mới trong các lĩnh vực như y tế, tài chính, giao thông, giáo dục và sản xuất công nghiệp. AI không chỉ giới hạn ở các hệ thống máy học đơn giản mà còn tiến xa hơn với học sâu (Deep Learning), thị giác máy tính (Computer Vision), xử lý ngôn ngữ tự nhiên (NLP), giúp cải thiện hiệu suất và tối ưu hóa nhiều khía cạnh của cuộc sống.

Môn học Trí tuệ nhân tạo cung cấp nền tảng kiến thức quan trọng về các thuật toán, mô hình AI cũng như phương pháp xây dựng các hệ thống thông minh. Trong quá trình học tập và nghiên cứu, sinh viên sẽ được tiếp cận với các chủ đề như tìm kiếm thông minh, logic mệnh đề, mạng nơ-ron nhân tạo, thuật toán di truyền và nhiều kỹ thuật tiên tiến khác. Những kiến thức này không chỉ giúp hiểu rõ nguyên lý hoạt động của AI mà còn hỗ trợ trong việc ứng dụng AI vào thực tiễn, giải quyết các bài toán tối ưu và ra quyết định.

Với sự phát triển mạnh mẽ của công nghệ, AI ngày càng có tác động lớn đến xã hội và nền kinh tế. Do đó, việc nghiên cứu và ứng dụng trí tuệ nhân tạo là một yêu cầu tất yếu đối với những ai muốn làm chủ công nghệ trong thời đại 4.0. Hy vọng rằng môn học này sẽ giúp sinh viên có cái nhìn toàn diện hơn về AI, từ lý thuyết đến thực hành, và mở ra nhiều cơ hội phát triển trong tương lai.**MỤC LỤC**

[LỜI NÓI ĐẦU 3](#_Toc27790)

[MỤC LỤC 3](#_Toc15741)

[MỤC LỤC HÌNH ẢNH 7](#_Toc9152)

[MỤC LỤC BẢNG 8](#_Toc969)

[BẢNG CÁC TỪ VIẾT TẮT 9](#_Toc23462)

[CHƯƠNG I. GIỚI THỆU BÀI TOÁN 10](#_Toc25582)

[1.1. Giới thiệu bài toán 10](#_Toc21036)

[1.2. Thuật toán di truyền 10](#_Toc3615)

*[1.2.1. Khái niệm](#_Toc12912)* [10](#_Toc12912)

*[1.2.2. Nguyên lý hoạt động của thuật toán di truyền](#_Toc14452)* [11](#_Toc14452)

[CHƯƠNG II. LÝ THUYẾT GIẢI THUẬT DI TRUYỀN 12](#_Toc6960)

[2.1. Khái niệm 12](#_Toc27537)

[2.2. Các tính chất của giải thuật di truyền 12](#_Toc19424)

[2.3. Các bước cơ bản 13](#_Toc17680)

[2.4. Nguyên lý hoạt động 14](#_Toc8152)

[2.6. Ứng dụng của GAs 16](#_Toc15976)

[CHƯƠNG III. ỨNG DỤNG GIẢI THUẬT DI TRUYỀN TRONG TỐI ƯU SẮP XẾP CHỖ NGỒI TIỆC CƯỚI 20](#_Toc4141)

[3.1. Mục tiêu 20](#_Toc9198)

[3.2. Các bước thực hiện và kết quả 20](#_Toc22830)

[3.2.1. Thư viện import 20](#_Toc1239)

*[3.2.3. Tham số GA](#_Toc27758)* [21](#_Toc27758)

*[3.2.4. Hàm khởi tạo quần thể](#_Toc30172)* [21](#_Toc30172)

*[3.2.5. Lai ghép](#_Toc23009)* [22](#_Toc23009)

*[3.2.6. Đột biến](#_Toc5928)* [25](#_Toc5928)

*[3.2.7. Giải thuật di truyền](#_Toc11519)* [26](#_Toc11519)

*[3.2.8. Hàm tính điểm thân thiết](#_Toc9575)* [27](#_Toc9575)

*[3.2.9. Hiển thị kết quả](#_Toc26010)* [29](#_Toc26010)

[KẾT LUẬN 31](#_Toc25390)

[DANH MỤC TÀI LIỆU THAM KHẢO 32](#_Toc16541)

**MỤC LỤC HÌNH ẢNH**

[Hình: 2.3. Các bước cơ bản 11](#_Toc12133)

[Hình: 3.2.1. Tên thư viện 18](#_Toc28207)

[Hình :3.2.2.Danh sách khách mời và điểm thân thiết 19](#_Toc3551)

[Hình:3.2.3.Tham số GA 19](#_Toc19157)

**MỤC LỤC BẢNG**

**BẢNG CÁC TỪ VIẾT TẮT**

|  |  |  |
| --- | --- | --- |
| **STT** | **TỪ VIẾT TẮT** | **VIẾT ĐẦY ĐỦ** |
| **1** | **CSDL** | **Cơ sở dữ liệu** |
| **2** |  |  |

**CHƯƠNG I. GIỚI THỆU BÀI TOÁN**

**1.1. Giới thiệu bài toán**

Trong các sự kiện quan trọng như tiệc cưới, việc sắp xếp chỗ ngồi hợp lý là một vấn đề quan trọng để đảm bảo sự thoải mái, hài hòa giữa khách mời. Một sơ đồ chỗ ngồi hợp lý cần phải tối ưu hóa mức độ thân thiết giữa các khách, giúp họ có trải nghiệm tốt hơn và tránh những sự sắp xếp không phù hợp. Tuy nhiên, khi số lượng khách lớn, số bàn nhiều, thì việc tìm ra cách xếp chỗ tốt nhất trở nên rất phức tạp, đòi hỏi một phương pháp tối ưu hóa hiệu quả.

Bài toán này có thể được mô hình hóa như một bài toán tối ưu tổ hợp, trong đó mỗi cách sắp xếp khách mời tương ứng với một cấu hình (hoặc cá thể), và mục tiêu là tìm ra cấu hình tối ưu nhất sao cho tổng điểm thân thiết giữa các khách mời cao nhất. Do số lượng các cách sắp xếp chỗ ngồi là quá lớn, không thể duyệt hết bằng phương pháp thông thường, ta cần áp dụng các thuật toán thông minh để giải quyết bài toán. Một trong những phương pháp hiệu quả là giải thuật di truyền (Genetic Algorithm - GA), giúp tìm kiếm nghiệm tốt trong không gian tìm kiếm rộng lớn.

**1.2. Thuật toán di truyền**

***1.2.1. Khái niệm***

Thuật toán di truyền (Genetic Algorithm - GA) là một thuật toán tìm kiếm và tối ưu hóa dựa trên quá trình tiến hóa tự nhiên của Charles Darwin. GA mô phỏng các cơ chế chọn lọc tự nhiên, lai ghép và đột biến, giúp tìm ra lời giải tối ưu mà không cần phải kiểm tra toàn bộ không gian tìm kiếm.

GA đặc biệt hiệu quả đối với các bài toán tối ưu hóa tổ hợp, nơi mà số lượng nghiệm có thể quá lớn để giải quyết bằng phương pháp duyệt toàn bộ (Brute-force). Thay vì thử tất cả các cách xếp chỗ, GA sử dụng một tập hợp nghiệm (quần thể) và dần dần cải thiện chúng qua từng thế hệ, giúp tìm ra một sơ đồ chỗ ngồi tốt nhất mà không mất quá nhiều thời gian tính toán.

***1.2.2. Nguyên lý hoạt động của thuật toán di truyền***

GA hoạt động dựa trên các thành phần chính sau:

* Khởi tạo quần thể (Initial Population)
* Đánh giá năng lực (Evaluation fitness)
* Chọn lọc tự nhiên (Selection)
* Lai ghép (Crossover)
* Đột biến (Mutation)
* Lặp lại

**CHƯƠNG II. LÝ THUYẾT GIẢI THUẬT DI TRUYỀN**

**2.1. Khái niệm**

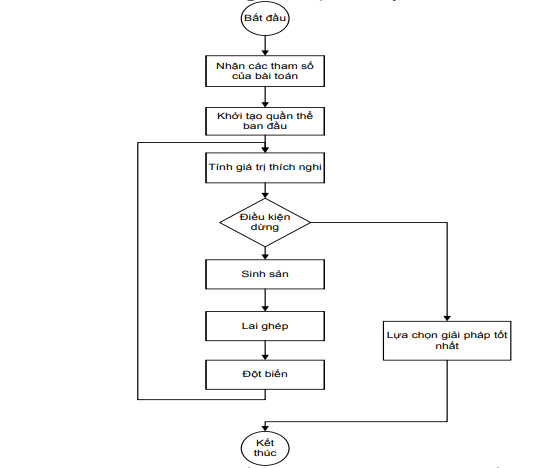
GAs là một kỹ thuật của khoa học máy tính nhằm tìm kiếm giải pháp thích hợp cho các bài toán tối ưu tổ hợp (combinatorial optimization), là một phân ngành của giải thuật tiến hóa, vận dụng các nguyên lý của tiến hóa như: di truyền, đột biến, chọn lọc tự nhiên, và trao đổi chéo. Nó sử dụng ngôn ngữ máy tính để mô phỏng quá trình tiến hoá của một tập hợp những đại diện trừu tượng (gọi là những nhiễm sắc thể), của các giải pháp có thể (gọi là những cá thể) cho bài toán tối ưu hóa vấn đề. Tập hợp này sẽ tiến triển theo hướng chọn lọc những giải pháp tốt hơn. GAs cũng như các thuật toán tiến hoá, đều được hình thành dựa trên một quan niệm được coi là một tiên đề phù hợp với thực tế khách quan. Đó là quan niệm "Quá trình tiến hoá tự nhiên là quá trình hoàn hảo nhất, hợp lý nhất và tự nó đã mang tính tối ưu". Quá trình tiến hoá thể hiện tính tối ưu ở chỗ thế hệ sau bao giờ cũng tốt hơn thế hệ trước.

Ngày nay, GAs càng trở nên quan trọng, đặc biệt là trong lĩnh vực tối ưu hoá, một lĩnh vực có nhiều bài toán thú vị, được ứng dụng nhiều trong thực tiễn nhưng thường khó và chưa có phương pháp hiệu quả để giải quyết

**2.2. Các tính chất của giải thuật di truyền**

GAs là kỹ thuật chung, giúp giải quyết vấn đề bằng cách mô phỏng sự tiến hóa của con người hay của sinh vật nói chung (dựa trên thuyết tiến hóa muôn loài của Darwin), trong điều kiện qui định sẵn của môi trường. Mục tiêu của GAs không nhằm đưa ra lời giải chính xác tối ưu mà là đưa ra lời giải tương đối tối ưu. Một cá thể trong GAs sẽ biểu diễn một giải pháp của bài toán. Tuy nhiên, không giống với trong tự nhiên là một cá thể có nhiều nhiễm sắc thể (NST) mà để giới hạn trong GAs, ta quan niệm một cá thể có một NST. Do đó, khái niệm cá thể và NST trong GAs coi như là tương đương. Một NST được tạo thành từ nhiều gen, mỗi gen có thể có các giá trị khác nhau để quy định một tình trạng nào đó. Trong GAs, một gen được coi như một phần tử trong chuỗi NST. Một tập hợp các cá thể có cùng một số đặc điểm nào đấy được gọi là quần thể. Trong thuật giải di truyền, ta quan niệm quần thể là một tập các lời giải của một bài toán.

**2.3. Các bước cơ bản**



Hình: 2.3. Các bước cơ bản

[Bắt đầu ] Nhận các tham số cho thuật toán.

[Khởi tạo] Sinh ngẫu nhiên một quần thể gồm n cá thể ( là n lời giải cho bài toán)

[Quần thể mới ] Tạo quần thể mới bằng cách lặp lại các bước sau cho đến khi quần thể mới hoàn thành

[Thích nghi] Ước lượng độ thích nghi eval(x) của mỗi cá thể.

[Kiểm tra ] Kiểm tra điều kiện kết thúc giải thuật.

[Chọn lọc] Chọn hai cá thể bố mẹ từ quần thể cũ theo độ thích nghi của chúng (cá thể có độ thích nghi càng cao thì càng có nhiều khả năng được chọn)

[Lai ghép] Với một xác suất lai ghép được chọn, lai ghép hai cá thể bố mẹ để tạo ra một cá thể mới.

[Đột biến] Với một xác suất đột biến được chọn, biến đổi cá thể mới

[Chọn kết quả] Nếu điều kiện dừng được thỏa mãn thì thuật toán kết thúc và trả về lời giải tốt nhất trong quần thể hiện tại

GAs có hai loại điều kiện dừng cơ bản (1) dựa trên cấu trúc nhiễm sắc thể, kiểm soát số gen được hội tụ, nếu số gen hội tụ vượt quá số phần trăm nào đó của tổng số gen, việc tìm kiếm sẽ kết thúc; (2) dựa trên ý nghĩa đặc biệt của một nhiễm sắc thể, đo tiến bộ của giải thuật trong một số thế hệ cho trước, nếu tiến bộ này nhỏ hơn một hằng số ε xác định, kết thúc tìm kiếm

**2.4. Nguyên lý hoạt động**

Nền tảng lý thuyết của GAs dựa trên biểu diễn chuỗi nhị phân và lý thuyết sơ đồ. Một sơ đồ là một chuỗi, dài bằng chuỗi nhiễm sắc thể, các thành phần của nó có thể nhận một trong các giá trị của tập ký tự biểu diễn gen hoặc một ký tự đại diện “\*”. Sơ đồ biểu diễn một không gian con của không gian tìm kiếm. Không gian con này là tập tất cả các chuỗi trong không gian lời giải mà với mọi vị trí trong chuỗi giá trị của gen trùng với giá trị của sơ đồ

Ví dụ: các chuỗi và sơ đồ có chiều dài 10.

Sơ đồ (\*111100100) sẽ khớp với hai chuỗi:

*{(0111100100), (1111100100)}*

Và sơ đồ (\*1\*1100100) sẽ khớp với 4 chuỗi

*{(0101100100), (0111100100), (1101100100), (1111100100)}*

Đương nhiên, sơ đồ (1001110001) chỉ khớp với chính nó, và sơ đồ (\*\*\*\*\*\*\*\*\*\*) khớp với tất cả các chuỗi có chiều dài 10. Rõ ràng là mỗi sơ đồ cụ thể có tương ứng 2r chuỗi, với r là số ký tự đại diện ‘\*’ có trong sơ đồ. Mặc khác, mỗi chuỗi chiều dài m sẽ khớp với 2m sơ đồ.

Một chuỗi chiều dài m, sẽ có tối đa 2m sơ đồ. Trong một quần thể kích thước n, có thể có tương ứng từ 2 m đến nx2m sơ đồ khác nhau.

Các sơ đồ khác nhau có những đặc trưng khác nhau. Các đặc trưng này thể hiện qua hai thuộc tính quan trọng bậc và chiều dài xác định.

Bậc của sơ đồ S (ký hiệu σ(S)) là chiều dài của chuỗi trừ đi số ký tự đại diện. Bậc xác định đặc trưng của sơ đồ.

Ví dụ: ba sơ đồ chiều dài 10

*S1=(\*\*\*001\*110)*

*S2=(\*\*\*\*00\*\*0\*)*

*S3=(11101\*\*001)*

Có bậc tương ứng:

*σ(S1)=6; σ(S2)=3; σ(S3)=8*

Khái niệm bậc của sơ đồ giúp cho việc tính xác suất sống còn của sơ đồ do ảnh hưởng của đột biến.

Chiều dài xác định của sơ đồ S (ký hiệu là δ(S)) là khoảng cách giữa hai vị trí cố định ở đầu và cuối. Nó định nghĩa “độ nén” của thông tin chứa trong một sơ đồ.

Ví dụ:

*δ(S1)=10-4=6; δ(S2)=9-5=4; δ(S3)=10-1=9*

Như vậy, một sơ đồ chỉ có một vị trí cố định duy nhất thì sẽ có chiều dài xác định là 0

Khái niệm chiều dài xác định của sơ đồ giúp tính xác suất sống còn của sơ đồ do ảnh hưởng của phép lai.

GAs sử dụng một quần thể của các lời giải có thể. Mỗi lời giải được đại diện bởi một NST, nó chỉ là một đại diện trừu tượng. Các NST được mã hóa thành các chuỗi nhị phân, mỗi vị trí trên chuỗi tồn tại hai giá trị là “1” hoặc “0”. Chẳng hạn như: *1 0 0 1 0 1 0 1 1 0*.

Độ tốt của một cá thể được đánh giá bằng hàm mục tiêu g(x) với x là một NST. Hàm mục tiêu g(x) sau khi được tính toán sẽ là cơ sở để đánh giá độ thích nghi của cá thể. Hàm thích nghi f(x) là sẽ quyết định khả năng một cá thể được chọn lọc vào thế hệ sau, việc ánh xạ g(x)→ f(x) có nhiều phương pháp ánh xạ khác nhau phụ thuộc vào mục đích của bài toán.

**2.5. So sánh GAs với kỹ thuật tối ưu khác**

Hoạt động của GAs đơn giản là việc mô phỏng sự tiến hóa và chọn lọc tự nhiên bằng máy tính bắt đầu từ một quần thể ngẫu nhiên. Bên cạnh đó để tối ưu ta cần hàm lượng giá hoặc hàm thích nghi để chọn cá thể tốt và loại bỏ cá thể xấu.

Thuật toán di truyền (GAs) khác với kĩ thuật tối ưu khác ở chỗ:

- GAs làm việc với bộ mã của biến chứ không phải làm việc trực tiếp trên biến.

- Hầu hết các kĩ thuật tối ưu thông thường tìm kiếm từ một đỉnh, trong khi đó GAs luôn hoạt động trên tập hợp đỉnh (điểm tối ưu), điều này là một ưu điểm của GAs giúp tăng cơ hội tiếp cận tối ưu toàn cục và tránh hội tụ sớm tại điểm cục bộ địa phương.

- GAs đánh giá hàm mục tiêu để phục vụ quá trình tìm kiếm, vì vậy có thể ứng dụng cho bất kì bài toán tối ưu nào (liên tục hay rời rạc).

- GAs thuộc lớp các thuật toán xác suất, các thao tác cơ bản của GAs dựa trên khả năng tích hợp ngẫu nhiên trong quá trình xử lý.

**2.6. Ứng dụng của GAs**

GAs được sử dụng cho những bài toán khó, và đã được ứng dụng thành công cho một số bài toán như: lập kế hoạch, điều khiển tương thích, chương trình trò chơi, các bài toán vận tải, bài toán người đi du lịch,…Sau đây là một vài ứng dụng tiêu biểu của GAs

* Bài toán người du lịch (TSP)

TSP được mô tả như sau: Một du khách muốn thăm những thành phố anh quan tâm; mỗi thành phố thăm qua đúng một lần; rồi trở về điểm khởi hành. Biết trước chi phí di chuyển giữa hai thành phố bất kỳ. Yêu cầu của bài toán là xây dựng một lộ trình thỏa các điều kiện trên với tổng chi phí nhỏ nhất.

TSP là bài toán tối ưu tổ hợp, không gian tìm kiếm là tập các hoán vị của n thành phố. Bất cứ hoán vị nào của n thành phố cũng là một lời giải chấp nhận được. Lời giải tối ưu là một hoán vị với chi phí tối thiểu của hành trình. Không gian tìm kiếm là n!. Có thể giải bài toán này bằng nhiều phương pháp: phương pháp nhánh cận, phương pháp gần đúng hay những phương pháp tìm kiếm heuristic. Phương pháp nhánh cận đã được chứng minh đạt sự tối ưu về lời giải, tuy nhiên phương pháp này lại mất khá nhiều thời gian khi số đỉnh của đồ thị lớn. Trong những năm gần đây, đã xuất hiện nhiều thuật toán đạt gần đến lời giải tối ưu của bài toán TSP: láng giềng gần nhất, đảo gần nhất, đảo xa nhất…và TSP cũng trở thành một đích ngắm của cộng đồng GAs. Với bài toán này chúng ta sẽ đánh số các thành phố và dùng một vector nguyên để biểu diễn một NST lộ trình *v=<i1,i2,…,in>* biểu diễn một lộ trình: từ i1 đến i2…, từ in-1 đến in và trở về i1 (v là một hoán vị của vector *<1,2,…,n>*), hàm lượng giá chính là chi phí của lộ trình.

* Bài toán lập lịch

Lập lịch là bài toán tổ chức sản xuất. Một công ty cần sản xuất nhiều loại hàng hóa; những hàng hóa này có thể được sản xuất theo những kế hoạch khác nhau. Mỗi kế hoạch xử lý gồm một chuỗi thao tác; những thao tác này sử dụng một số tài nguyên và cần thời gian chạy máy. Một lịch sản xuất là một kế hoạch thực hiện các đơn đặt hàng. Trong đó, một số đơn đặt hàng có thể được thực hiện với cùng những thao tác tương đương. Nhiệm vụ là lên kế hoạch, lập lịch sản xuất, để thực hiện các đơn đặt hàng này nhanh nhất có thể.

Bài toán lập lịch là chọn một chuỗi các thao tác đồng thời chỉ định về thời gian bắt đầu/ kết thúc và các tài nguyên cần thiết cho mỗi thao tác. Điều cần quan tâm chính yếu là chi phí thời gian máy rỗi, năng lực lao động và các đơn đặt hàng cần hoàn thành đúng hạn.Ý tưởng chính trong phương pháp là mã hóa biểu diễn của lịch phân công là các toán tử di truyền phải thực hiện theo cách có ý nghĩa, và một bộ giải mã phải luôn tạo ra một lời giải hợp lệ cho bài toán. Thủ tục giải mã mô phỏng các thao tác của công việc theo cách mà khi một máy tính sẵn sàng chọn lựa, thì thao tác cho phép đầu tiên từ danh sách ưu tiên được lấy ra. Ví dụ nếu danh sách ưu tiên của máy m1 là: *m1(40 o3 o1 o2 ‘chờ’ ‘nhàn rỗi’)*, thì thủ tục giải mã vào thời điểm 40 có thể thực hiện đơn đặt hàng o3 trên máy m1. Nếu không được, thủ tục giải mã sẽ thực hiện đơn đặt hàng o1 và o2 (nghĩa là, tìm ở o1 trước; nếu không được mới tìm ở o2). Biểu diễn này bảo đảm tạo một lịch phân công hợp lệ.

* Lập thời khóa biểu cho trường học

Bài toán thời khóa biểu là một bài toán kết hợp nhiều ràng buộc không tầm thường thuộc nhiều loại. Có nhiều phiên bản của bài toán thời khóa biểu, một trong những bài toán này có thể được mô tả như sau: Có một danh sách các giáo viên, một danh sách các khoảng thời gian, một danh sách các lớp. Bài toán cần tìm thời khóa biểu tối ưu (giáo viên – thời gian – lớp); hàm mục tiêu phải thỏa những mục tiêu này (các ràng buộc mềm) gồm: Có một số giờ được xác định trước cho mỗi giáo viên và mỗi lớp; Chỉ một giáo viên trong một lớp vào một giờ nhất định; Một giáo viên không thể dạy hai lớp cùng lúc; Đối với mỗi lớp được xếp thời khóa biểu vào một khoảng thời gian, phải có một giáo viên…Ngoài ra còn có các mục tiêu sư phạm như trải một số lớp ra nguyên tuần, những mục tiêu thuộc cá nhân như những giáo viên hợp đồng không phải dạy buổi chiều, và các mục tiêu về tổ chức như mỗi giờ có một giáo viên bổ sung sẵn sàng chỗ dạy tạm thời.

Biểu diễn nhiễm sắc thể tự nhiên nhất cho bài toán này là biểu diễn ma trận: một ma trận *(R)ij (1 <= i <= mvà 1 <= j <= n )*, ở đây mỗi hàng tương ứng với một giáo viên, mỗi cột tương ứng với một giờ, các phần tử của ma trận R là các lớp *(rij ∈ {C1,...Ck})*. Các ràng buộc chủ yếu được xử lý bởi các toán tử di truyền và thuật giải sửa chữa được sử dụng để loại bỏ những trường hợp mà có nhiều hơn một giáo viên xuất hiện trong cùng một lớp vào cùng một giờ.

* Phân hoạch đối tượng và đồ thị

Bài toán phân hoạch là cần chia n đối tượng thành k loại. Lớp bài toán này gồm nhiều bài toán nổi tiếng như bài toán đóng thùng (gán các mặt hàng vào các thùng), bài toán tô màu đồ thị (gán các nút của đồ thị vào các màu cụ thể…).

Bài toán đóng thùng (Bin Packing – BP) được phát biểu như sau: Cho danh sách gồm n đồ vật L=a1,a2,a3,…,an và các thùng giống nhau với cùng sức chứa B. Kích thước của đồ vật ai là si thỏa mãn *0 < sj <= B ∇ i = 1,2,...n*. Vấn đề đặt ra là tìm cách xếp các đồ vật vào các thùng sao cho số lượng thùng cần phải sử dụng là ít nhất.

* Vạch đường cho robot di chuyển

Tìm đường là hướng dẫn robot di chuyển đến đích mà không bị lạc hay va vào những đối tượng khác. Trong bài toán này, một lộ trình được lập trước để robot đi theo, lộ trình này có thể dẫn dắt robot đi tới đích một cách hoàn hảo. Tuy nhiên, các nhà khoa học muốn cải tiến hơn bằng cách vạch lộ trình nội tại, phụ thuộc vào tri thức thu được từ việc cảm nhận môi trường cục bộ để xử lý các chướng ngại chưa biết.

Bộ tìm đường tiến hóa (EN) được đề xuất. Phần đầu của thuật giải là tìm lộ trình toàn cục tối ưu từ điểm khởi đầu đến đến đích, phần thứ hai có trách nhiệm xử lý những va chạm có thể xảy ra hay những vật cản chưa được biết trước bằng cách thay một phần của lộ trình toàn cục gốc bằng một lộ trình con tối ưu.

* Bài toán Robot sơn phòng

Bài toán yêu cầu thiết kế một giải thuật di truyền để tối ưu hóa đường đi của một robot phun sơn trong một căn phòng hình chữ nhật có kích thước 20x40. Căn phòng được biểu diễn dưới dạng một ma trận 2D, ban đầu tất cả các ô có giá trị 0, nghĩa là chưa được sơn. Khi robot sơn một ô, giá trị của ô đó chuyển thành 1. Robot có thể di chuyển theo bốn hướng: lên, xuống, trái, phải, và có thể thực hiện các hành động như tiến lên, quay trái, quay phải hoặc quay ngẫu nhiên. Đặc biệt, nếu robot đi qua cạnh của phòng, nó sẽ xuất hiện ở cạnh đối diện, nghĩa là phòng có tính chất liên tục theo chu vi. Mục tiêu là giúp robot sơn toàn bộ phòng một cách hiệu quả nhất, tức là sơn được nhiều ô nhất trong thời gian ngắn nhất, tránh di chuyển thừa hoặc sơn trùng một khu vực nhiều lần. Giải thuật di truyền sẽ được triển khai bằng Python, sử dụng thư viện NumPy để xử lý ma trận và đánh giá độ hiệu quả của từng giải pháp. Trong quá trình thực thi, thuật toán sẽ trải qua các bước như khởi tạo quần thể các giải pháp ngẫu nhiên, đánh giá độ thích nghi dựa trên hiệu suất sơn phòng, chọn lọc các cá thể tốt nhất, thực hiện lai ghép để tạo ra thế hệ mới và đột biến để tăng tính đa dạng. Cuối cùng, giải thuật sẽ được kiểm tra trên một tập hợp các phòng được định nghĩa trước nhằm tìm ra giải pháp tối ưu giúp robot hoàn thành nhiệm vụ nhanh nhất và hiệu quả nhất.

**CHƯƠNG III. ỨNG DỤNG GIẢI THUẬT DI TRUYỀN TRONG TỐI ƯU SẮP XẾP CHỖ NGỒI TIỆC CƯỚI**

**3.1. Mục tiêu**

Sử dụng thuật toán di truyền (Genetic Algorithm - GA) để tìm ra sơ đồ chỗ ngồi tối ưu cho khách mời tại một sự kiện.

**Mục tiêu tối ưu hóa**: Xếp chỗ sao cho **những người có mối quan hệ thân thiết được ngồi gần nhau**, tối đa hóa tổng điểm thân thiết.

Các bước chính:

* Khởi tạo danh sách khách mời và điểm thân thiết giữa các nhóm khách.
* Tạo quần thể ban đầu gồm nhiều sơ đồ chỗ ngồi ngẫu nhiên.
* Tính điểm thân thiết của từng sơ đồ.
* Lai ghép giữa các sơ đồ chỗ ngồi tốt nhất.
* Áp dụng đột biến để tăng đa dạng.
* Lặp lại qua nhiều thế hệ để tìm sơ đồ tốt nhất.

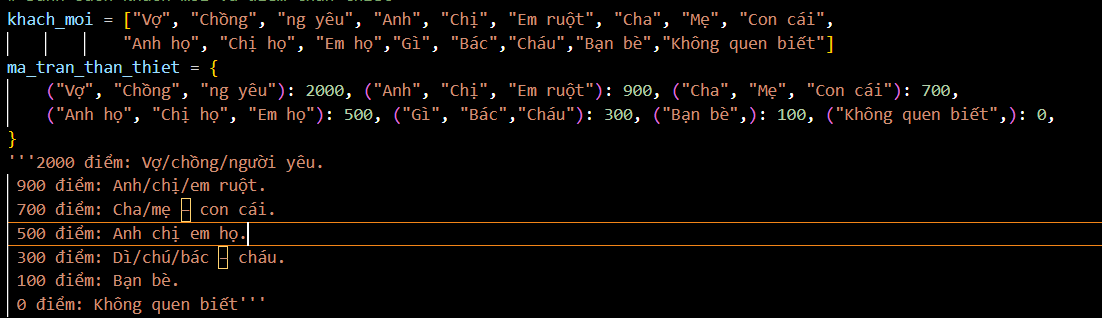
**3.2. Các bước thực hiện và kết quả**

**3.2.1. Thư viện import**



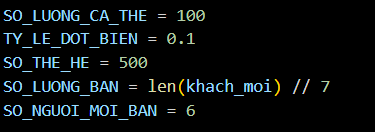
Hình: 3.2.1. Tên thư viện

***3.2.2. Danh sách khách mời và điểm thân thiết***



Hình :3.2.2.Danh sách khách mời và điểm thân thiết

***3.2.3. Tham số GA***



Hình:3.2.3.Tham số GA

***3.2.4. Hàm khởi tạo quần thể***

def tao\_quan\_the():

    """Sinh quần thể ban đầu với mỗi khách chỉ xuất hiện ở một bàn duy nhất."""

    quan\_the = []

    for \_ in range(SO\_LUONG\_CA\_THE):

        khach\_xao\_tron = random.sample(khach\_moi, len(khach\_moi))  # Xáo trộn danh sách khách mời

        so\_do = [khach\_xao\_tron[i:i + SO\_NGUOI\_MOI\_BAN] for i in range(0, len(khach\_xao\_tron), SO\_NGUOI\_MOI\_BAN)]

        quan\_the.append(so\_do)

    return quan\_the

* Khởi tạo danh sách chứa quần thể:

quan\_the = []

* Tiếp theo dùng vòng lặp for để tạo số lượng cá thể theo yêu cầu

For in range(SO\_LUONG\_CA\_THE):

* Sau khi đã có số lượng cá thể và danh sách quần thể thì random các quần thể đó vào vị trí chỗ ngồi

khach\_xao\_tron = random.sample(khach\_moi, len(khach\_moi))

So\_do = [khach\_xao\_tron[I:I + SO\_NGUOI\_MOI\_BAN] for in rang(0, len(khach\_xao\_tron), SO\_KHACH\_MOI\_BAN)]

* Tiếp đến là thêm sơ đồ vào quần thể

quan\_the.append(so\_do)

***3.2.5. Lai ghép***

def lai\_ghep(cha, me):

    """ Lai ghép hai bố mẹ để tạo con, đảm bảo không trùng lặp khách mời. """

    diem\_cat = random.randint(1, SO\_LUONG\_BAN )

    con = cha[:diem\_cat] + me[diem\_cat:]

    # Đảm bảo không có khách nào xuất hiện hai lần

    khach\_da\_co = set()

    so\_do\_moi = []

    for ban in con:

        ban\_moi = [khach for khach in ban if khach not in khach\_da\_co]

        khach\_da\_co.update(ban\_moi)

        so\_do\_moi.append(ban\_moi)

    # Thêm khách còn thiếu từ danh sách gốc

    khach\_thieu = set(khach\_moi) - khach\_da\_co

    for ban in so\_do\_moi:

        while len(ban) < SO\_NGUOI\_MOI\_BAN and khach\_thieu:

            ban.append(khach\_thieu.pop())

    return so\_do\_moi

* Khai báo hàm: def lai\_ghep(cha, me):
* Chọn điểm cắt:

Điểm cắt (diem\_cat) là vị trí được quyết định số bàn được lấy từ cha và mẹ

Chọn ngẫu nhiên từ 1 đến SO\_LUONG\_BAN (trong sơ đồ)

diem\_cat = random.randint(1, SO\_LUONG\_BAN)

* Tạo con bằng cách ghép 2 phần của cha và mẹ

Cắt cha từ đầu đến diem\_cat

Cắt mẹ từ diem\_cat trở đi

Ghép 2 phần lại để tạo con

Con = cha[:diem\_cat] + me[diem\_cat:]

* Loại bỏ những khách mời trùng lặp

Dùng tập hợp (set) khach\_da\_co để lưu những khách đã được thêm

Lọc lại từng bàn, chỉ giữ những khách chưa xuất hiện

Cập nhật khach\_da\_co để tránh trùng lặp

khach\_da\_co = set()

so\_do\_moi = []

    for ban in con:

        ban\_moi = [khach for khach in ban if khach not in khach\_da\_co]

        khach\_da\_co.update(ban\_moi)

        so\_do\_moi.append(ban\_moi)

* Thêm khách mời còn thiếu

Tìm những khách còn thiếu bằng phép trừ tập hợp

*khach\_thieu = set(khach\_moi) - khach\_da\_co*

Sau khi xem được những khách còn thiếu thì ta sẽ thêm khách vào những bàn còn thiếu người

*for ban in so\_do\_moi:*

*while len(ban) < SO\_NGUOI\_MOI\_BAN and khach\_thieu:*

*ban.append(khach\_thieu.pop())*

* Trả về sơ đồ con đã hợp lệ

Sau khi đã thêm khách mời thì sẽ *return so\_do\_moi* 1 sơ đồ mới hợp lệ hơn

***3.2.6. Đột biến***

def dot\_bien(so\_do):

    """ Đột biến bằng cách hoán đổi ngẫu nhiên hai khách mời."""

    if random.random() < TY\_LE\_DOT\_BIEN:

        ban1, ban2 = random.sample(range(len(so\_do)), 2)

        if so\_do[ban1] and so\_do[ban2]:  # Kiểm tra bàn có khách không

            i, j = random.randint(0, len(so\_do[ban1]) - 1), random.randint(0, len(so\_do[ban2]) - 1)

            so\_do[ban1][i], so\_do[ban2][j] = so\_do[ban2][j], so\_do[ban1][i]

    return so\_do

* Đầu tiên là xác định xác suất xảy ra đột biến, nếu giá trị sinh ra nhỏ hơn TY\_LE\_DOT\_BIEN thì sẽ thực hiện đột biến được. Nếu ko thì giữ nguyên sơ đồ và trả về không có thay đổi

*if random.random() < TY\_LE\_DOT\_BIEN:*

* Tiếp là chọn 2 bàn ngẫu nhiên để hoán đổi khách

*ban1, ban2 = random.sample(range(len(so\_do)), 2)*

* Kiểm tra xem bàn đó có khách không để tránh lỗi trường hợp có 1 bàn rỗng. Nếu sơ đồ có bàn lỗi thì câu lệnh if sẽ ngăn lỗi

*if so\_do[ban1] and so\_do[ban2]:*

* Chọn ngẫu nhiên một khách ở mỗi bàn để hoán đổi

*I,j=random.randint(0,len(so\_do[ban1])-1),random.randint(0,len(so\_do[ban2]))*

* Sau đó là hoán đổi 2 khách mời

*so\_do[ban1][i], so\_do[ban2][j] = so\_do[ban2][j], so\_do[ban1][i]*

* Khi hoán đổi được hoàn tất thì trả về so\_do mới

***3.2.7. Giải thuật di truyền***

def giai\_thuat\_di\_truyen():

    """ Giải thuật di truyền để tối ưu sơ đồ chỗ ngồi."""

    quan\_the = tao\_quan\_the()

    for \_ in range(SO\_THE\_HE):

        quan\_the = sorted(quan\_the, key=tinh\_diem\_than\_thiet, reverse=True)  #tối ưu

        quan\_the\_moi = quan\_the[:10]

        while len(quan\_the\_moi) < SO\_LUONG\_CA\_THE:

            cha, me = random.choices(quan\_the[:30], k=2)

            con = lai\_ghep(cha, me)

            con = dot\_bien(con)

            quan\_the\_moi.append(con)

        quan\_the = quan\_the\_moi

    return sorted(quan\_the, key=tinh\_diem\_than\_thiet, reverse=True)[0]

* Sử dụng vòng lặp tiến hóa qua các thế hệ. Chúng ta sẽ phải chạy thuật toán di truyền trong SO\_THE\_HE để tìm sơ đồ tối ưu nhất. Nếu SO\_THE\_HE = 50 thì chúng ta phải chạy 50 lần để cải thiện sơ đồ

*for in range(SO\_THE\_HE):*

* Chọn lọc tự nhiên: Đầu tiên cần sắp xếp quần thể theo tổng điểm giảm dần. Sau đó ta dùng hàm tinh\_diem\_than\_thiet() để tính tổng điểm cho từng cá thể. Cá thể nào có điểm cao hơn sẽ được ưu tiên cho vào sinh sản

*quan\_the = sorted(quan\_the, key = tinh\_diem\_than\_thiet, reverse = True)*

* Duy trì các cá thể ưu tú: Đó là chọn ra 10 cá thể từ thế hệ trước để đảm bảo gen không bị mất và Elitism là giải pháp giúp giữ cá thể tốt nhất

*quan\_the\_moi = quan\_the[:10]*

* Lai ghép để tạo thế hệ mới: Chọn ra 2 cá thể mạnh nhất làm cha mẹ. Sau đó ta gọi lại hàm lai\_ghep(cha, me) để tạo ra 1 cá thể con. Gọi thêm hàm dot\_bien(con) để thực hiện đột biến. Cứ lặp di lặp lại cho đủ số lượng cá thể

w*hile len(quan\_the\_moi) < SO\_LUONG\_CA\_THE:*

*cha, me = random.choices(quan\_the[:30], k = 2)*

*con = lai\_ghep(cha, me)*

*con = dot\_bien(con)*

*quan\_the\_moi.append(con)*

* Trả về cá thể tối ưu nhất: Sau khi thực hiện lai ghép thì sẽ sắp xếp lại theo điểm thân thiết và trả về thế hệ mới tốt nhất cuối cùng

Return sorted(quan\_the, key = tinh\_diem\_than\_thiet, reverse = True)[0]

***3.2.8. Hàm tính điểm thân thiết***

def tinh\_diem\_than\_thiet(so\_do):

    """ Tính tổng điểm thân thiết cho sơ đồ chỗ ngồi."""

    diem = 0

    for ban in so\_do:

        for i in range(len(ban)):

            for j in range(i + 1, len(ban)):

                for nhom, diem\_nhom in ma\_tran\_than\_thiet.items():

                    if ban[i] in nhom and ban[j] in nhom:

                        diem += diem\_nhom

                        break  # Dừng kiểm tra sau khi tìm thấy nhóm phù hợp

    return diem

* Tham số đầu vào: *def tinh\_diem\_than\_thiet(so\_do):*

So\_do: 1 danh sách chứa các bàn, mỗi bàn là một danh sách khách mời

Ví dụ:

*so\_do = [*

*["Vợ", "Chồng", "ng yêu", "Anh", "Chị", "Bạn bè"],*

*["Cha", "Mẹ", "Con cái", "Gì", "Bác", "Em họ"],*

*["Anh họ", "Chị họ", "Cháu", "Không quen biết", "Bạn bè", "Em ruột"]*

*]*

* Biến lưu điểm tổng: *diem = 0* sẽ lưu lại điểm tổng thân thiết của sơ đồ chỗ ngồi
* Vòng lặp duyệt qua từng bàn: *for ban in so\_do:*

Dùng để duyệt qua từng bàn trong sơ đồ để tính điểm

* Duyệt qua từng cặp khách hàng trong bàn:

*for i in range(len(ban)):*

*for j in range(i + 1, len(ban)):*

Duyệt tất cả các cặp khách hàng trong bàn để kiểm tra mức độ thân thiết

Sử dụng 2 vòng lặp (i, j) để chọn từng cặp khách:

i chạy từ 0 đến *len(ban) - 1*

J chạy từ *i + 1* đến *len(ban)* tránh trùng lặp cặp

* Xác định điểm thân thiết của từng cặp khách:

*For nhom, diem\_nhom in ma\_tran\_than\_thiet.item():*

*ma\_tran\_than\_thiet* chứa mức điểm thân thiết của các cặp khách

* Sau khi dùng vòng lặp để chọn từng cặp khách rồi thì chúng ta phải kiểm tra các cặp khách đó có thuộc nhóm thân thiết không:

*If ban[i] in nhom and ban[j] in nhom:*

* Nếu như cặp đó có thuộc nhóm thân thiết thì cộng điểm vào biến *diem:*

diem += diem\_nhom

* Sau khi tìm thấy nhóm phù hợp thì dùng break để thoát khỏi vòng lặp và trả về tổng điểm *return diem*

***3.2.9. Hiển thị kết quả***

* In ra sơ đồ tối ưu và tổng điểm có được

*def main():*

*so\_do\_toi\_uu = giai\_thuat\_di\_truyen()*

*print("Sơ đồ chỗ ngồi tối ưu:")*

*for i, ban in enumerate(so\_do\_toi\_uu):*

*print(f"Bàn {i+1}: {' | '.join(ban)}")*

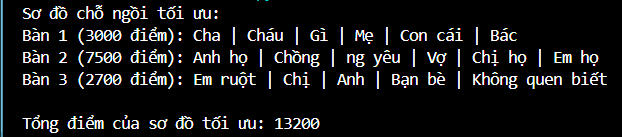
*tong\_diem = tinh\_diem\_than\_thiet(so\_do\_toi\_uu)*

*print(f"\nTổng điểm của sơ đồ tối ưu: {tong\_diem}")*

*if \_\_name\_\_ == "\_\_main\_\_":*

*main()*

* Kết quả:



Như vậy ta đã có được số lượng 3 bàn với sơ đồ tối ưu nhất có thể và tổng điểm đạt cao nhất là 13200

**KẾT LUẬN**

Thuật toán di truyền mang lại nhiều ưu điểm vượt trội trong bài toán tối ưu sắp xếp chỗ ngồi tiệc cưới. Trước hết, nó có thể tìm ra sơ đồ chỗ ngồi gần tối ưu trong không gian tìm kiếm rộng lớn mà phương pháp truyền thống khó thực hiện. Bằng cách sử dụng các cơ chế lai ghép và đột biến, thuật toán có thể liên tục cải thiện kết quả, đảm bảo những người có mối quan hệ thân thiết được xếp gần nhau, tối đa hóa trải nghiệm của khách mời. Ngoài ra, giải thuật di truyền không yêu cầu công thức toán học phức tạp mà có thể áp dụng trực tiếp vào dữ liệu thực tế, giúp dễ dàng triển khai và điều chỉnh.

Tuy nhiên, phương pháp này cũng có một số nhược điểm đáng lưu ý. Do tính ngẫu nhiên trong lai ghép và đột biến, thuật toán có thể mắc kẹt ở nghiệm cục bộ, không đảm bảo tìm ra sơ đồ chỗ ngồi tối ưu nhất. Hơn nữa, việc chạy qua nhiều thế hệ để tìm giải pháp tốt nhất có thể tiêu tốn nhiều thời gian và tài nguyên tính toán, đặc biệt khi số lượng khách mời lớn. Một hạn chế khác là thuật toán không thể đảm bảo ràng buộc cứng, chẳng hạn như yêu cầu một số người nhất định phải ngồi cùng nhau hoặc tránh ngồi chung với người khác. Điều này đòi hỏi các điều chỉnh thủ công hoặc bổ sung ràng buộc vào thuật toán.

**DANH MỤC TÀI LIỆU THAM KHẢO**

1. Nguyễn Hồng Sơn (2007), *Giáo trình hệ thống Mạng máy tính CCNA* (Semester 1), NXB Lao động xã hội.
2. Phạm Quốc Hùng (2017), *Đề cương bài giảng Mạng máy tính*, Đại học SPKT Hưng Yên.
3. James F. Kurose and Keith W. Ross (2013), *Computer Networking: A top-down approach sixth Edition*, Pearson Education.