**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THỰC PHẨM TP.HỒ CHÍ MINH**

-----o0o----

**Logo

Description automatically generated**

**ĐỒ ÁN LẦN 01**

**TRÍ TUỆ NHÂN TẠO**

**Thành phố Hồ Chí Minh, 28 tháng 09 năm 2022**

**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THỰC PHẨM TP.HỒ CHÍ MINH**

-----o0o----

**Logo

Description automatically generated**

**ĐỀ TÀI:**

**TÌM HIỂU THUẬT TÓAN DI TRUYỀN**

**Giảng viên hướng dẫn:** Trần Đình Toàn

**Nhóm: 09**

**Trưởng nhóm:** Nguyễn Nhâm Ngọ - 2001207130

**Thành viên:**

1. Phan Thị Mỹ Quyên - 2001207102
2. Trân Thanh Nhạc - 2001207111

**Thành phố Hồ Chí Minh, 26 tháng 10 năm 2022**

Mục lục

[**1.** **Giới thiệu** 2](#_Toc117710234)

[**2.** **Sơ đồ thuật toán** 3](#_Toc117710235)

[**3. Bài toán: Guessing Password - Đoán mật khẩu** 4](#_Toc117710236)

[**4. Kết luận** 9](#_Toc117710237)

[**5. Tài liệu tham khảo** 9](#_Toc117710238)

1. **Giới thiệu**  
     
   Genetic Algorithms (GAs) - Giải thuật di truyền là một kỹ thuật khoa học máy tính nhằm giải quyết các bài toán tối ưu tổ hợp. GAs dựa trên quá trình thích nghi tiến hóa của các quần thể sinh học dựa trên học thuyết của Darwin. Nó vận dụng các nguyên lý: di truyền, đột biến, chọn lọc tự nhiên và trao đổi chéo. GAs dùng một số thuật ngữ của ngành di truyền học như: nhiễm sắc thể, quần thể (Population), Gene... Nhiễm sắc thể được tạo thành từ các Gene (được biểu diễn của một chuỗi tuyến tính). Mỗi Gene mang một số đặc trưng và có vị trí nhất định trong nhiễm sắc thể. Mỗi nhiễm sắc thể sẽ biểu diễn một lời giải của bài toán.  Trong bài viết này tôi sẽ giải thích các khái niệm song song với việc lập trình ở một bài toán cụ thể.   
     
   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](https://en.wikipedia.org/wiki/Genetic_algorithm_scheduling), [hệ thống điều khiển](https://www.google.com.vn/url?sa=t&rct=j&q=&esrc=s&source=web&cd=4&cad=rja&uact=8&ved=0ahUKEwjy8NS--6vXAhVGOo8KHSc2DcoQFghEMAM&url=https%3A%2F%2Farc.aiaa.org%2Fdoi%2Fpdf%2F10.2514%2F3.20898&usg=AOvVaw0jy_BToGn-IuQLYTAoE2W1), [bài toán người đi du lịch](https://www.google.com.vn/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&cad=rja&uact=8&ved=0ahUKEwifp6fN-qvXAhUIMY8KHSYBDcIQFggvMAE&url=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FTravelling_salesman_problem&usg=AOvVaw1cGfv9RcYOXZ8Hc_9tZU6u),…
2. **Sơ đồ thuật toán**Diagram

   Description automatically generated

Giải thuật sẽ được thực hiện qua các bước sau:

* **Khởi tạo quần thể:** Sinh ra ngẫu nhiên một quần thể gồm *n*cá thể (trong đó *n* là lời giải cho bài toán).
* **Tính giá trị thích nghi:** Ước lượng độ thích nghi của mỗi cá thể.
* **Điều kiện dừng:** Kiểm tra điều kiện để 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).
* **Trao đổi chéo**: Với một xác suất được chọn, trao đổi chéo 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 quá:**Nếu thỏa mãn điều kiện dừng thì giải thuật kết thúc và chọn được lời giải tốt nhất trong quần thể hiện tại.

Ta có thể thấy rằng, khi **Điều kiện dừng** chưa được thỏa mán, quần thể mới sẽ liên tục được tạo ra bằng cách lặp lại 3 bước **Chọn lọc**, **Trao đổi chéo** và **Đột biến.**

GAs có 2 điều kiện dừng cơ bản:

1. Dựa trên cấu trúc nhiễm sắc thể, kiểm soát số gene được hội tụ, nếu số gene được hội tụ tại một điểm hoặc vượt quá điểm đó thì giải thuật kết thúc.
2. Dựa trên ý nghĩa đặc biệt của nhiễm sắc thể, đo sự thay đổi của giải thuật sau mỗi thế hệ, nếu thay đổi này nhỏ hơn một hằng số xác định thì giải thuật kết thúc.

# **3. Bài toán: Guessing Password - Đoán mật khẩu**

Về bài toán, sử dụng những ký tự trong bảng chữ cái để tái tạo lại mật khẩu. Cụ thể, ở bài này chúng ta sẽ bắt đầu với những ký tự: a-z, A-Z, !. để tái tạo nên mật khẩu: "Hello World!".

Các bước giải bài toán:

* **Khởi tạo quần thể:**Sinh ra ngẫu nhiên một đoạn dài 11 ký tự (dài bằng với mục tiêu "Hello World".
* **Tính giá trị thích nghi:** Đếm số ký tự trùng giữa đoạn tạo ra và "Hello World".
* **Điều kiện dừng:** Kiểm tra xem số ký tự trùng bằng 11 hay không.
* **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).
* **Trao đổi chéo (Không thực hiện ở bài toán này)**: Với một xác suất được chọn, trao đổi chéo 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 quá:**Khi số ký tự trùng bằng 11 thì dừng giải thuật.

**Chromosomes và Gene**  
Trong sinh học, nói một cách khái quát, trong một Chromosomes (nhiễm sắc thể) sẽ có thể chứa rất nhiều loại gene khác nhau.

Ở bài toán của chúng ta, tất cả những trình tự đoạn 11 ký tự ( độ dài bằng với "Hello World") được gọi là Chromosomes. Và mỗi loại ký tự sẽ đại diện cho 1 loại Gene khác nhau.

geneSet = " abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"

target = "Hello World"

Đầu tiên chúng ta khai báo biến geneSet chứa các ký tự trong bảng chữ cái, mỗi ký tự đại diện cho một loại Gene. Cùng với đó, khai báo biến target là đoạn ký tự mục tiêu "Hello World".

**Bước 1: Khởi tạo quần thể**

**import** **random**

**def** initialize\_chromosomes(length):

chromosomes = []

**while** len(chromosomes) < length:

sampleSize = min(length - len(chromosomes), len(geneSet))

chromosomes.extend(random.sample(geneSet, sampleSize))

**return** ''.join(chromosomes)

Sử dụng random.sample để tạo ngẫu nhiên từ geneSet theo độ dài cho trước. Chúng ta có thể có test bằng initialize\_chromosomes(11). Kết quả sẽ có dạng:  'yTtumWSgfkM'  
Chú ý: Vòng While sinh ra nhằm xử lý vấn đề nếu độ dài của Chromosomes dài hơn số lượng loại Gene ban đầu   
  
**Bước 2: Tính giá trị thích nghi**

Chúng ta thấy rằng, Chromosomes có khả năng là những trường hợp như:

* Sekmo xosmd
* Seklo Wocle
* Fello Wosld
* Gello World
* Hello World

Có thể nhận ra ngay đoạn cuối cùng là đoạn chính xác. Nhưng làm sao chúng ta có thể "đo" được độ tối ưu của Chromosomes này?  
Hàm lỗi (error function/ cost function/loss function) là một hàm số giúp đo độ tối ưu của một Chromosome. Ở hàm này, chúng ta đang hướng tới giá trị càng nhỏ càng tốt.  
Với bài toán Đoán mật khẩu, để đo độ tối ưu, chúng ta đếm số kí tự sai khác trên cùng vị trí giữa Chromosome và đoạn kí tự mục tiêu "Hello World". Vì vậy, hàm lỗi ở đây sẽ tối ưu ở số kí tự sai khác bằng 0.   
Dùng quy luật trên để tính sai khác cho các trường hợp Chromosomes trên:

* Sekmo xosmd(6)
* Seklo Wocle(4)
* Fello Wosld(2)
* Gello World(1)
* Hello World(0)

**def** error\_function(chromosome):

**return** len(target) - sum(1 **for** expected, actual \

**in** zip(target, chromosome) \

**if** expected == actual)

Hàm error\_function sẽ tính số ký tự sai khác ở cùng một vị trí giữa Chromosome và "Hello World". Có thể test hàm trên như sau: error\_function("Hello World") sẽ cho kết quả bằng 0.  
  
**Bước 3:** **Điều kiện điểm dừng**  
Kiểm tra điều kiện điểm dừng là error\_function có bằng 0 hay không.  
  
**Bước 4: Tiến hóa**  
Trong bước này, chúng ta sẽ gộp chung cả 3 kỹ thuật: **chọn lọc**, **trao đổi chéo** và **đột biến**. Mục tiêu của bước này là tạo ra thế hệ mới tối ưu hơn thế hệ cũ. Độ tối ưu sẽ được đo theo hàm lỗi. Chúng ta có thể mường tượng ra rằng, đây là một vòng lặp, mỗi vòng lặp sẽ thực hiện 3 kỹ thuật trên, sau đó check điều kiện dừng. Nếu không đáp ứng được thì vòng lặp sẽ tiếp tục cho đến khi đáp ứng được thì thôi.  
  
Ở bài toán này, chúng ta sẽ chỉ sử dụng chọn lọc và đột biến. Phần về trao đổi chéo tôi sẽ có một bài riêng vì nó có chút phức tạp hơn. 

**def** mutate(parent):

index = random.randrange(0, len(parent))

childGenes = list(parent)

newGene, alternate = random.sample(geneSet, 2)

childGenes[index] = alternate \

**if** newGene == childGenes[index] \

**else** newGene

**return** ''.join(childGenes)

index sẽ là một số được chọn ngẫu nhiên tức là đột biến sẽ xảy ra một cách ngẫu nhiên trên Chromosome. newGene và alternate cũng sẽ được ngẫu nhiên tạo ra từ geneSet.   
Vì sao lại phải tạo ra tận 2 đột biến? Câu trả lời là đột biến có thể vô nghĩa tại 1 điểm nên cần 1 đột biến khác thay thế.  
  
**Hàm hỗ trợ**  
Xây dựng hàm display giúp hiển thị sự thay đổi qua từng thế hệ và thời gian chạy tương ứng

**import** **datetime**

**def** display(guess):

timeDiff = datetime.datetime.now() - startTime

fitness = error\_function(guess)

print("**{0}\t{1}\t{2}**".format(guess, fitness, str(timeDiff)))

**Main**  
Sau đây sẽ là phần chạy main cho giải thuật GAs:

random.seed(1)

startTime = datetime.datetime.now()

bestParent = initialize\_chromosomes(len(target))

bestFitness = error\_function(bestParent)

display(bestParent)

**while** True:

child = mutate(bestParent)

childFitness = error\_function(child)

**if** bestFitness <= childFitness:

**continue**

display(child)

**if** childFitness == 0:

**break**

bestFitness = childFitness

bestParent = child

hJYVdpgEBDO 11 0:00:00

HJYVdpgEBDO 10 0:00:00

HJYVdpgEBDd 9 0:00:00.001003

HJYVopgEBDd 8 0:00:00.001003

HJYVopgErDd 7 0:00:00.004010

HJYVopgErld 6 0:00:00.005012

HJlVopgErld 5 0:00:00.008021

HJlVo gErld 4 0:00:00.008021

HelVo gErld 3 0:00:00.009036

HelVo gorld 2 0:00:00.010026

Hello gorld 1 0:00:00.017045

Hello World 0 0:00:00.020052

Khởi tạo quần thể với Chromosome đầu tiên bestParent . Tính hàm lỗi cho Chromosome này cho chúng ta kết quả là 11, cho thấy chưa có vị trí nào giữa Chromosome khởi tạo và "Hello World" trùng nhau.   
Chúng ta sẽ thấy rằng, việc chọn lọc đã diễn ra ở điều kiện vòng lặp, tức là cứ mỗi khi hàm lỗi giảm là sẽ giữ lại bestParent thay thế cho Chromosome cũ kém tối ưu hơn.

Trong sinh học, nói một cách khái quát, trong một Chromosomes (nhiễm sắc thể) sẽ có thể chứa rất nhiều loại gene khác nhau.

Ở bài toán của chúng ta, tất cả những trình tự đoạn 11 ký tự ( độ dài bằng với "Hello World") được gọi là Chromosomes. Và mỗi loại ký tự sẽ đại diện cho 1 loại Gene khác nhau.

**4. Kết luận**  
Chúng ta đã cùng nhau xây dựng một giải thuật di truyền và hiểu các bước diễn ra bên trong giải thuật.  
Giải thuật trên gần như là dạng đơn giản nhất. Nếu có bài tiếp theo, tôi sẽ giới thiệu thêm về một số vấn đề bên trong và xử lý những bài toán phổ biến và có đôi chút phức tạp hơn.

# **5. Tài liệu tham khảo**

**1.**[**https://github.com/handcraftsman/GeneticAlgorithmsWithPython**](https://github.com/handcraftsman/GeneticAlgorithmsWithPython)

**2. GIẢI THUẬT DI TRUYỀN (GAs) VÀ CÁC ỨNG DỤNG. ThS. Trần Kim Hương, Ths. Nguyễn Thị Ngọc Chi.**

**3.** **https://github.com/nerophung?tab=repositories**