TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN TP.HCM

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO SEMINAR

Môn Logic Mờ Và Ứng Dụng

- **Thành viên nhóm:**

Huỳnh Tiến Nam **1512333**
Đào Thanh Thiện **1712782**

Hồ Chí Minh, Tháng 1 Năm 2021

Mục lục

LỜI NÓI ĐẦU 3
1 Giới thiệu về Genetic Algorithm 4
1. Giới thiệu. 4
2. Cơ sở sinh học. 4
3. Genetic algorithm đơn giản. 4
4. So sánh Genetic algorithm với các kỹ thuật tối ưu khác. 4
5. Ưu điểm và hạn chế của Genetic algorithm. 5
6. Ứng dụng của Genetic algorithm. 5
2 Cài đặt Genetic Algorithm bằng MATLAB 6
2.1 Giới thiệu về MATLAB 6
2.2 Cấu trúc dữ liệu 6
2.3 Chromosomes. 6
2.4 Phenotypes. 6
2.5 Objective function values. 7
2.6 Fitness values. 7
2.7 Multiple subpopulations. 7
2.8 Toolbox Function 8
2.9 Genetic Algorithm Graphical User Interface Toolbox 8
3 Demo code 13
3.1 Bài toán TSP 13
3.2 Thuật toán Nearest Neighbor 14
3.3 Thuật toán GA 14
3.4 Nhận xét kết quả: 14
4 Tài liệu tham khảo. 16

LỜI NÓI ĐẦU

Nguồn gốc của các thuật toán tiến hoá là một nỗ lực để bắt chước một số quá trình diễn ra trong quá trình tiến hoá tự nhiên. Mặc dù các chi tiết về quá trình tiến hoá sinh học vẫn chưa được hiểu hoàn toàn (thậm chí đến ngày nay), nhưng có một số điểm được hỗ trợ bởi bằng chứng thực nghiệm mạnh mẽ:

- Tiến hoá là một quá trình hoạt động trên nhiễm sắc thể.

- Chọn lọc tự nhiên là cơ chế liên hệ nhiễm sắc thể với hiệu quả của thực thể mà chúng đại diện, do đó giúp sinh vật thích nghi tốt hơn.

- Quá trình tiến hoá diễn ra trong giai đoạn sinh sản. Tồn tại số lượng lớn các cơ chế sinh sản trong tự nhiên. Những trường hợp phổ biến nhất là đột biến và tái tổ hợp.

Dựa vào các đặc điểm trên, ba mô hình tính toán tiến hoá được phát triển độc lập. Thuật toán tiến hoá (EA) là một quá trình lặp đi lặp lại và ngẫu nhiên hoạt động trên một tập hợp các cá thể (quần thể).

Genetic Algorithm là mô hình tính toán nổi bật nhất và được sử dụng rộng rãi nhất trong hệ thống artificial-life. Các mô hình phi tập trung này tạo cơ sở để hiểu nhiều hệ thống và các hiện tượng khác trên thế giới.

1 Giới thiệu về Genetic Algorithm

1. Giới thiệu.

Charles Darwin đã nêu thuyết tiến hoá tự nhiên về nguồn gốc các loài. Qua nhiều thế hệ, các loài sinh vật phát triển dựa trên nguyên tắc chọn lọc tự nhiên "những con có sức khoẻ tốt nhất sẽ sống sót". Các kiểu gen, kiểu hình của các loài sinh vật được phát triển phù hợp với đặc tính và nhu cầu của sinh vật. Do đó, nó hoạt động rất hiệu quả trong tự nhiên.

Trong tự nhiên, sự cạnh tranh tài nguyên giữa các loài và các cá thể khác nhau trong cùng một loài, một đàn dẫn đến các việc các cá thể hoạt động kém hơn có ít cơ hội sống sót hơn và dần bị đào thải. Ngược lại các cá thể thích nghi tốt hơn sẽ tạo ra lượng lớn con cái thừa hưởng tính trạng tốt của bố mẹ. Sau một vài thế hệ, các loài tiến hoá tự phát để ngày càng thích nghi tốt hơn với môi trường của chúng.

Năm 1975, Holland đã phát triển ý tưởng này trong cuốn sách "Adaptation in natural and artificial systems". Ông mô tả cách áp dụng các nguyên tắc tiến hoá và xây dựng các Genetic algorithm đầu tiên. Lý thuyết của Holland được phát triển thêm và giờ đây Genetic algorithm trở thành công cụ mạnh mẽ để giải quyết các vấn đề tìm kiếm và tối ưu hoá.

2. Cơ sở sinh học.

Bộ môn khoa học giải quyết các cơ chế gây ra sự giống và khác nhau trong một loài gọi là Di truyền học. Khoa học về di truyền giúp chúng ta phân biệt được giữa kế thừa và các biến thể, và tìm cách giải thích những điểm giống nhau và khác nhau dựa trên khái niệm về Genetic algorithm và bắt nguồn trực tiếp từ di truyền tự nhiên, nguồn gốc và sự phát triển của chúng. Các khái niệm về Genetic algorithm có nguồn gốc trực tiếp từ quá trình tiến hoá tự nhiên.

3. Genetic algorithm đơn giản.

Để thực hiện được một hàm Genetic algorithm trước hết ta cần chú ý các hàm sau:

- Reproduction operator: được áp dụng trực tiếp trên nhiễm sắc thể, và sữ dụng để thực hiện đột biến và tái tổ hợp các giải pháp của bài toán. Là một hàm quan trọng vì hành vi của GA cực kỳ phụ thuộc vào nó.

- Fitness function: là hàm đánh giá cá thể là tốt hay xấu. Hàm này thay đổi tuỳ thuộc vào yêu cầu của bài toán.

Sau khi ta đã xác định được 2 hàm trên ta sẽ bắt đầu thực hiện Genetic algorithm. Đầu tiên ta cần tạo ra một quần thể ngẫu nhiên. Sau đó lặp lại quá trình để quần thể tiến hoá.

- Selection: chọn các thể để sinh sản

- Reproduction: tạo ra các cá thể con từ các cá thể được chọn.

- Evaluation: Đánh giá tính phù hợp của các nhiễm sắc thể mới.

- Replacement: Các cá thể cũ từ quần thể cũ bị thay thế bởi các cá thể mới được sinh ra.

Thuật toán dừng lại khi tập hợp hội tụ về giải pháp tối ưu.

4. So sánh Genetic algorithm với các kỹ thuật tối ưu khác.

Genetic algorithm khác với các kỹ thuật tối ưu hoá thông thường theo những cách sau:

- GA hoạt động với mã hoá của tập giải pháp chứ không phải với chính hàm giải pháp

- Hầu hết các kỹ thuật tối ưu hoá thông thường đều tìm kiếm từ một điểm duy nhất nhưng GA luôn hoạt động trên toàn bộ tổng thể các điểm.

- GA sữ dụng fitness function để đánh giá. Chúng có thể áp dụng cho bất kì bài toán tối ưu hoá liên tục hay rời rạc nào.

- GA không sữ dụng các phương pháp xác định.

5. Ưu điểm và hạn chế của Genetic algorithm.

Ưu điểm	Hạn chế
+ Khả năng chạy song song. + Các giá trị được tạo ngẫu nhiên bằng phân phối chuẩn. + Không gian giải pháp rộng hơn. + Hàm fitness áp dụng được trên nhiều bài toán. + Dễ tìm được tối ưu toàn cục. + Có thể áp dụng vào bài toán có đa mục tiêu. + Chỉ sử dụng các hàm đánh giá chức năng. + Dễ dàng sữ đổi cho các vấn đề khác nhau. + Xữ lí tốt các dữ liệu noise. + Dễ dàng xữ lí trên không gian lớn và thiếu thông tin + Hoạt động tốt cho vấn đề đa phương thức, trả về một bộ giải pháp + Không yêu cầu kiến thức hay gradient của reponse surface + Sự gián đoạn trên reponse surface có rất ít ảnh hưởng đến khả năng tổng quản trên tổng thể + Có khả năng chống lại sự mắc kẹt trong optima cục bộ.	+ Xác định fitness function + Định nghĩa biểu diễn cho bài toán + Xãy ra hội tụ sớm + Các vấn đề lựa chọn tham số + Không sữ dụng gradient + Không thể dễ dàng kết hợp thông tin cụ thể của vấn đề + Không giỏi xác định optima cục bộ + Không teminator hiệu quả + Cần kết hợp với kỹ thuật tìm kiếm cục bộ + Gặp khó khăn khi tìm ra tối ưu toàn cục chính xác + Yêu cầu số lượng lớn các đánh giá + Cấu hình không đơn giản

6. Ứng dụng của Genetic algorithm

Genetic algorithm được ứng dụng vào các vấn đề như sau:

- Nonlinear dynamical systems–predicting, data analysis
- Robot trajectory planning
- Evolving LISP programs (genetic programming)
- Strategy planning
- Finding shape of protein molecules
- TSP and sequence scheduling

2 Cài đặt Genetic Algorithm bằng MATLAB

2.1 Giới thiệu về MATLAB

MATLAB là phần mềm cung cấp môi trường tính toán số và lập trình, do Mathworks thiết kế. MATLAB cho phép tính toán số với ma trận, vẽ đồ thị hàm số hay biểu diễn thông tin, thực hiện thuật toán, tạo các giao diện người dùng và liên kết với những chương trình máy tính viết trên nhiều ngôn ngữ lập trình khác nhau.

MATLAB có nhiều chức năng hữu ích cho người sữ dụng Genetic algorithm và những người muốn thử nghiệm Genetic algorithm lần đầu tiên. Với tính linh hoạt của ngôn ngữ cấp cao của MATLAB, các vấn đề có thể được mã hoá trong file .m trong một khoảng thời gian ngắn. Kết hợp điều này với phân tích dữ liệu nâng cao, công cụ trực quan hoá và công cụ miền ứng dụng cho mục đích đặc biệt của MATLAB và người dùng được cung cấp một môi trường đồng nhất để khám phá tiềm năng của các thuật toán di truyền.

2.2 Cấu trúc dữ liệu

Về cơ bản MATLAB chỉ hỗ trợ một kiểu dữ liệu, một ma trận chữ nhật gồm các phần tử số thực hoặc số phức. Các cấu trúc dữ liệu chính trong Genetic algorithm toolbox là:

2.2.1 Chromosomes.

Cấu trúc dữ liệu Chromosomes lưu toàn bộ toàn bộ quần thể trong một ma trận duy nhất có kích thước Nind x Lind. Mỗi hàng tương ứng với kiểu gen của một cá thể.

$$\text{Chrom} = \begin{bmatrix} g_{1,1} & g_{1,2} & g_{1,3} & \cdots & g_{1,\text{Lind}} \\ g_{2,1} & g_{2,2} & g_{2,3} & \cdots & g_{2,\text{Lind}} \\ g_{3,1} & g_{3,2} & g_{3,3} & \cdots & g_{3,\text{Lind}} \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ g_{\text{Nind},1} & g_{\text{Nind},2} & g_{\text{Nind},3} & \cdots & g_{\text{Nind,Lind}} \end{bmatrix} \begin{matrix} \text{individual 1} \\ \text{individual 2} \\ \text{individual 3} \\ \cdot \\ \text{individual Nind} \end{matrix}$$

Biểu diễn dữ liệu này không bắt buộc một cấu trúc trên cấu trúc Chromosomes, chỉ yêu cầu tất cả các Chromosome có chiều dài bằng nhau. Do đó, các quần thể có cấu trúc hoặc quần thể với các cơ sở kiểu gen khác nhau có thể được sử dụng trong Genetic Algorithm Toolbox với điều kiện là chức năng giải mã phù hợp, ánh xạ Chromosomes thành Phenotypes, được sử dụng.

2.2.2 Phenotypes.

Mỗi chuỗi chứa trong cấu trúc Chromosomes được giải mã thành một vector hàng có thứ tự Nvar. Được lưu trữ dưới dạng ma trận số có kích thước Nind x Nvar, mỗi hàng tương ứng với một cá thể cụ thể.

```
Phen = bin2real(Chrom) % map genotype to phenotype
```

$$= \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & \cdots & x_{1,\,Nvar} \\ x_{2,1} & x_{2,2} & x_{2,3} & \cdots & x_{2,\,Nvar} \\ x_{3,1} & x_{3,2} & x_{3,3} & \cdots & x_{3,\,Nvar} \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ x_{Nind,1} & x_{Nind,2} & x_{Nind,3} & \cdots & x_{Nind,Nvar} \end{bmatrix} \begin{matrix} \text{individual 1} \\ \text{individual 2} \\ \text{individual 3} \\ \cdot \\ \text{individual Nind} \end{matrix}$$

2.2.3 Objective function values.

Các giá trị của Objective function có thể là vô hướng hoặc vector. Giá trị hàm Objective function được lưu trữ trong ma trận có kích thước Nind x Nobj, với mỗi hàng tương ứng với Objective function của từng cá nhân cụ thể.

```
Objv = OBJFUN(Phen) % Objective Function
```

$$= \begin{bmatrix} y_{1,1} & y_{1,2} & y_{1,3} & \cdots & y_{1,Nvar} \\ y_{2,1} & y_{2,2} & y_{2,3} & \cdots & y_{2,Nvar} \\ y_{3,1} & y_{3,2} & y_{3,3} & \cdots & y_{3,Nvar} \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ y_{Nind,1} & y_{Nind,2} & y_{Nind,3} & \cdots & y_{Nind,Nvar} \end{bmatrix} \begin{matrix} \text{individual 1} \\ \text{individual 2} \\ \text{individual 3} \\ \cdot \\ \text{individual Nind} \end{matrix}$$

2.2.4 Fitness values.

Fitness values được lấy từ objective function values thông qua một hàm chia tỷ lệ. Fitness là giá trị vô hướng không âm và được lưu trữ trong vector cột có độ dài Nind.

```
Fitn = ranking(ObjV) % fitness function
```

$$\text{Fitn} = \begin{matrix} f_1 & \text{individual 1} \\ f_2 & \text{individual 2} \\ f_3 & \text{individual 3} \\ \cdots & \\ f_{Nind} & \text{individual Nind} \end{matrix}$$

2.2.5 Multiple subpopulations.

Một quần thể đơn lẻ được chia thành nhiều quần thể con bằng cách sửa đổi việc sử dụng cấu trúc dữ liệu để các quần thể con được lưu trữ trong các khối liền kề trong một ma trận.

$$
\begin{aligned}
\text{Chrom} = \quad & \ldots \\
& \text{Ind}_1\ \text{Subpop}_1 \\
& \text{Ind}_2\ \text{Subpop}_1 \\
& \ldots \\
& \text{Ind}_N\ \text{Subpop}_1 \\
& \text{Ind}_1\ \text{Subpop}_2 \\
& \text{Ind}_2\ \text{Subpop}_2 \\
& \ldots \\
& \text{Ind}_N\ \text{Subpop}_2 \\
& \ldots \\
& \text{Ind}_1\ \text{Subpop}_{\text{subpop}} \\
& \text{Ind}_2\ \text{Subpop}_{\text{subpop}} \\
& \ldots \\
& \text{Ind}_N\ \text{Subpop}_{\text{subpop}}
\end{aligned}
$$

2.3 Toolbox Function

The Genetic Algorithm Toolbox là tập hợp các chức năng mở rộng khả năng của Optimization Toolbox và môi trường tính toán số của MATLAB.

Thuật toán di truyền sử dụng ba loại quy tắc chính ở mỗi bước để tạo ra thế hệ tiếp theo từ quần thể hiện tại:

+ Selection: chọn lọc những cá thể đóng góp vào quần thể ở thế hệ sau.

+ Crossover: kết hợp hai bố mẹ để tạo thành con cái cho thế hệ sau.

+ Mutate: áp dụng các thay đổi ngẫu nhiên đối với từng cá nhân cha mẹ để hình thành con cái.

Thuật toán di truyền tại dòng lệnh, gọi hàm Genetic algorithm với cú pháp:

[x fval] = ga(@fitnessfun, nvars, options)

Với:

X - Điểm mà tại đó giá trị cuối cùng đạt được

Fval - giá trị cuối cùng của fitness function

@fitnessfun – là hàm fitness function được lưu trong file .m

Nvar – số biến độc lập cho fitness function

Options - cấu trúc chứa các tuỳ chọn cho GA

2.4 Genetic Algorithm Graphical User Interface Toolbox

Điều đầu tiên phải làm để sử dụng GA là quyết định xem có thể tự động xây dựng các giải pháp cho vấn đề hay không. Ví dụ, trong bài toán TSP, mọi tuyến đường đi qua các thành phố được đề cập đều có khả năng là một giải pháp, mặc dù có lẽ không phải là giải pháp tối ưu. Nó phải làm điều đó bởi vì một GA yêu cầu một tập hợp ban đầu P của các giải pháp. Sau đó, phải quyết định biểu diễn "gen" nào sẽ sử dụng, ta có một số lựa chọn thay thế như nhị phân, số nguyên,

kép, hoán vị, v.v (binary, integer,double, permutation). Nhị phân và kép được sử dụng phổ biến nhất vì chúng linh hoạt nhất.

Sau khi lựa chọn đại diện gen, nó phải được quyết định: Phương pháp chọn bố mẹ từ quần thể P (Bánh xe Roulette chi phí, Lấy mẫu chung ngẫu nhiên, Bánh xe Roulette Xếp hạng, v.v.), cách những cặp bố mẹ này sẽ "giao phối" để tạo ra con cháu , phương pháp đột biến (tùy chọn nhưng hữu ích), phương pháp sẽ sử dụng để điền thế hệ tiếp theo và điều kiện kết thúc của thuật toán (số thế hệ, giới hạn thời gian, ngưỡng chất lượng chấp nhận được).

Phương trình của hàm Rastrigin và matlab (m - file) được cung cấp dưới đây:

“function y = rasfun(x)

n=2;

s=0;

for j=1:n

s=s+(x(j)^2-10*cos(2*pi*x(j)));

end

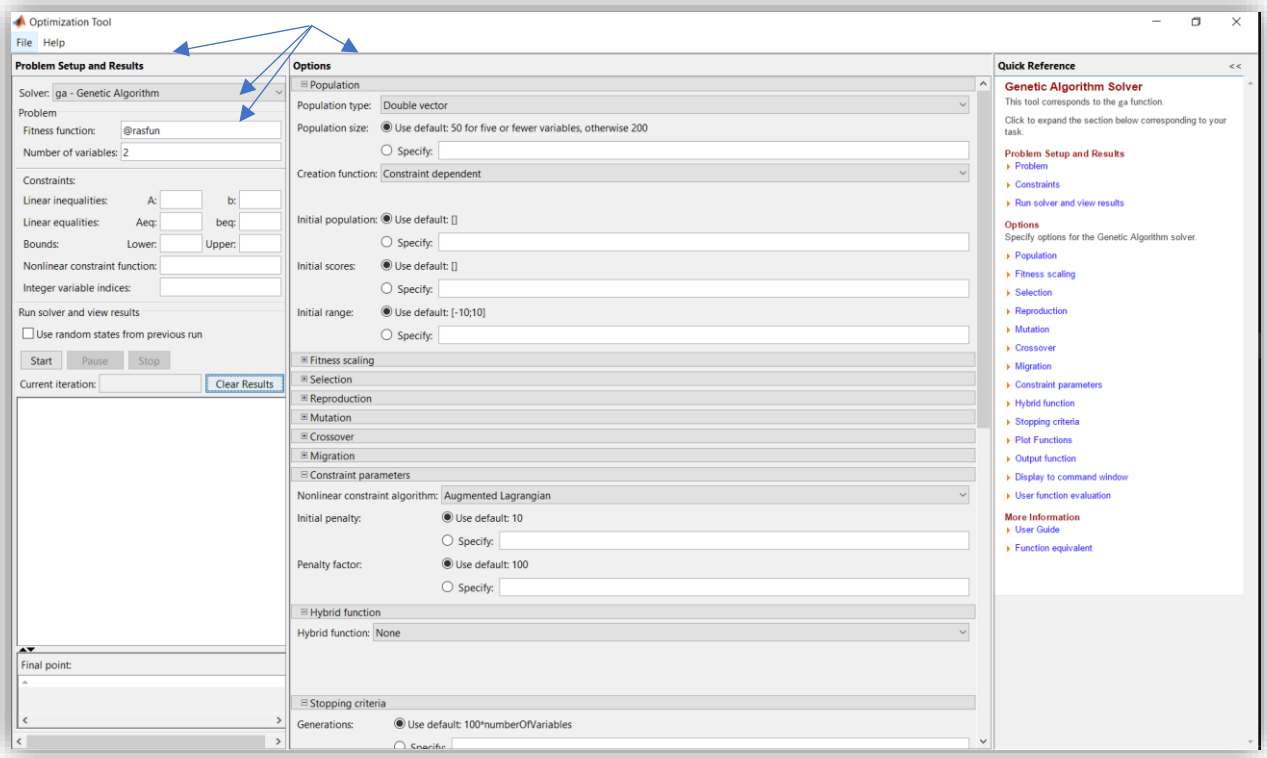
y=10*n+s;”

Lưu file “rasfun.m”

Mở Optimization Tool bằng 1 trong 2 cách sau:

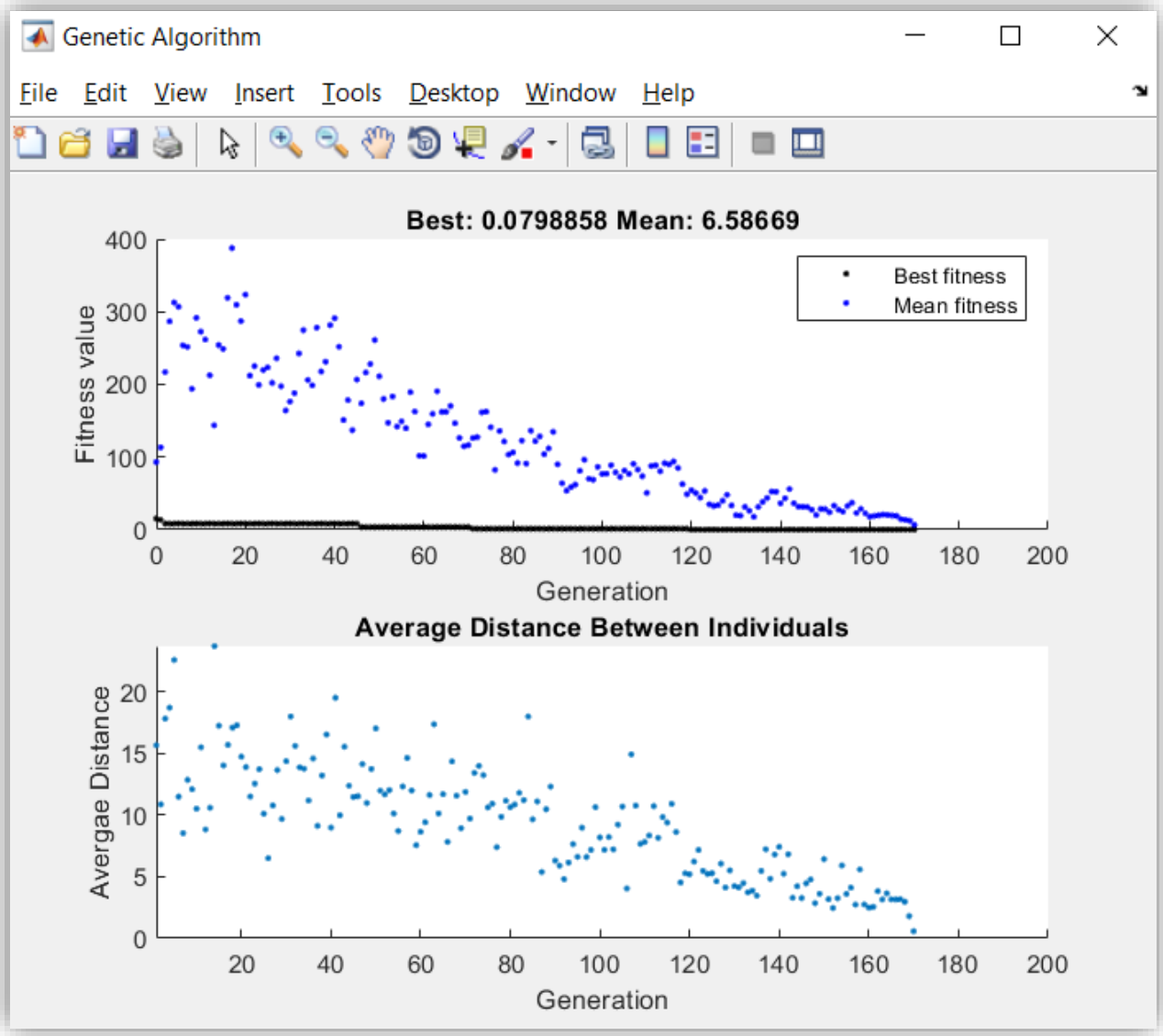
- Gõ “optimtool” vào command window của matlab
- Ở tab “APPS” chọn Optimization

Chúng ta sẽ triển khai hàm Rastrigin trong trường thích hợp và số biến là **2** và kiểu quần thể là **double vector**.

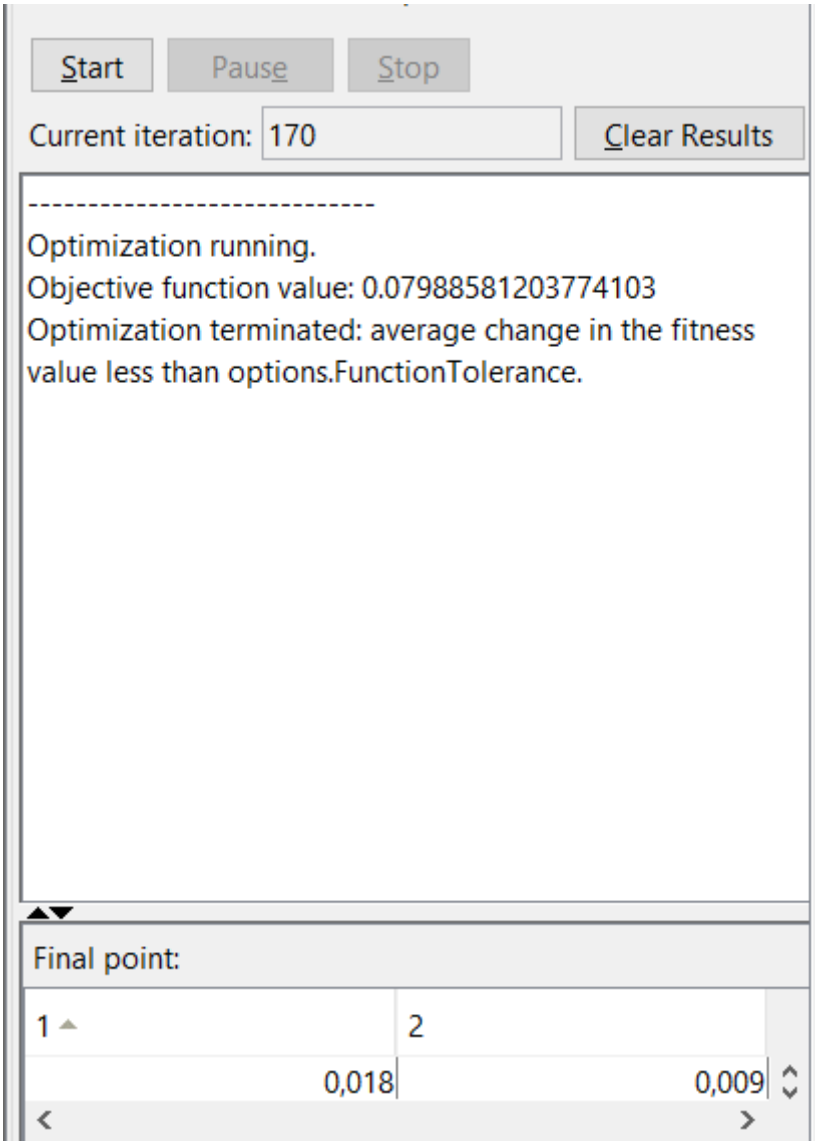


Ở mục **Plot functions**, ta tích vào 2 ô **Best fitness** và **Distance**. Còn lại để mặc định.

Khi mọi thứ sẵn sàng ta bấm **Start**, thuật toán sẽ bắt đầu, đồ thị sẽ xuất hiện và kết quả được hiện thị như hình sau.


Genetic Algorithm
File Edit View Insert Tools Desktop Window Help
Best: 0.0798858 Mean: 6.58669
Best fitness
Mean fitness
Fitness value
Generation
Average Distance Between Individuals
Avergae Distance
Generation

Hình 1



Giá trị tốt nhất của hàm fitness (giá trị nhỏ nhất từ khi chúng ta cực tiểu hàm) và điều kiện chấm dứt đã được in, cùng với giải pháp (final point - rất gần với (0;0)). Vì phương pháp này ngẫu nhiên, đừng mong có thể tái sinh kết quả giống hệt như trên trong một lần chạy khác.

Từ hình 1 có thể thấy giá trị fitness nhỏ dần. Đó là một dấu hiệu cho thấy việc tối ưu hóa diễn ra vì không chỉ giá trị fitness của cá thể tốt nhất bị giảm, ngay cả fitness mean của quần thể cũng bị giảm (nghĩa là, về giá trị fitness, toàn bộ quần thể đã được cải thiện chúng ta có các giải pháp tốt hơn trong quần thể).

Hiệu suất của GA bị ảnh hưởng bởi sự đa dạng của quần thể ban đầu. Nếu khoảng cách trung bình giữa các cá thể lớn, đó là dấu hiệu của sự đa dạng cao; nếu khoảng cách trung bình nhỏ thì nó thể hiện tính đa dạng thấp trong quần thể. Nếu độ đa dạng quá cao hoặc quá thấp, thuật toán di truyền có thể không hoạt động tốt.

Tăng quy mô dân số cho phép di truyền thuật toán để tìm kiếm nhiều điểm hơn và do đó thu được kết quả tốt hơn. Tuy nhiên, quy mô dân số càng lớn, thuật toán di truyền càng mất nhiều thời gian để tính toán thế hệ.

Cuối cùng, một tham số khác ảnh hưởng đến sự đa dạng của quần thể là **Fitness Scaling**. Nếu giá trị fitness khác nhau quá rộng, các cá thể có các giá trị thấp nhất (nhớ lại rằng ta đang cực tiểu) tái tạo quá nhanh, chiếm lĩnh quần thể quá nhanh và ngăn GA tìm kiếm các khu vực khác của không gian giải pháp. Mặt khác, nếu các giá trị khác nhau chỉ một chút thôi, tất cả các cá nhân đều có cùng cơ hội sinh sản và việc tìm kiếm sẽ tiến bộ rất chậm.

Fitness Scaling điều chỉnh các giá trị fitness (giá trị được chia tỷ lệ) trước bước lựa chọn của GA. Điều này được thực hiện mà không thay đổi thứ tự xếp hạng, tức là cá nhân tốt nhất dựa trên giá trị fitness thô vẫn là người tốt nhất trong xếp hạng được chia tỷ lệ. Chỉ các giá trị được thay đổi và do đó xác suất của một cá thể được chọn để giao phối bằng thủ tục chọn lọc. Điều này ngăn GA hội tụ quá nhanh, cho phép thuật toán tìm kiếm không gian giải pháp tốt hơn.

Bảng **Selection** trong **Options** điều khiển **Selection Function**, nghĩa là, cách các cá thể được chọn để trở thành cha mẹ. Lưu ý rằng cơ chế này hoạt động trên giá trị được chia tỷ lệ, như đã mô tả trước. Hầu hết các phương pháp nổi tiếng đều được trình bày (uniform, roulette và tournament). Một cá thể có thể được chọn nhiều lần với tư cách là cha mẹ, trong trường hợp đó, nó góp phần gen của nó cho nhiều hơn một con.

Bảng **Reproduction** trong **Options** kiểm soát cách GA tạo ra thế hệ tiếp theo. Ở đây bạn chỉ định lượng tinh hoa và phần nhỏ dân số của thế hệ tiếp theo được tạo ra thông qua giao phối (phần còn lại do đột biến tạo ra). Các tùy chọn là:

Elite Count: số lượng cá thể với các giá trị fitness tốt nhất trong thế hệ hiện tại được đảm bảo tồn tại đến thế hệ sau. Những những cá nhân được gọi là những người con ưu tú. Giá trị mặc định số Elite là 2.

Lưu ý rằng nếu **Elite Count lớn** thì quá trình **hội tụ sớm** sẽ xảy ra và có thể làm cho việc tìm kiếm **ít hiệu quả** hơn.

Crossover Fraction: là tỷ lệ của các cá thể ở thế hệ tiếp theo, ngoài những đứa trẻ ưu tú, được tạo ra bởi sự giao nhau (còn lại là do đột biến). **Crossover fraction** là **'1'** chỉ ra rằng tất cả cá thể con ngoại trừ cá thể con ưu tú là các cá thể lai ghép (tái tổ hợp). **Crossover fraction** là **'0'** cho biết rằng tất cả cá thể con đều là cá thể con đột biến.

Mutation (Đột biến): Đây là thay đổi ngẫu nhiên một hoặc nhiều các ký tự trong chuỗi đại diện cho một cá thể.

3 Demo code

3.1 Bài toán TSP

Bài toán người bán hàng (TSP) được biết đến là một trong những bài toán được nghiên cứu sâu rộng trong tối ưu hóa được thực hiện bởi một người bán hàng, người đi từ một thành phố đến tất cả các thành phố khác đúng một lần và quay trở lại thành phố xuất phát để tìm con đường ngắn nhất. Các cách tiếp cận để giải quyết TSP, thường kiểm tra tất cả các khả năng cho chuyến tham quan N thành phố mà yêu cầu N! phép cộng toán học. Tổng số đường đi có thể có tăng đáng kể với luỹ thừa của N. Trong số thuật toán meta-heuristic có thể giải quyết bài toán TSP, ta tập trung vào hai cái điển hình NN và GA. Thuật toán láng giềng gần nhất là một thuật toán tham lam tìm ra giải pháp ứng cử viên cho TSP bằng cách sử dụng các phương tiện đơn giản. Mặt khác thuật toán GA tuân theo các nguyên tắc tiến hóa để giải quyết vấn đề tối ưu hóa, trong trường hợp của ta là TSP. GA là một tìm kiếm heuristic bắt chước quá trình tiến hóa tự nhiên. Tham số GA như như tỷ lệ đột biến, tỷ lệ trao đổi chéo, kích thước quần thể và tối đa số thế hệ có thể ảnh hưởng đến hiệu suất của thuật toán giải quyết các vấn đề tối ưu hóa.

3.2 Thuật toán Nearest Neighbor

Phương pháp NN so sánh các khoảng cách xảy ra từ một điểm dữ liệu đến hàng xóm gần nhất của nó trong một tập dữ liệu được phân bố ngẫu nhiên. Thuật toán NN là một thuật toán tham lam theo một thủ tục tham lam rất đơn giản để giải quyết tsp. Chiến lược đầu tiên được giới thiệu và sử dụng để giải quyết vấn đề tsp là thuật toán NN, bắt đầu với một thành phố được chọn ngẫu nhiên và nó sẽ thêm thành phố gần nhất nhưng chưa được thăm đến thành phố cuối cùng vào chuyến đi cho đến khi tất cả các thành phố được viếng thăm.

Các bước:

a) Một đỉnh sẽ được chọn ngẫu nhiên được xem như đỉnh hiện tại
b) Một cạnh ngắn nhất sẽ được chọn để kết nối đỉnh hiện tại với đỉnh chưa được truy cập V.
c) Đỉnh hiện tại là đỉnh V.
d) Đỉnh V được đánh dấu là đã được truy cập.
e) Nếu tất cả các đỉnh trong miền đều được truy cập, chấm dứt thủ tục.

Đầu ra của thuật toán là dãy tất cả các đỉnh được truy cập. Điều này có nghĩa là một chuyến đi ngắn nhưng không phải là một chuyến đi tối ưu. Bởi vì bản chất của nó, thuật toán NN đã bỏ lỡ một số tuyến đường ngắn có thể dễ dàng phát hiện được với mắt thường

3.3 Thuật toán GA

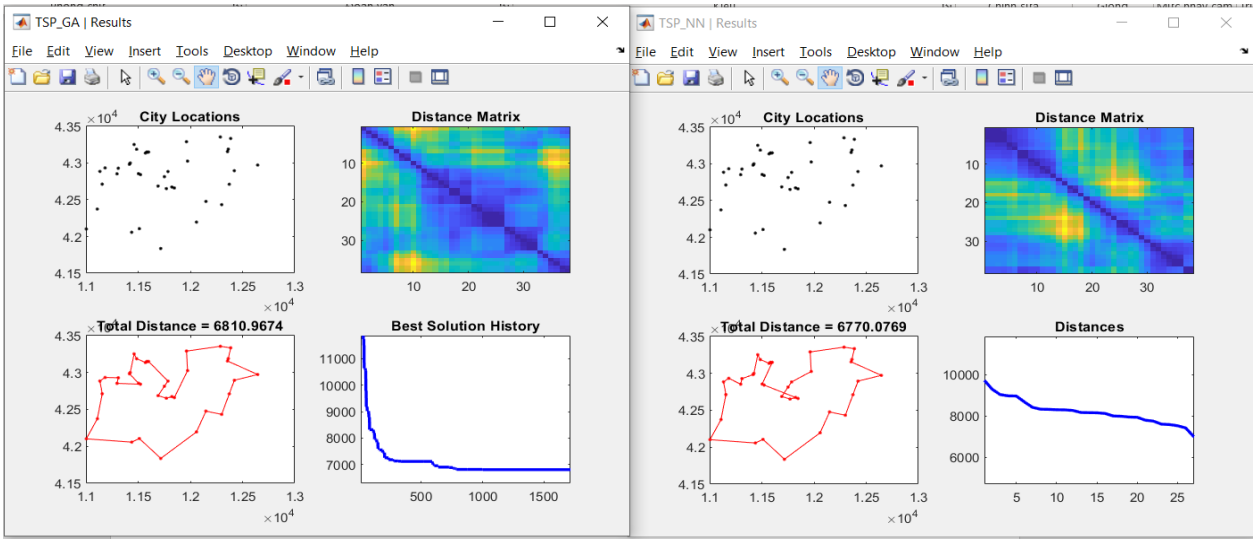
Các bước chung của GA như sau:

1) Tạo ra số lượng ngẫu nhiên của N chuyến đi.
2) Đánh giá độ thích nghi của mỗi chuyến đi.
3) Tạo ra một quần thể mới.
   - Chọn hai phụ huynh từ một quần thể theo độ thích nghi của chúng.
   - Nếu có một sự tái tổ hợp giữa hai phụ huynh, hai con sẽ được sinh ra, và nếu không có quá trình sự tái tổ hợp thì trẻ sẽ được sao chép từ cha mẹ.
   - Nếu thao tác đột biến được thực hiện, điều này sẽ thay đổi chuyến đi từ đầu ra, nếu không thì các cá thể con giống quần thể trước.
   - Thêm các cá thể con mới vào một quần thể mới.
4) Đánh giá độ thích nghi của mỗi chuyến đi.
5) Nếu các tiêu chí dừng được thỏa mãn, thuật toán dừng và hiển thị chuyến đi tốt nhất, nếu không phải nó sẽ bắt đầu từ bước 3 và tiếp tục lặp lại.

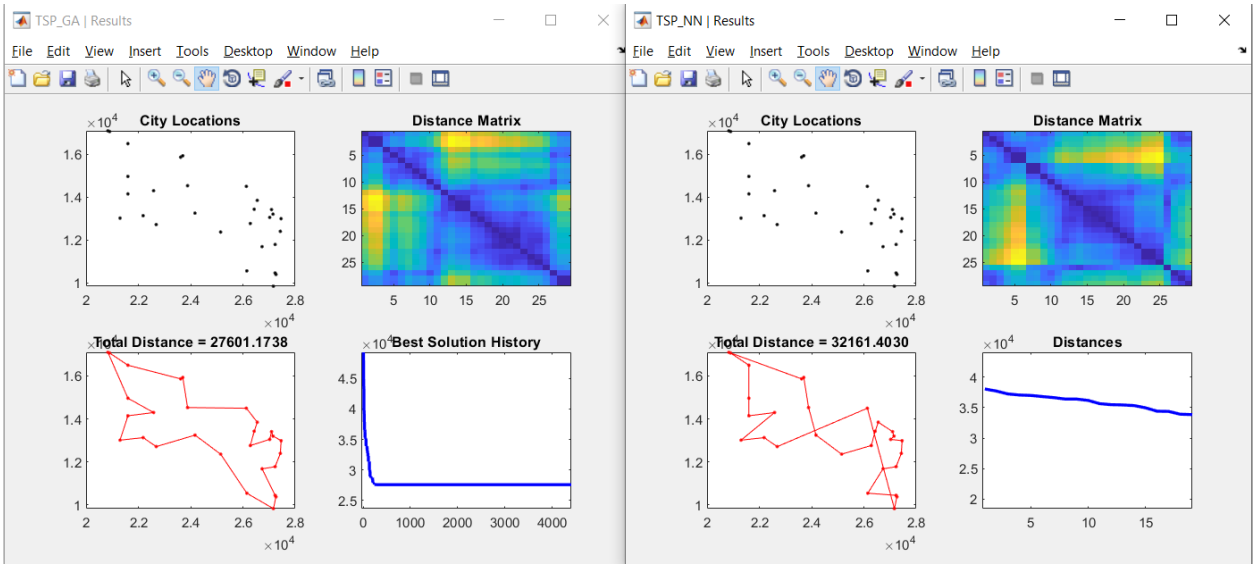
3.4 Nhận xét kết quả:

- Ta có thể được quan sát thấy rằng GA khá ổn định về chi phí khi nó tạo ra các giải pháp gần nhau hơn.
- Ngược lại với GA, thuật toán NN tạo ra một loạt các giải pháp mà một số trong số đó cách xa giải pháp tối ưu và một số là rất gần với giải pháp tối ưu. Thời gian chạy của thuật toán NN tốt hơn GA.
- Hiệu suất của thuật toán NN chủ yếu dựa vào số lượng thành phố, nó thực hiện rất tốt cho một số ít các thành phố nhỏ hơn 50. Tuy nhiên, đối với số lượng lớn các thành phố hoạt động thấp hơn và các giải pháp không chính xác.

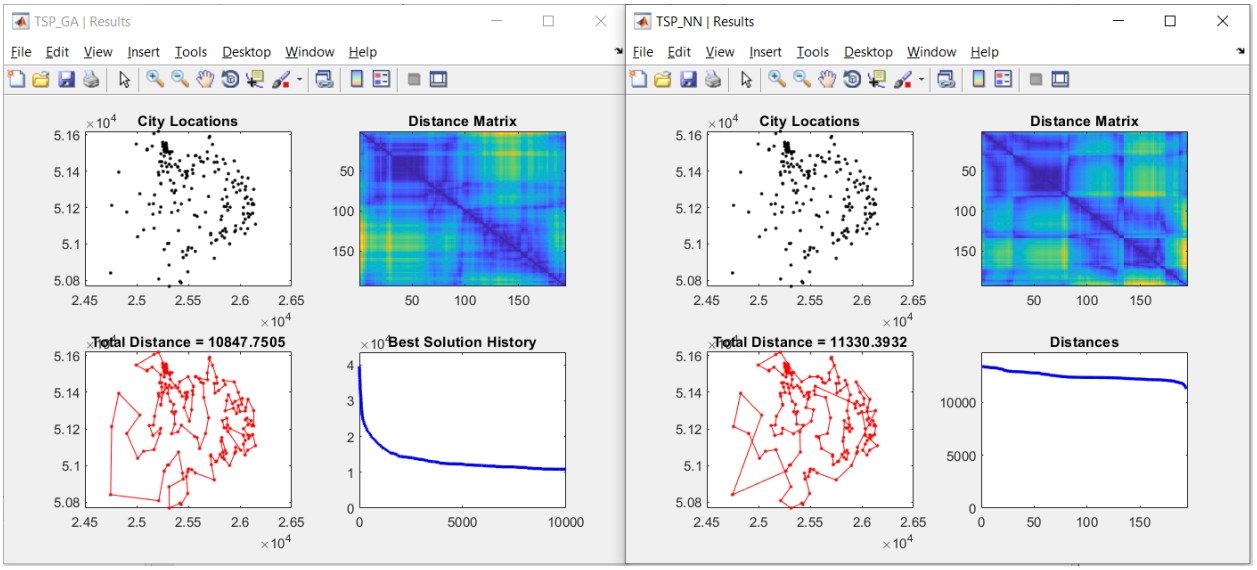
Djibouti - 38 Cities (dj38)



Western Sahara - 29 Cities (wi29)



Qatar - 194 Cities (qa194)



4 Tài liệu tham khảo.

[1] Sivanandam, S.N. Deepa, S. N. - Introduction to Genetic Algorithms - 2008

[2] Besan A. AlSalibi, Marzieh Babaeian Jelodar and Ibrahim Venkat - A Comparative Study between the Nearest Neighbor and Genetic Algorithms: A revisit to the Traveling Salesman Problem - International Journal of Computer Science and Electronics Engineering (IJCSEE) Volume 1, Issue 1 (2013) ISSN 2320–4028 - 12/2012

[3] The MathWorks, Inc - Genetic Algorithm and Direct Search Toolbox User's Guide - 2004

[4] Mr. Manish Saraswat, Mr. Ajay Kumar Sharma - Genetic Algorithm for optimization using MATLAB - Volume 4, No. 3, March 2013 (Special Issue) International Journal of Advanced Research in Computer Science - 1/2013

[5] MATLAB documentation