LAPORAN ANALISIS

“IMPLEMENTASI ALGORITMA *PARTICLE SWARM OPTIMIZATION* MENGGUNAKAN *ROSENBROCK FUNCTION*”

Disusun oleh :

Nama : Marde Fasma

NRP : 05111640000046

Mata Kuliah : Topik Khusus

Kelas : B

Departemen : Informatika

Daftar Isi

[Latar Belakang 3](#_Toc526489915)

[Motivasi Persoalan 4](#_Toc526489916)

[Ide 4](#_Toc526489917)

[Metode Penyelesaian 4](#_Toc526489918)

[1. Initial Population 4](#_Toc526489919)

[2. Fitness Function 6](#_Toc526489920)

[3. Normalize Fitness Function 6](#_Toc526489921)

[4. Selection 7](#_Toc526489922)

[5. Crossover 7](#_Toc526489923)

[6. Mutation 8](#_Toc526489924)

[Generation 10](#_Toc526489925)

[Generasi pertama 10](#_Toc526489926)

[Generasi kedua 10](#_Toc526489927)

[Generasi ketiga 11](#_Toc526489928)

[Generasi ke empat 11](#_Toc526489929)

[Generasi ke lima 12](#_Toc526489930)

[Generasi ke enam 12](#_Toc526489931)

[Generasi ke tujuh 13](#_Toc526489932)

[Generasi ke delapan 13](#_Toc526489933)

[Generasi ke sembilan 14](#_Toc526489934)

[Generasi ke-10 14](#_Toc526489935)

[Generasi ke-15 15](#_Toc526489936)

[Generasi ke 50 15](#_Toc526489937)

[Generasi ke 2500 16](#_Toc526489938)

[Generasi ke-7910 16](#_Toc526489939)

# Latar Belakang

*Particle Swarm Optimization* (PSO) merupakan metode komputasi yang mengoptimalkan masalah dengan cara iterative serta meningkatkan solusi dari kandidat sebelumnya. PSO didasarkan pada perilaku sekawanan burung atau ikan. Algoritma PSO meniru perilaku sosial organisme ini. Perilaku sosial terdiri dari tindakan individu dan pengaruh dari individu-individu lain dalam suatu kelompok. Kata partikel menunjukkan, misalnya, seekor burung dalam kawanan burung. Setiap individu atau partikel berperilaku dengan cara menggunakan kecerdasannya (*intelligence*) sendiri dan juga dipengaruhi perilaku kelompok kolektifnya. Dengan demikian, jika satu partikel atau seekor burung menemukan jalan yang tepat atau pendek menuju ke sumber makanan, sisa kelompok yang lain juga akan dapat segera mengikuti jalan tersebut meskipun lokasi mereka jauh dikelompok tersebut.

Dalam PSO, kawanan diasumsikan mempunyai ukuran tertentu dengan setiap partikel posisi awalnya terletak di suatu lokasi yang acak dalam ruang multidimensi. Setiap partikel diasumsikan memiliki dua karakteristik: posisi dan kecepatan. Setiap partikel bergerak dalam ruang/space tertentu dan mengingat posisi terbaik yang pernah dilalui atau ditemukan terhadap sumber makanan atau nilai fungsi objektif. Setiap partikel menyampaikan informasi atau posisi terbaiknya kepada partikel yang lain dan menyesuaikan posisi dan kecepatan masing-masing berdasarkan informasi yang diterima mengenai posisi tersebut.

Dari sini, saya akan mengambil permasalahan kelompok burung yang sedang mencari makanan. Dengan menentukan *Global Best* yang akan di gantikan bila mana ada yang seekor burung yang lebih baik dari global best tersebut.

# Motivasi Persoalan

Motivasi saya mengambil permasalahan kawanan burung mencari makan adalah karena permasalahan tersebut mudah dikerjakan dalam bentuk bahasa pemrograman. Walaupun masih sedikit dokumentasi yang terkait permasalahan tersebut, itu bukan menjadi suatu permasalahan. Dengan menggunakan *Rosenbrock Function* saya lebih dipermudah untuk menentukan nilai *fitness* serta fungsi dari *Rosenbrock* lebih mudah di visualisasi.

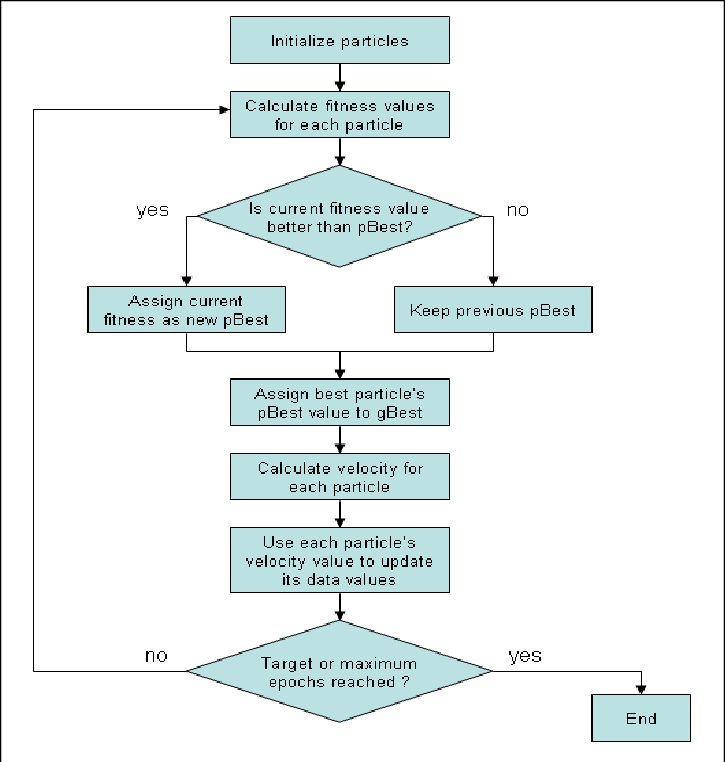
# Ide

Ide yang akan saya gunakan dalam PSO adalah melakukan iterasi sebanyak 50 kali karena saya menganggap iterasi / generasi ke 50 sudah cukup dekat dan bagus dengan nilai tujuan. Pertama saya akan melakukan pembuatan partikel berjumlah 10 partikel dengan posisi tiap tiap partikel didapatkan secara random. Dari masing masing partikel akan dihitung *fitness* dengan menggunakan *Rosenbrock Function*. Setelah mendapatkan nilai *fitness*, maka akan dilakukan sorting. Dalam setiap iterasi, dilakukan *update* nilai *global best* agar dapat menjadi acuan tujuan dalam generasi berikutnya. Hingga mencapai 50 generasi maka nilai dari koordinat tujuan akan ditampilkan.

# Metode Penyelesaian

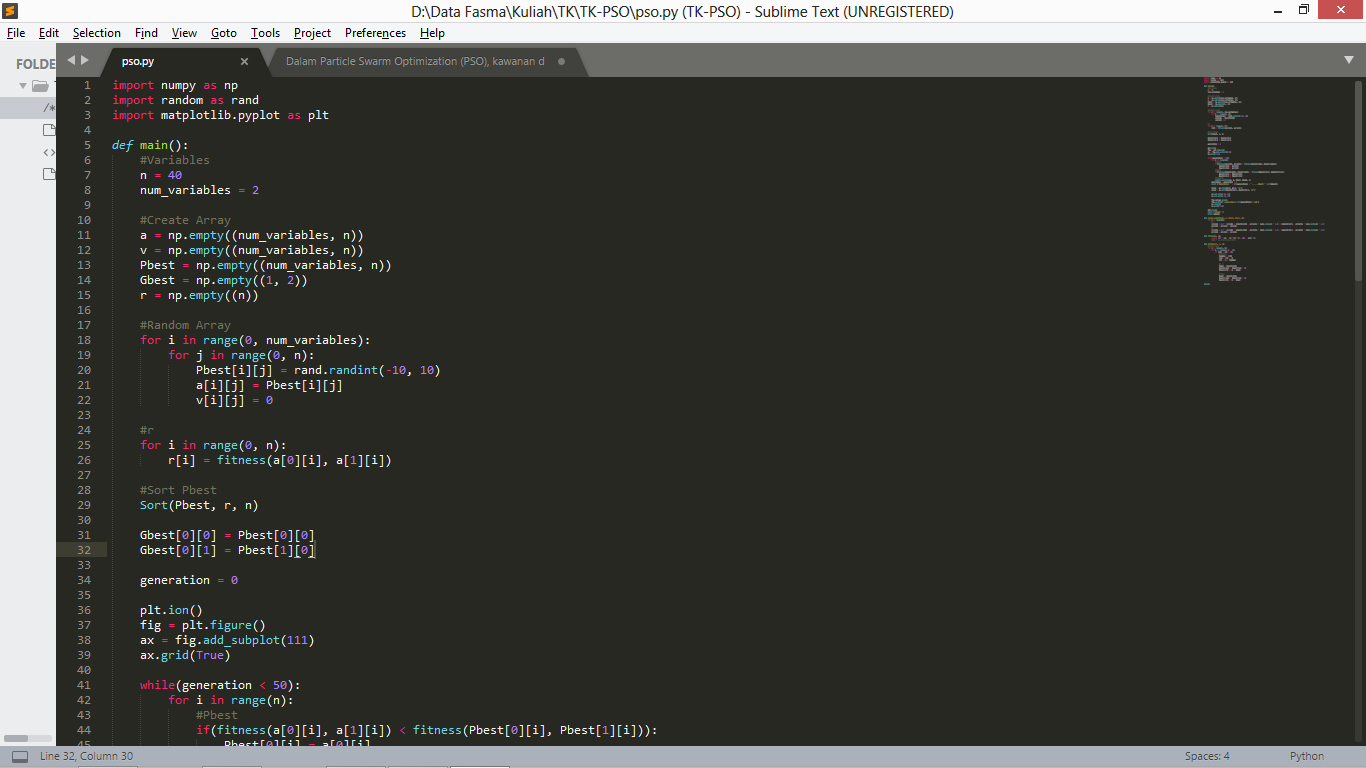
Dalam melakukan penyelesaian permasalahan dengan PSOterdapat 2 fase yang akan dilakukan secara berulang ulang untuk menentukan generasi kegenerasinya. Kali ini saya menggunakan Bahasa pemrograman Python sebagai Bahasa implementernya. Saya juga melakukan penambahan library dengan menggunakan library yang disediakan oleh matplotlib (matplotlib.org) serta NumPy Berikut 2 fase beserta pseudocode tersebut :

## Flow Chart



## Initial Particles

Dalam *Initial Particles* kita hanya melakukan pembuatan matriks beserta mengisi matriks tersebut dengan sebuah nilai random. Berikut kode sumbernya :

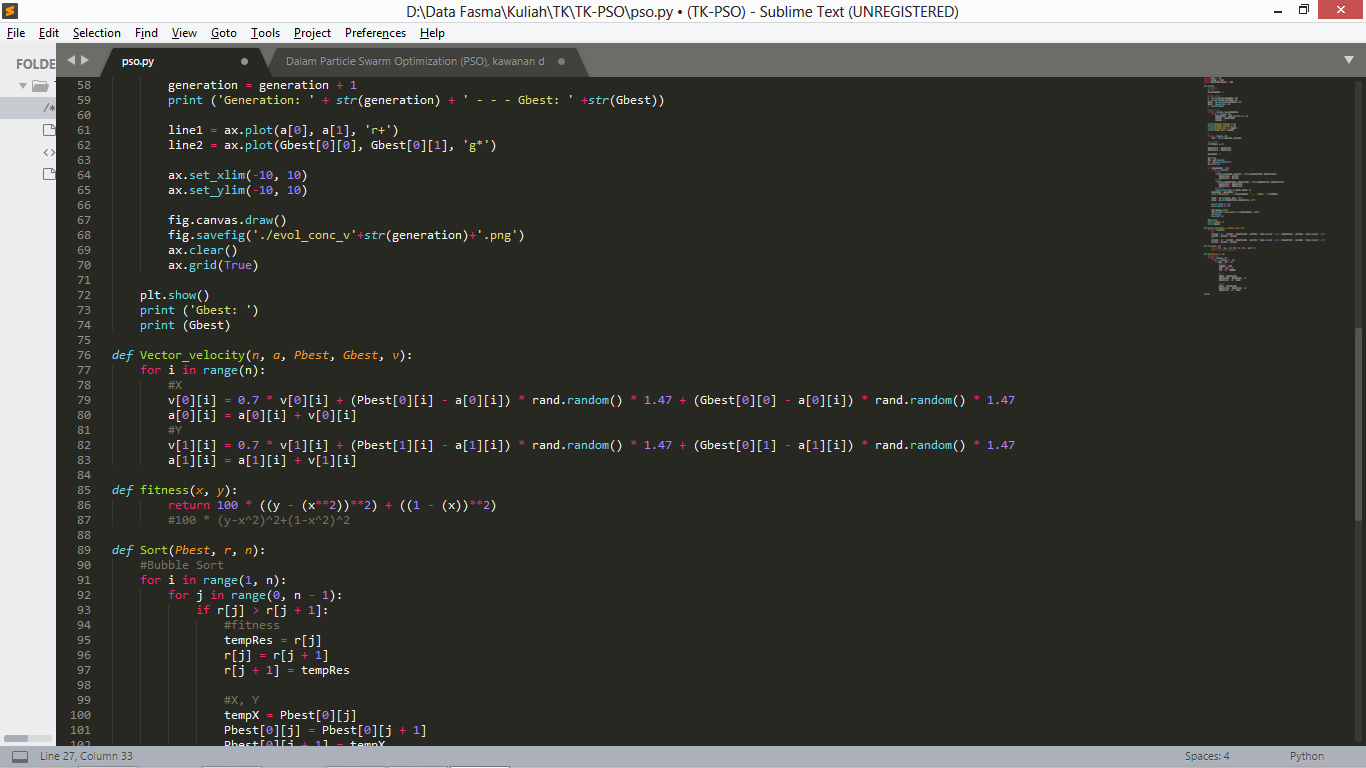


Dalam fungsi ini terdapat akan menghasilkan matriks a sebagai posisi dan matriks v sebagai velocity serta r untuk fitness dari masing masing partikel. Berikut hasil dari inisialisasi matriks :

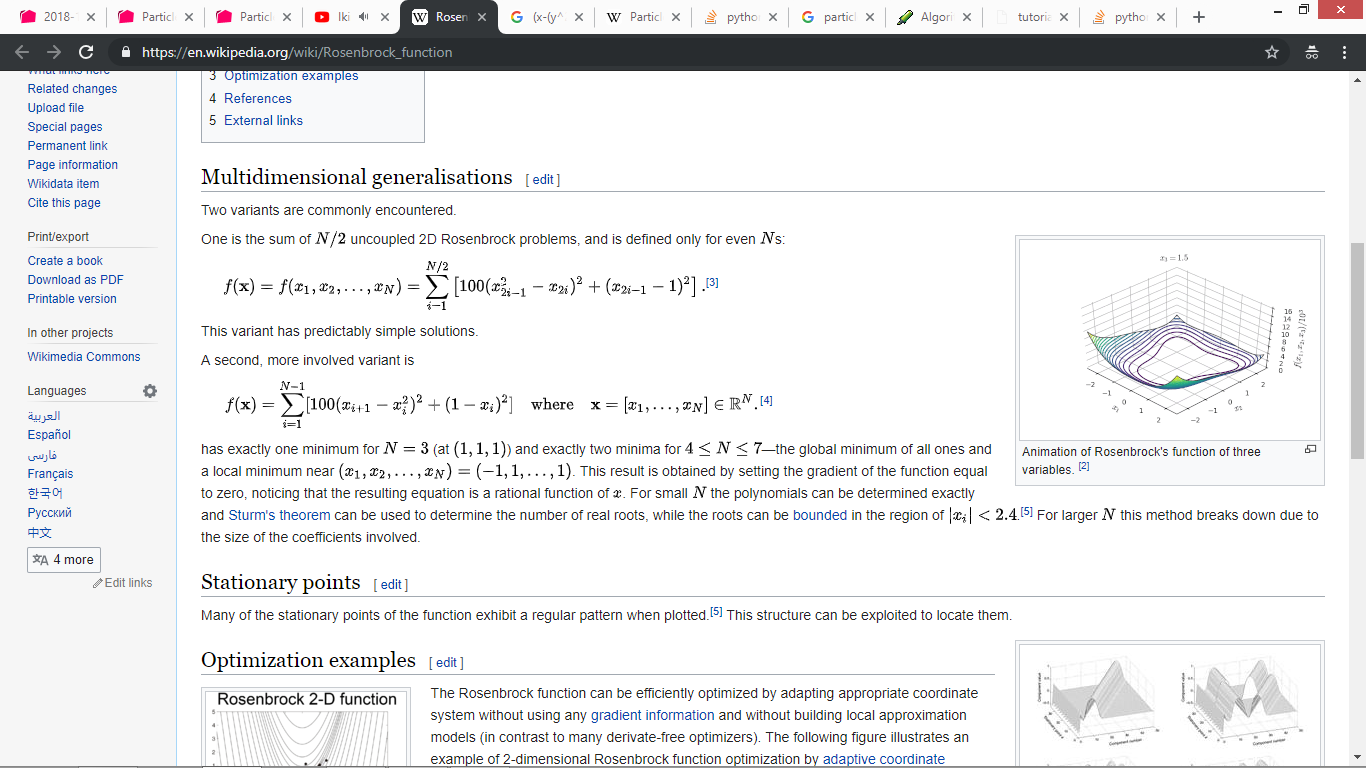
Untuk visualisasi dapat dilihat dalam plot berikut :

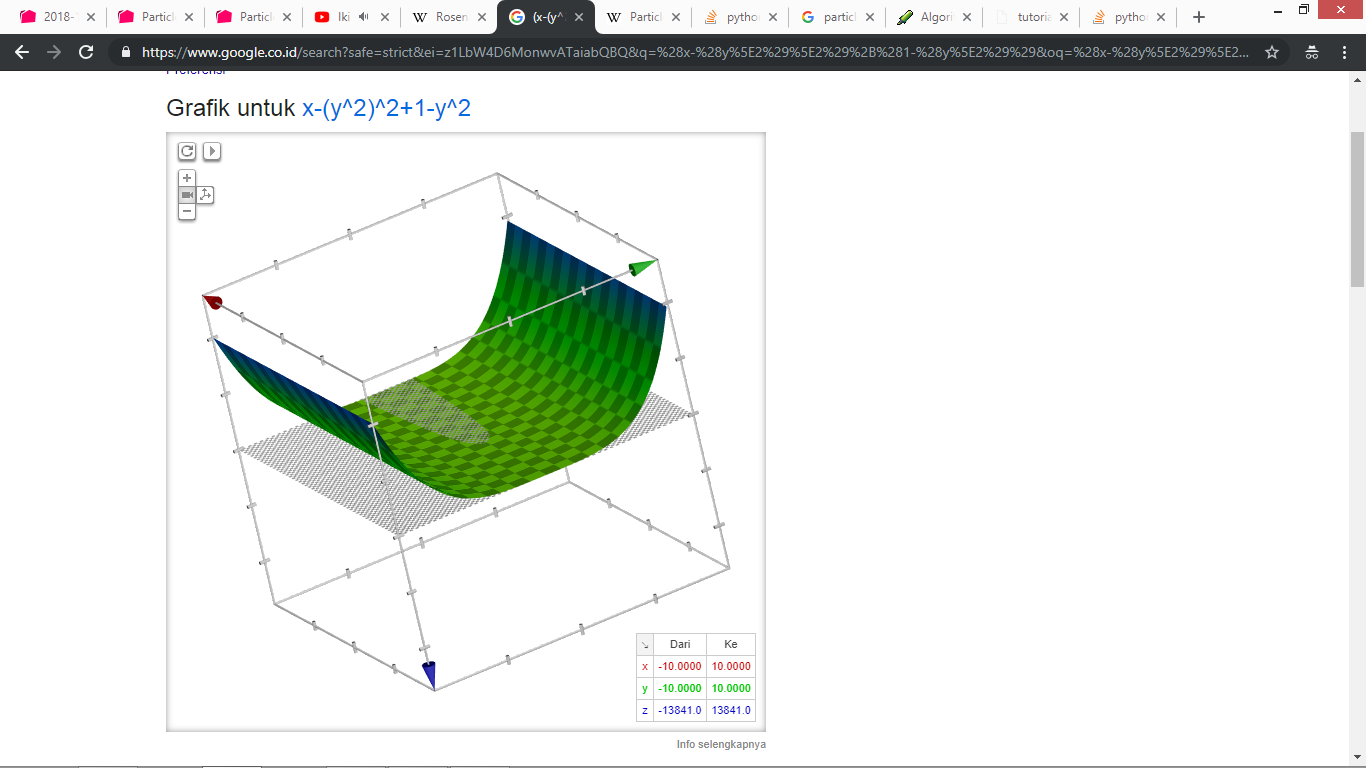
## Fitness Function

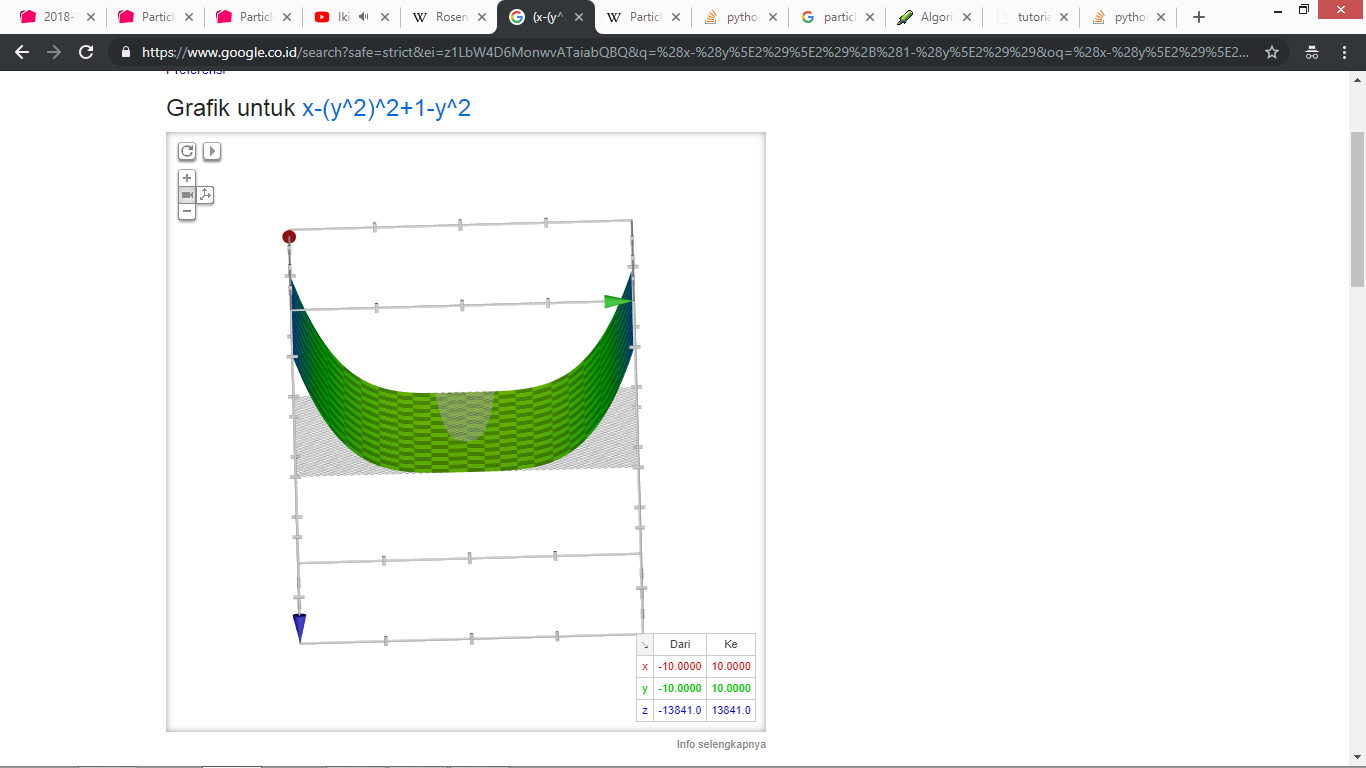
Dalam *Fitness Function* kita akan menghitung seberapa besar individu itu bersaing dengan partikel lainya. Berikut sumber kode untuk melakukan perhitungan *Fitness* :



Dalam algoritma diatas, merupakan rumus dari *Rosenbrock Function*. Dalam penulisan matematika dapat diartikan sebagai berikut :



Dalam Bab Latar belakang saya mengatakan bahwa *Rosenbrock Function* merupakan rumus yang mudah. Karena jika rumus tersebut kita gambarkan kedalamm grafik multidimensional maka akan membentuk fungsi yang sederhana. Berikut visualisasi dari fungsi diatas :

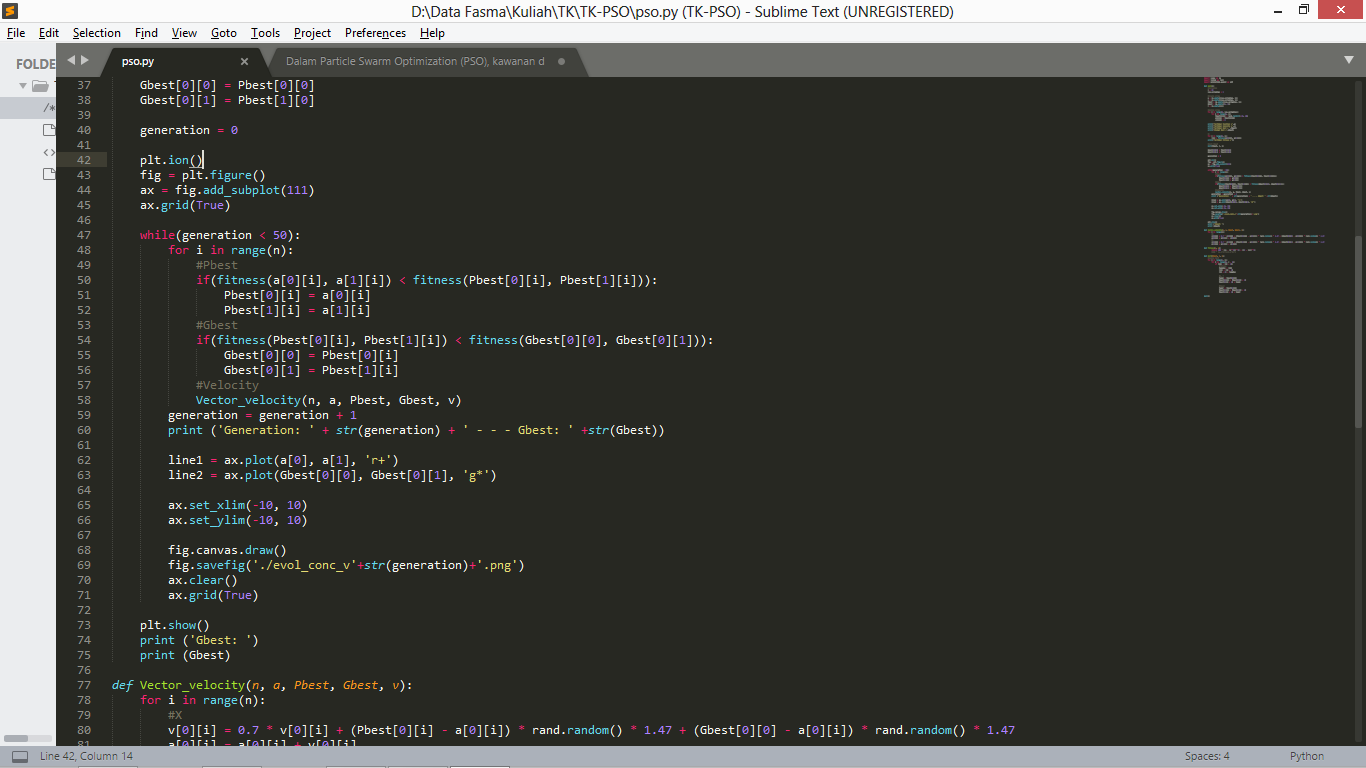


Dari matriks posisi dan velocity diatas, mari kita coba hitung dengan program *Microsoft excel* dan hasilnya sebagai berikut :

Hasil dari program sebagai berikut :

## Iterasi

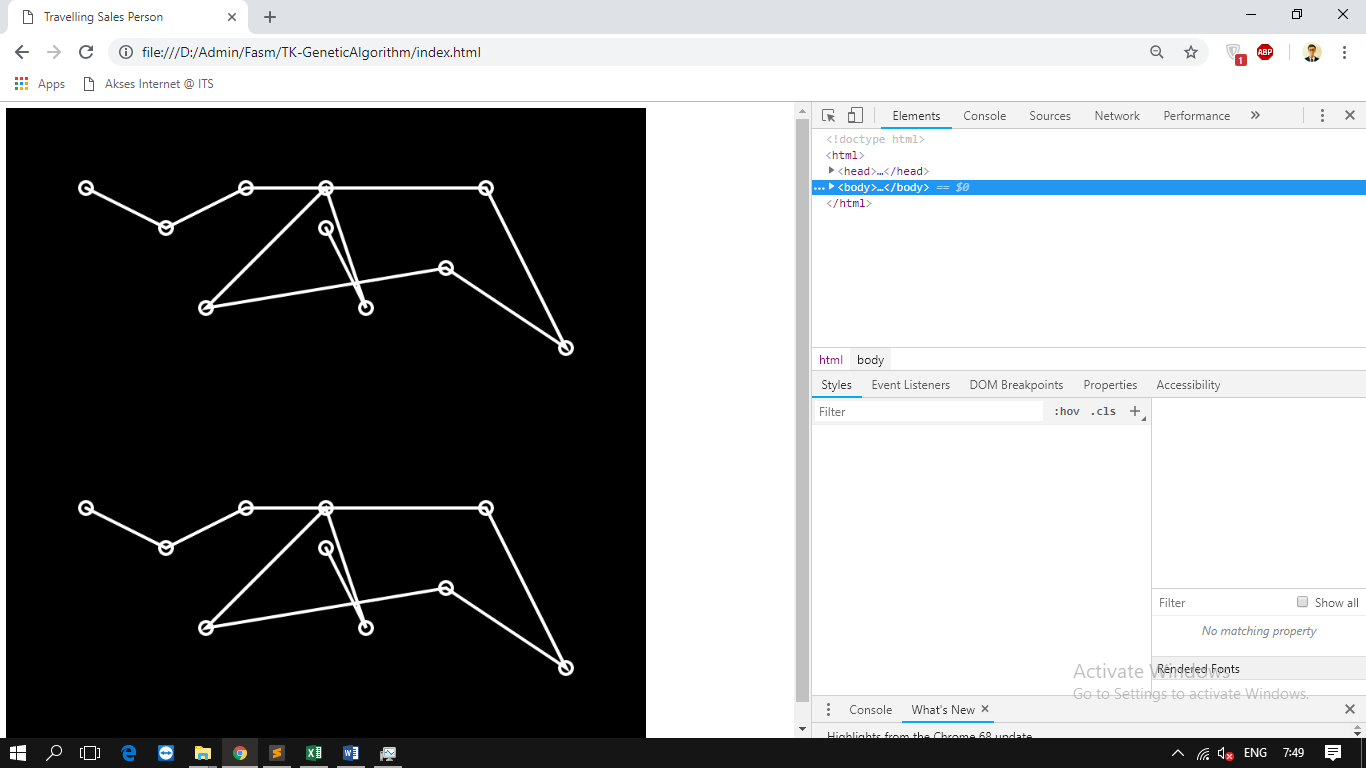
Target dari PSO ini adalah sebanyak 50 generasi. Sesuai dengan flowchart diatas, dalam setiap generasi terdapat pemilihan siapa yang akan menjadi Gbest setiap generasi. Setelah mendapatkan Gbest, masing masing partikel akan dihitung velocitynya.



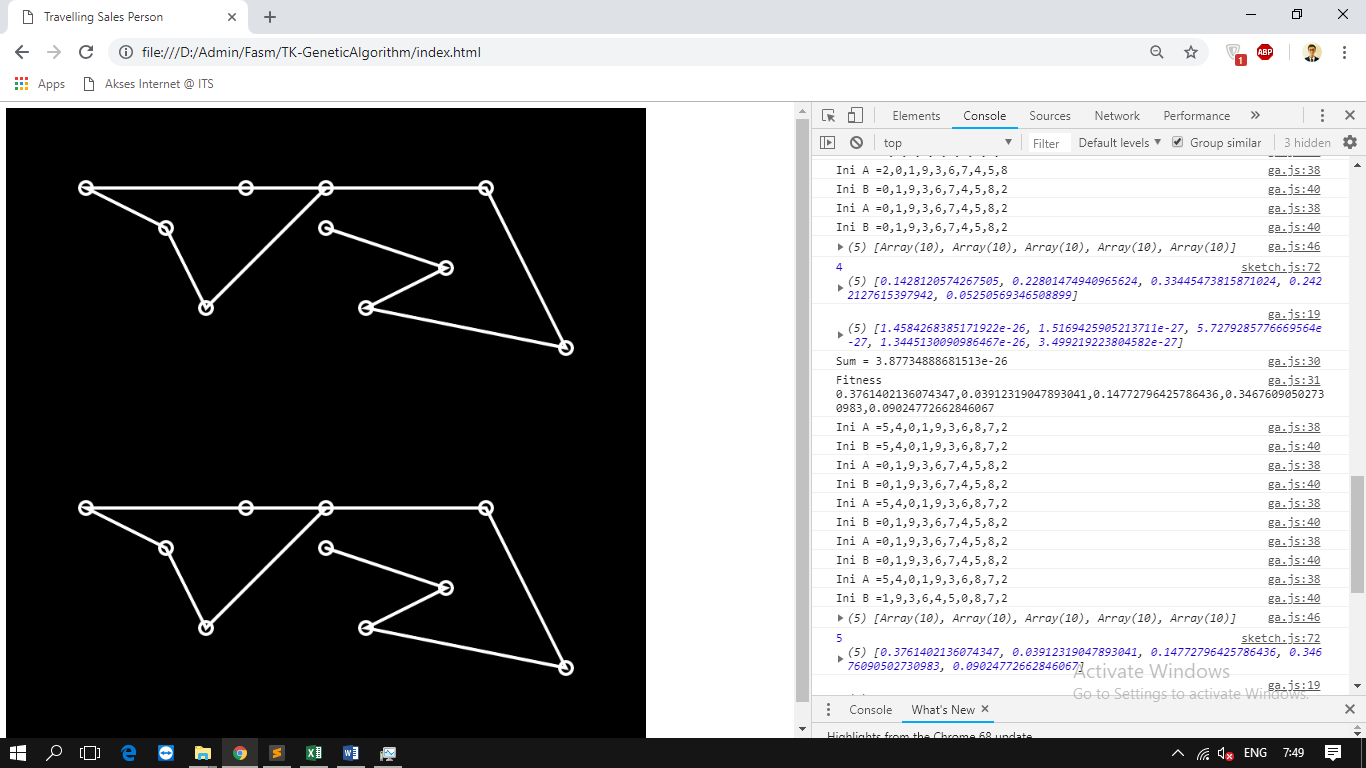
.

# Generation

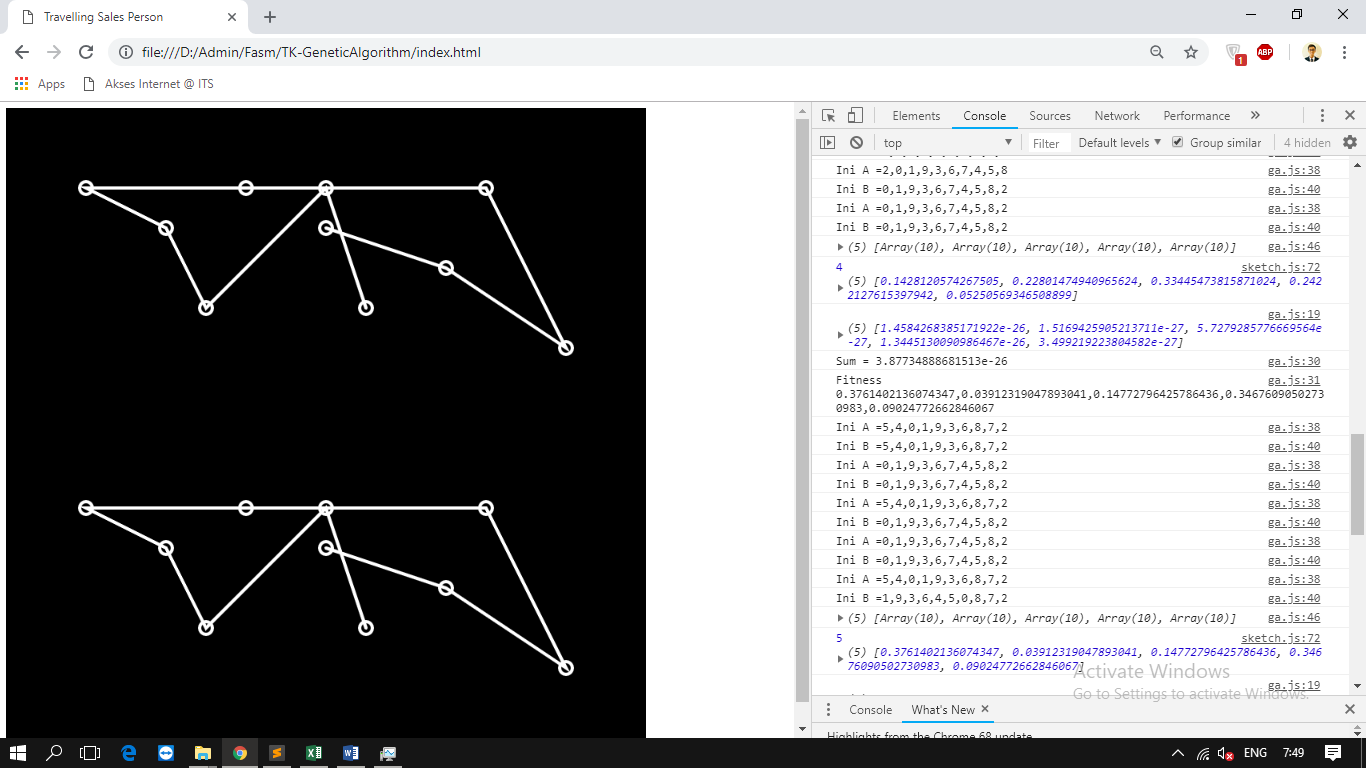
### Generasi pertama



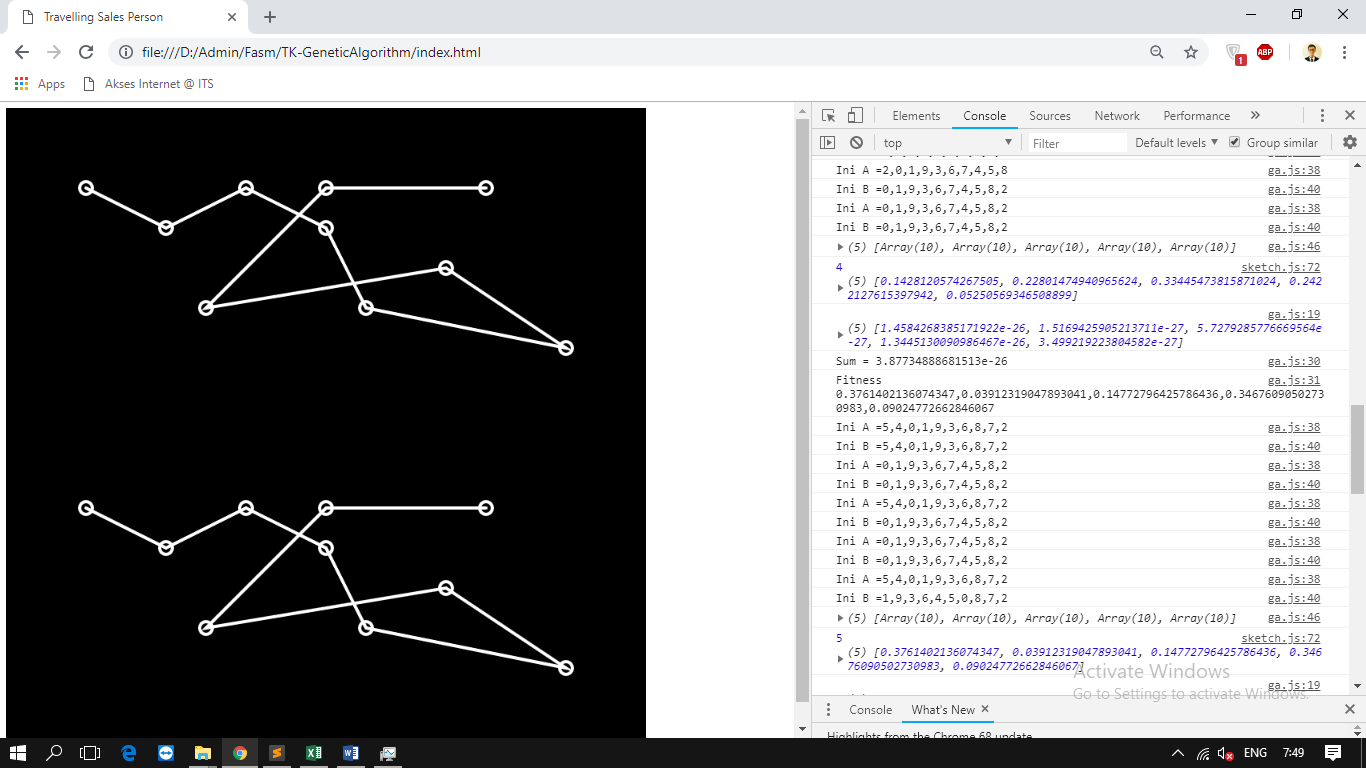
### Generasi kedua



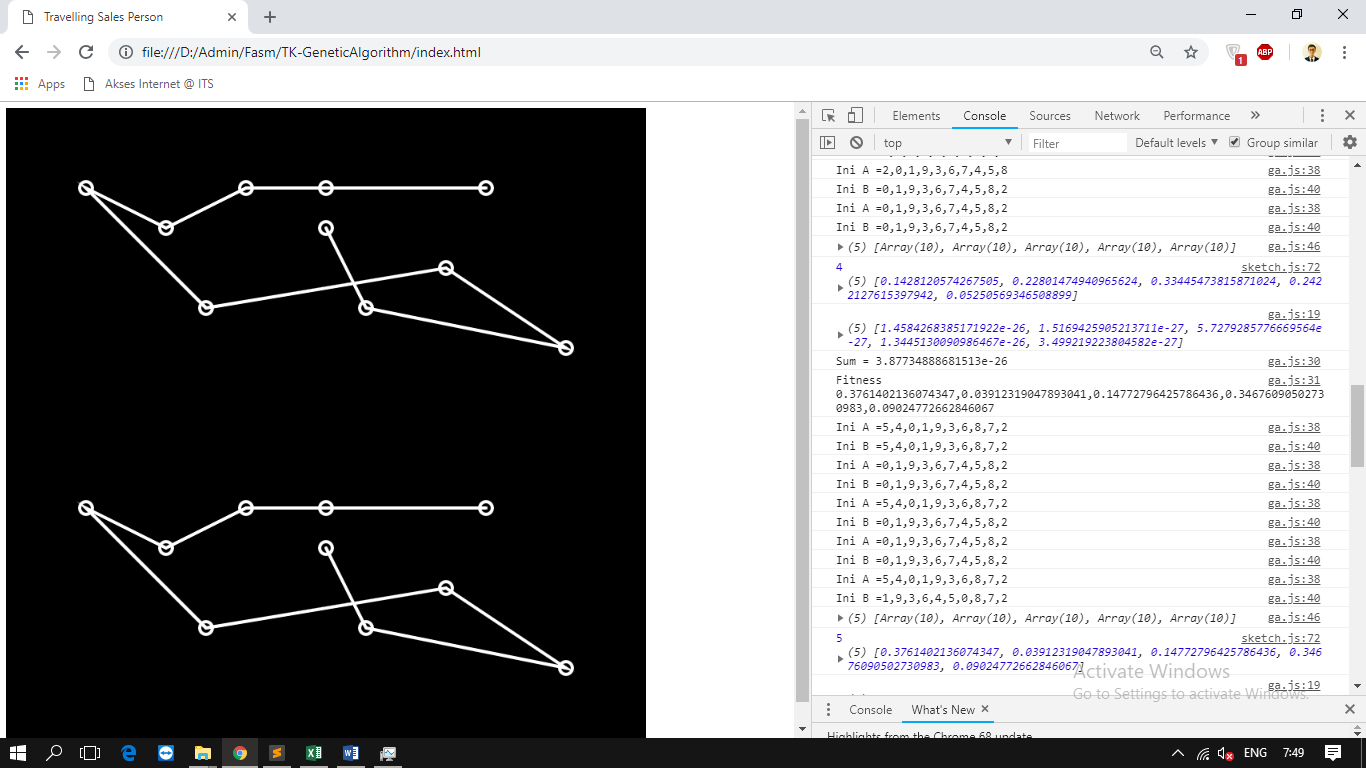
### Generasi ketiga



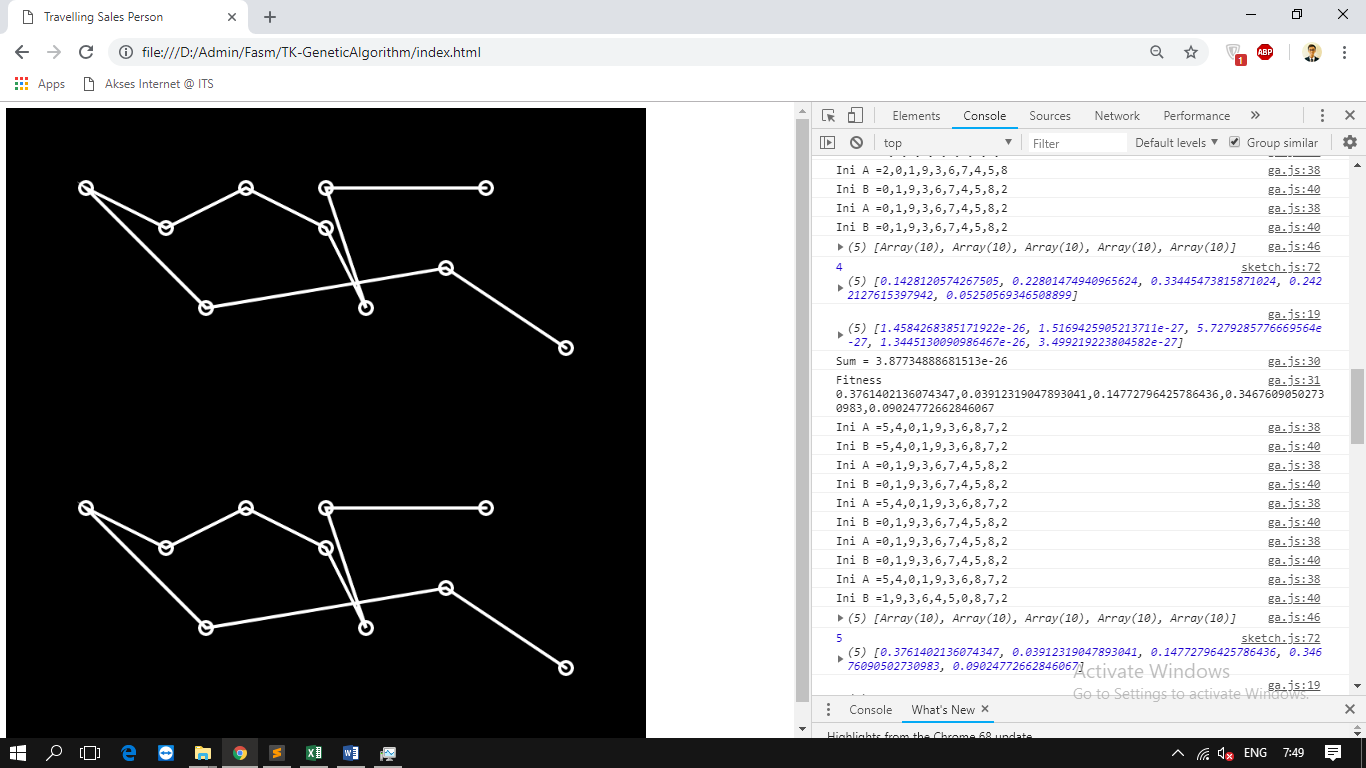
### Generasi ke empat



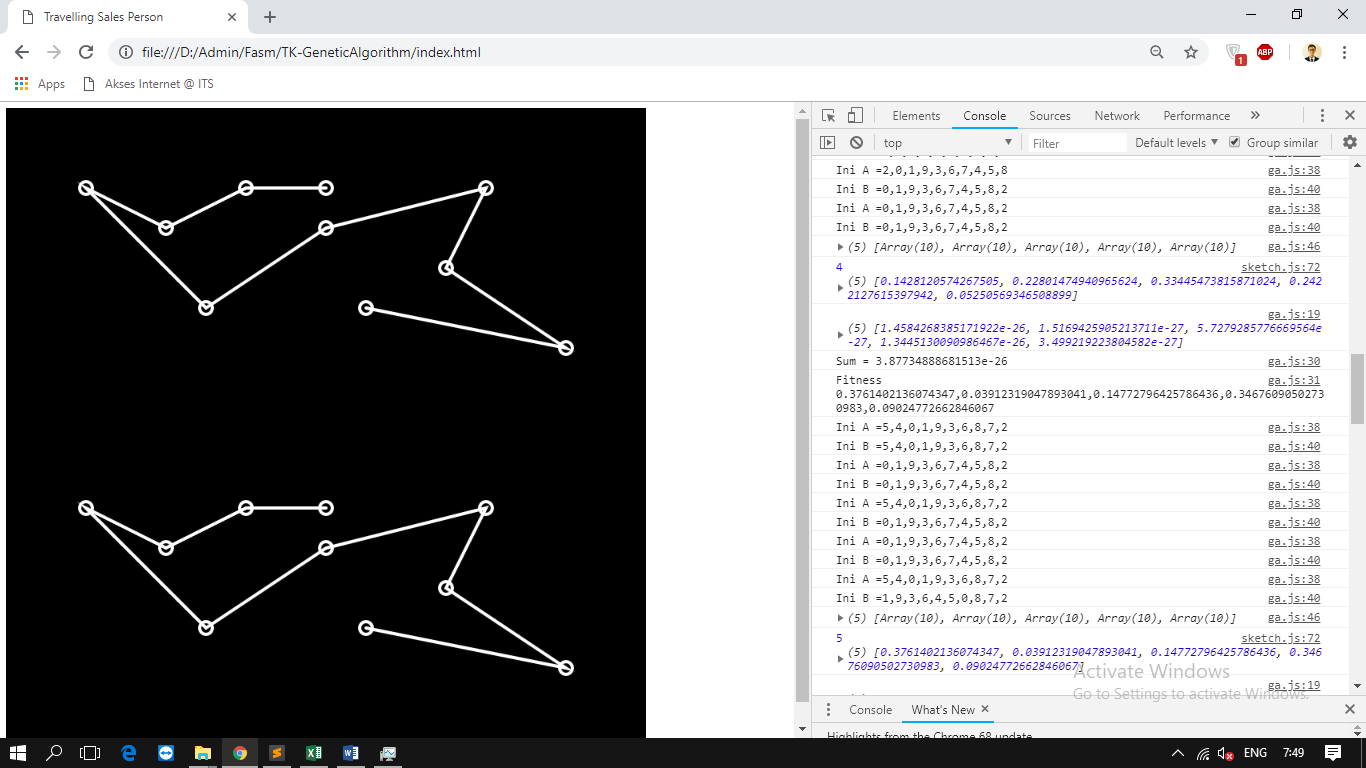
### Generasi ke lima



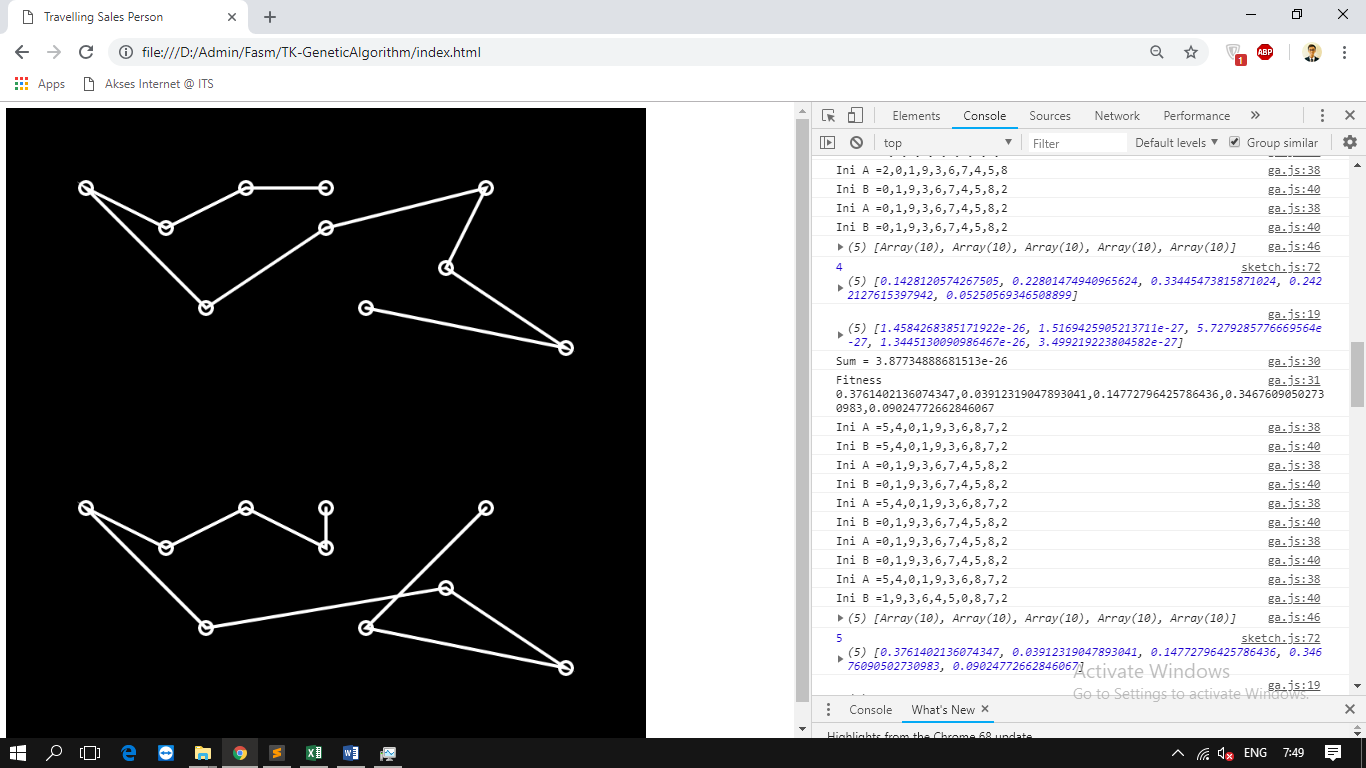
### Generasi ke enam



### Generasi ke tujuh

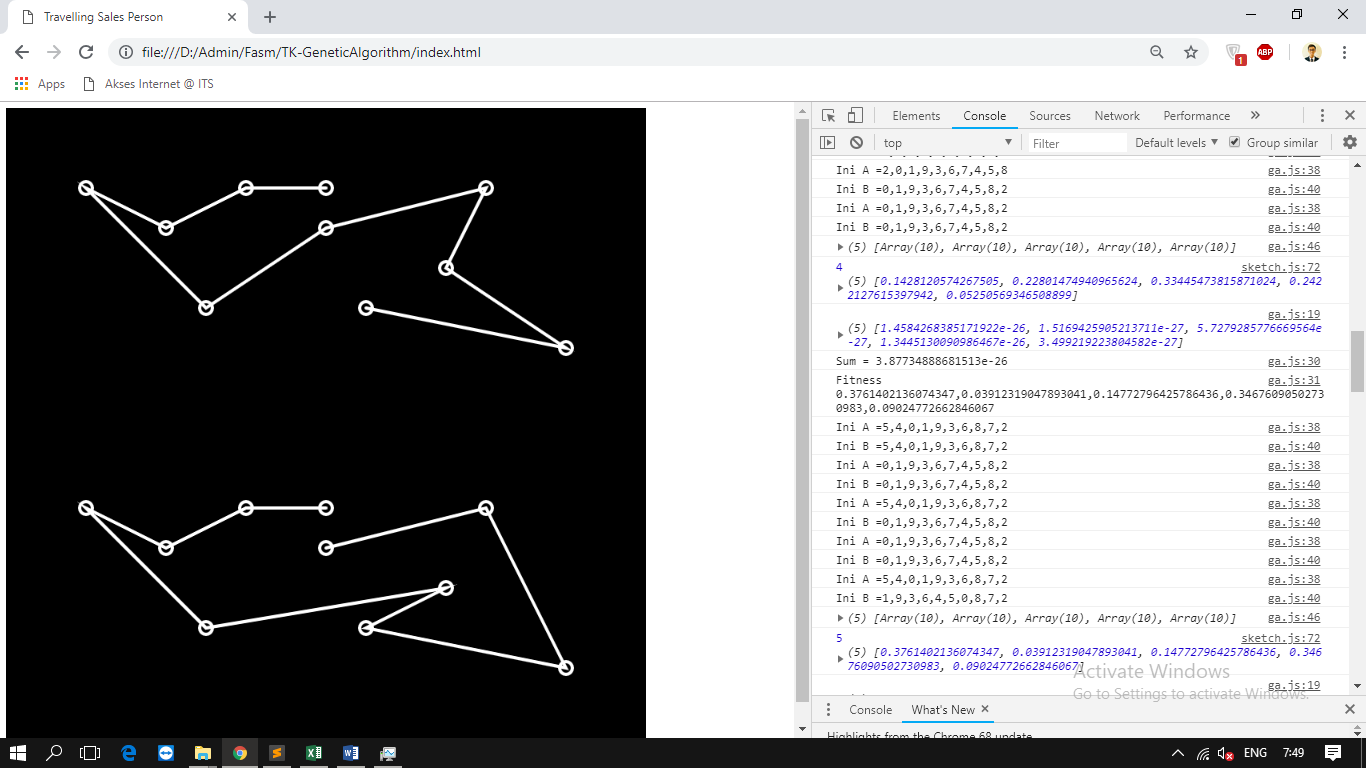


### Generasi ke delapan



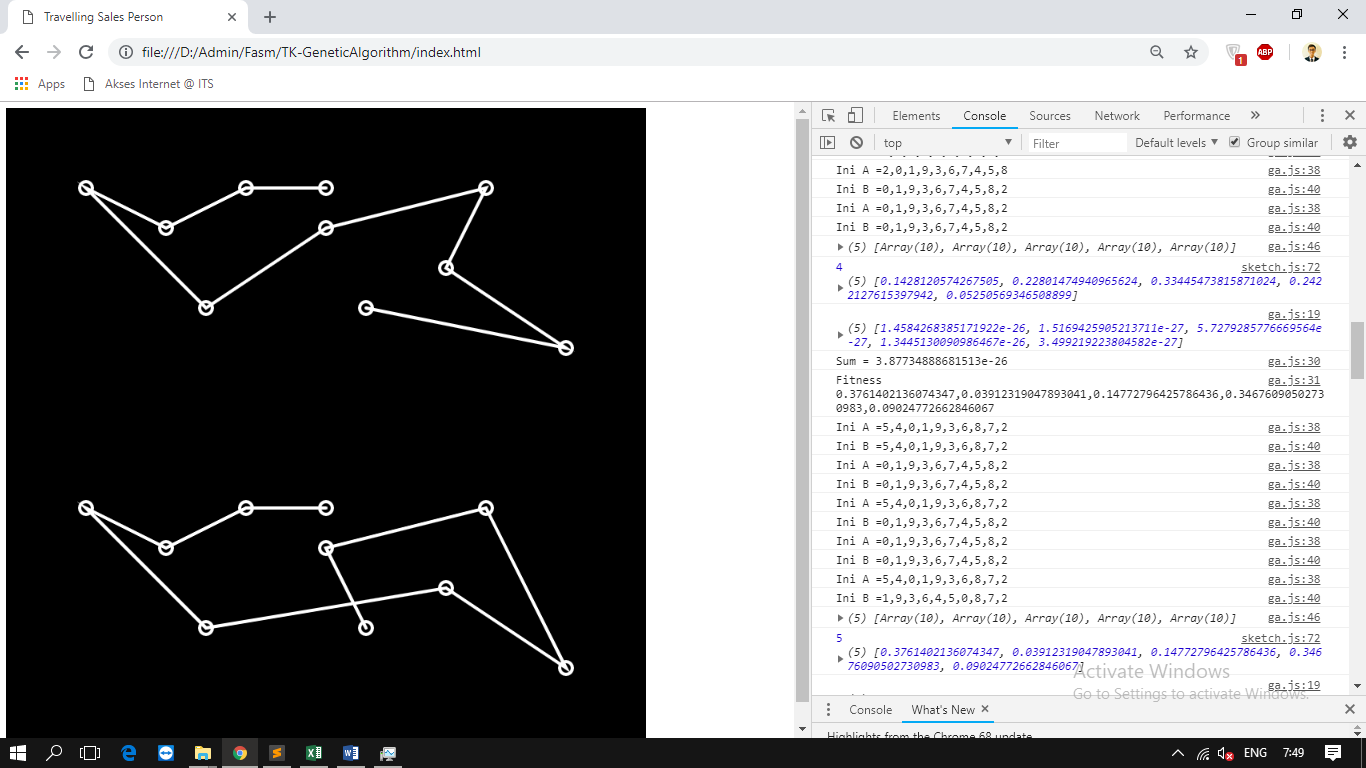
Di generasi kedelapan, graph bagian atas tetap seperti generasi ke tujuh. Hal ini dikarenakan bestRecord dari generasi ke tujuh lebih baik dari pada currentRecord yang dimiliki generasi ke delapan.

### Generasi ke sembilan

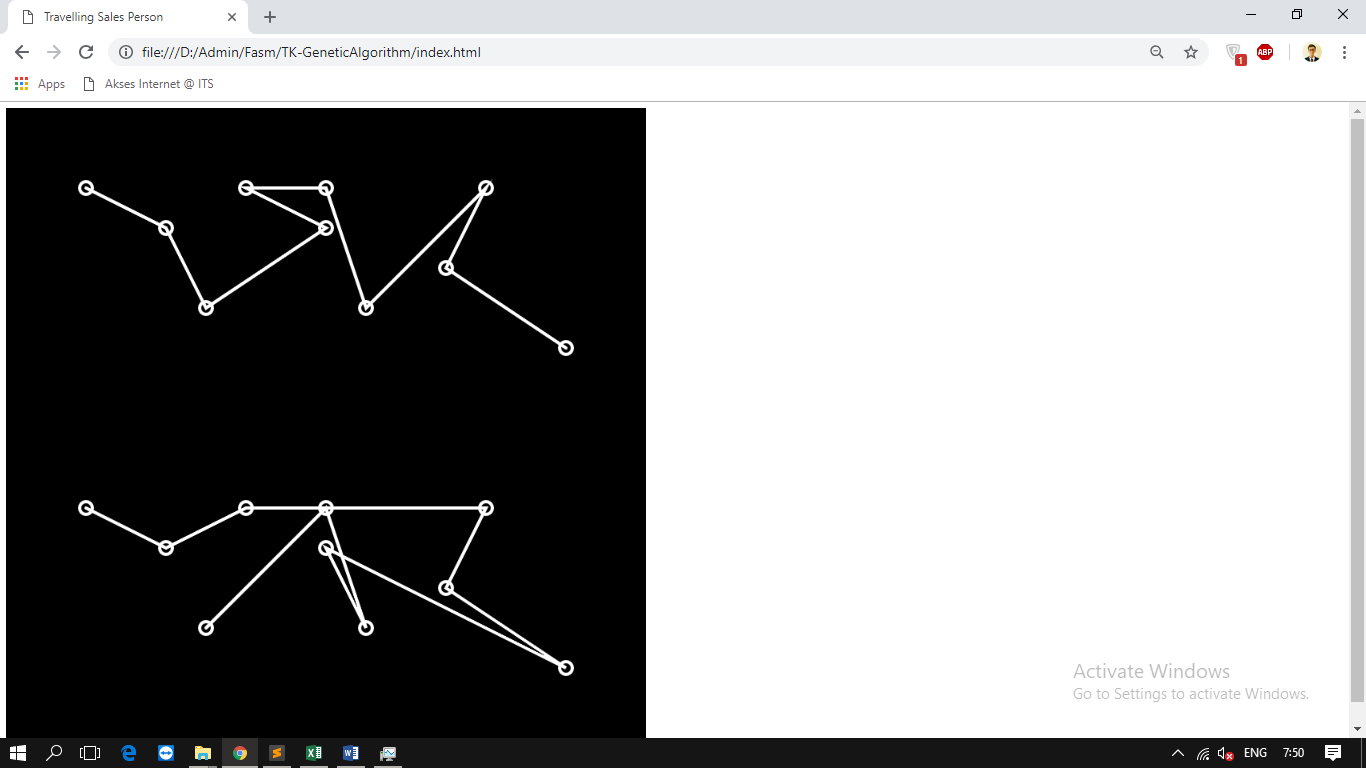


Sama seperti generasi ke delapan, masih belum baik dari generasi sebelumnya

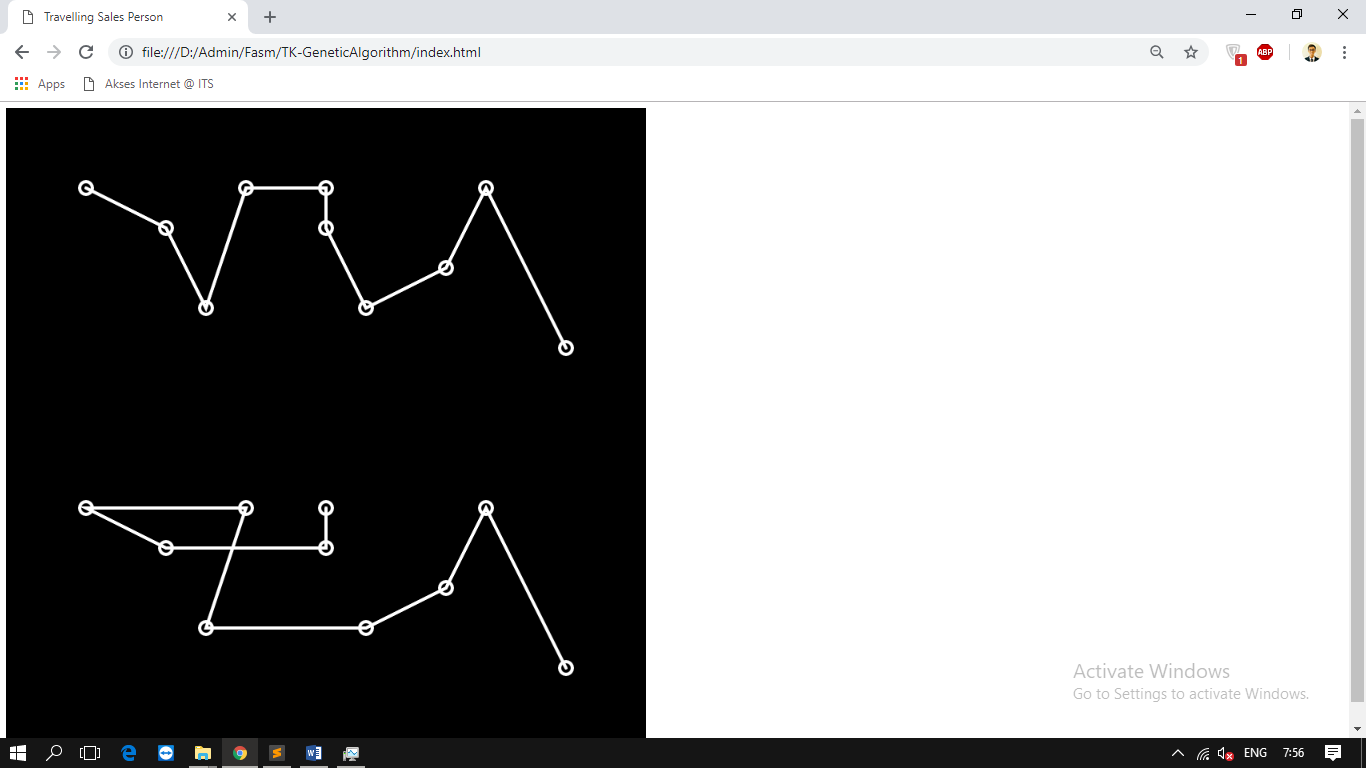
### Generasi ke-10



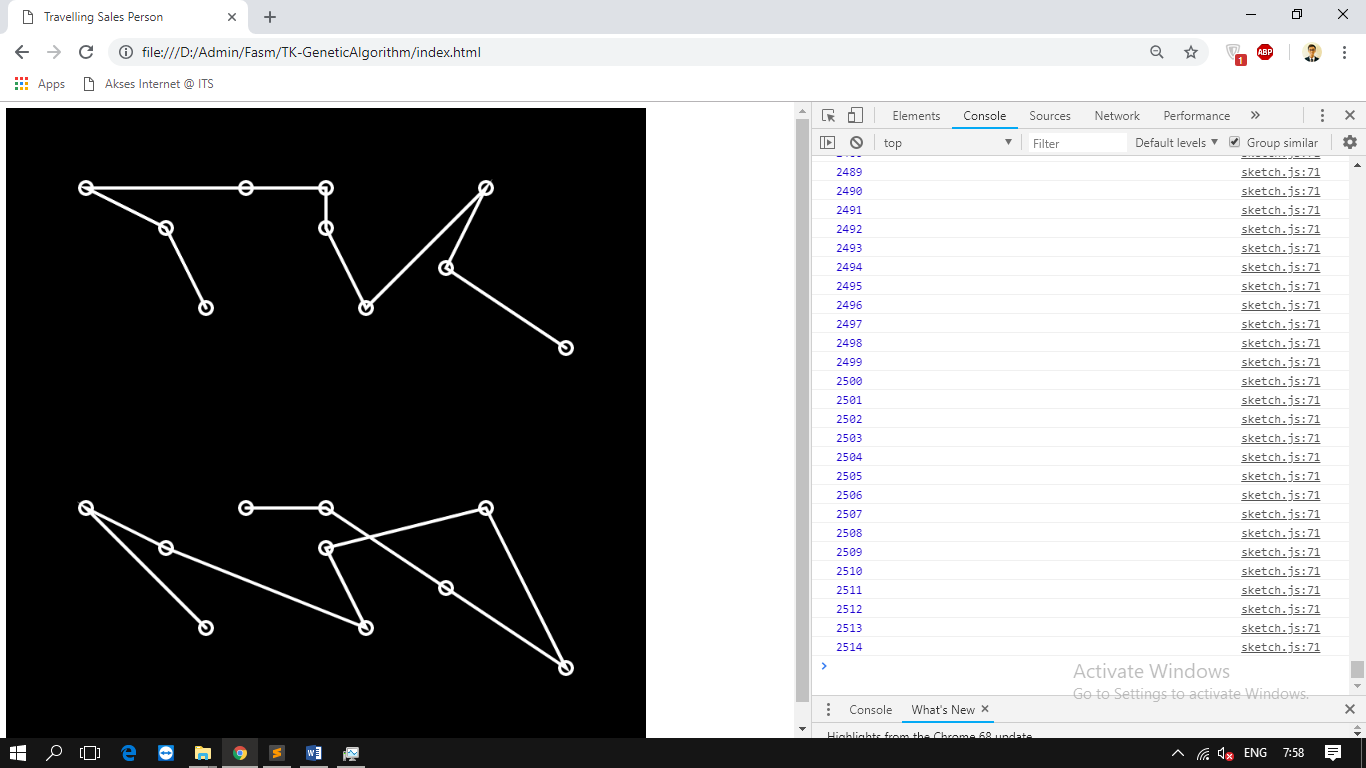
### Generasi ke-15



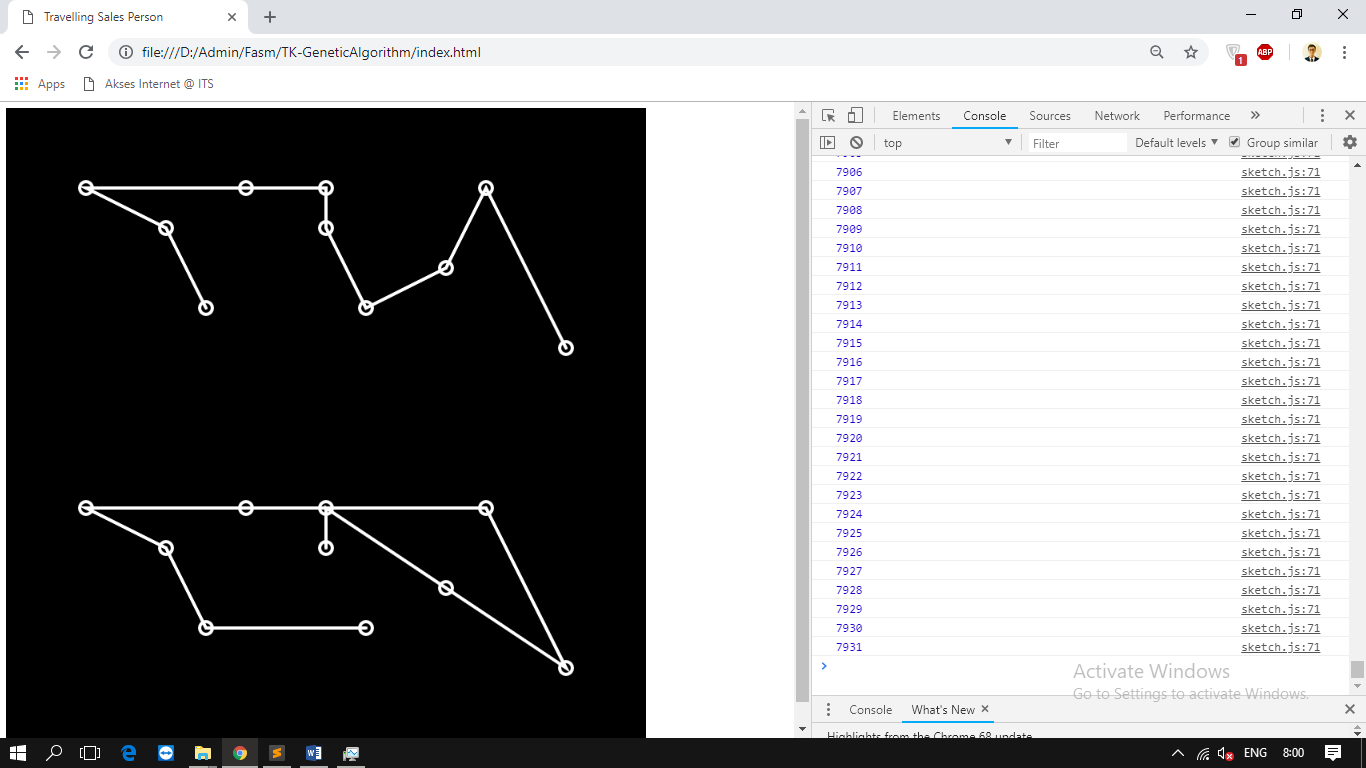
### Generasi ke 50



### Generasi ke 2500



### Generasi ke-7910



Dari hasil diatas, kita dapat simpulkan bahwa memang graph bagian atas sudah sangat pendek jika kita lihat secara sekilas.