*J-INTECH (Journal of Information and Technology)*
**Terakreditasi Kemendikbud SK No. 204/E/KPT/2022**
**E-ISSN: 2580-720X || P-ISSN: 2303-1425**

# Optimasi Jalur Terpendek Menggunakan Algoritma Genetika

Jasmani[1*], Ali Mahmudi[2]

[1,2] *Institut Teknologi Nasional Malang, Teknologi Industri, Teknik Informatika, Malang, Indonesia*

***Informasi Artikel***

Diterima: 16-01-2023
Direvisi: 10-07-2023
Diterbitkan: 11-07-2023

***Kata Kunci***
*lalu lintas, rute, lokasi, Kecerdasan buatan, Algoritma Genetika*

***Email Korespondensi:**
*jhaz2010@gmail.com*

***Abstrak***
Masalah lalu lintas di kota besar cukup rumit untuk dihadapi. Berbagai permasalahan lalu lintas seperti kemacetan lalu lintas, kecelakaan dan lain-lain. Salah satu masalah yang termasuk dalam masalah lalu lintas adalah pencarian jalur atau rute menuju suatu lokasi. *Pathfinding* sangat diperlukan bagi pengguna jalan yang tidak mengetahui jalan mana yang harus ditempuh untuk mencapai tujuannya di suatu kota. Terutama bagi pengguna jalan yang baru pertama kali berkunjung ke kota tersebut. Mungkin juga bagi penduduk kota besar sendiri yang tidak hafal atau tidak tahu jalan mana yang harus ditempuh untuk sampai ke tempat yang diinginkan. Penghematan waktu dan biaya adalah faktor lain yang mengharuskan pengguna jalan menemukan jalur terpendek untuk mencapai tujuan dengan lebih cepat. Oleh karena itu, pencarian jalur terpendek menjadi masalah yang perlu diselesaikan secara komputerisasi dengan kecerdasan buatan (Artificial Intelligence). Dalam kasus yang kami coba bahas menggunakan metode algoritma genetika.

***Abstract***
*Traffic problems in a big city are quite complicated to deal with. Various traffic problems such as traffic jams, accidents and others. One of the problems included in the traffic problem is the search for a path or route to a location. Pathfinding is very necessary for road users who do not know which way to go to get to their destination in a city. Especially for road users who are visiting the city for the first time. It is also possible for residents of big cities themselves who do not know by heart or do not know which route to take to get to a place they want. Time and cost savings are other factors that require road users to find the shortest path to get to their destination more quickly. Because of that, the search for the shortest path becomes a problem that deserves to be solved computerized with artificial intelligence (Artificial Intelligence). In the case that we try to discuss using the genetic algorithm method.*

## 1. Pendahuluan

Untuk menggunakan atau memfungsikan sebuah komputer maka harus terdapat program yang terdistribusi di dalamnya, tanpa program tersebut komputer hanyalah menjadi sebuah kotak yang tak berguna. Dalam kehidupan, sering dilakukan perjalanan dari satu tempat atau kota ke tempat yang lain dengan mempertimbangkan efisiensi, waktu dan biaya sehingga diperlukan ketepatan dalam menentukan jalur terpendek antar suatu kota. Hasil penentuan jalur terpendek akan menjadi pertimbangan dalam pengambilan keputusan untuk menununjukkan jalur yang akan ditempuh. Hasil yang didapatkan juga membutuhkan kecepatan dan keakuratan dengan bantuan komputer (S. Lukas dkk : 2005).

Permasalahan lalu lintas pada suatu kota besar merupakan persoalan yang cukup rumit untuk ditangani. Berbagai permasalahan lalu lintas misalnya, kemacetan, kecelakaan dan lain-lain. Salah satu masalah yang termasuk dalam permasalahan lalu lintas adalah pencarian jalur atau rute menuju suatu lokasi.Pencarian jalur sangat diperlukan bagi pengguna jalan yang tidak tahu jalan mana yang akan dilalui agar sampai ke tempat tujuannya dalam suatu kota. Apalagi bagi pengguna jalan yang baru pertama kalinya mengunjungi kota tersebut. Tidak menutup kemungkinan juga bagi penduduk kota besar itu sendiri yang tidak hafal atau tidak mengetahui jalur mana yang harus dilalui untuk menuju suatu tempat yang mereka kehendaki.

Penghematan waktu dan biaya menjadi faktor lainnya yang mengharuskan pengguna jalan mencari suatu jalur yang terpendek agar lebih cepat sampai ketempat tujuan. Oleh karena hal itulah maka pencarian jalur terpendek menjadi suatu permasalahan yang patut untuk diselesaikan secara komputerisasi dengan kecerdasan buatan.

## 2. Metode Penelitian

Pada prinsip kerja telah dijelaskan penggunaan algoritma genetika untuk menyelesaikan masalah jalur terpendek. Pada bagian ini akan dijelaskan mengenai implementasi algoritma genetika tersebut menggunakan MATLAB 7.5. Masukan dari program enurut Kusumadewi S & Purnomo H (2005) dan Kusumadewi (2003) adalah (1)File *input* yang berisi data rute perjalanan seperti *cost* tiap jalur. (2) *Node* sumber (*source node*). (3) *Node* tujuan (*destination node*). (4) Parameter-parameter algoritma genetika, yaitu ukuran populasi (*popsize*), batas generasi maksimum (banyak iterasi), probabilitas *crossover*, serta probabilitas mutasi. Berikut adalah fungsi-fungsi yang digunakan pada program beserta dengan penjelasannya menurut Pandjaitan (2007).

### 2.1 Gaspp

Algoritma genetika untuk masalah jalur terpendek secara keseluruhan akan dijalankan oleh fungsi gaspp. Fungsi ini akan memanggil beberapa fungsi lain seperti totalCost, randChrom, crossover, mutation, elitism, dan validatePath. Di dalam fungsi seperti reproduksi, *crossover,* dan mutasi.

### 2.2. fitnesV

Fungsi untuk menentukan nilai *fitness* untuk masing-masing kromosom pada suatu populasi.

### 2.3 totalCost

Fungsi untuk menghitung total *cost* semua jalur yang *valid* atau dapat dilalui pada jalur dari *node* sumber ke *node* tujuan.

### 2.4 initChrom

Fungsi untuk melakukan proses *order crossover.*

### 2.5 Mutation

Fungsi untuk melakukan proses *insertion mutation.*

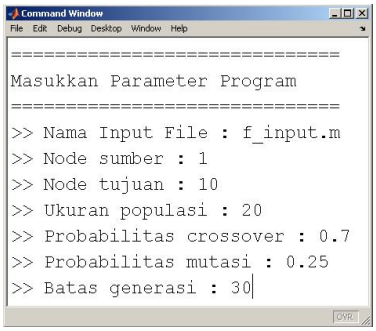
### 2.6 Elitism

Fungsi untuk membangun populasi baru untuk generasi berikutnya.

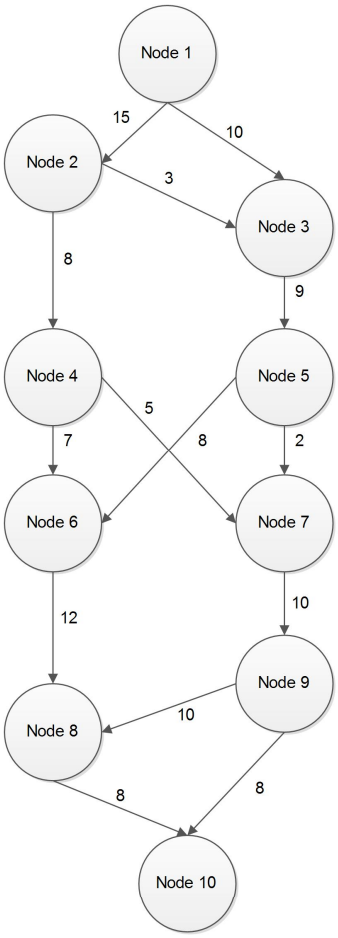
### 2.7 ValidatePate

Fungsi untuk memeriksa apakah jalur yang terkandung dalam kromosom adalah *valid* atau tidak.

Pemanggilan program dilakukan dengan cara mengetik “gaspp” pada MATLAB *Command Window.* Program akan berhenti apabila telah mencapai batas generasi yang telah ditentukan oleh pengguna. Contoh pemanggilan program saat memasukkan data dan hasil keluaran yang diperoleh pada contoh. Berikut ini contoh persoalan masalah jalur terpendek dari suatu rute perjalanan yang terdapat 10 simpul. *Node* sumber adalah *node* 1 dan *node* akhir adalah *node* 10, maka panjang kromosom atau banyaknya gen dalam satu kromosom adalah 10. Pada contoh ini, masalah akan diselesaikan menggunakan algoritma genetika. Ketik “gaspp” pada MATLAB *Command Window* dan tekan E*nter.* Maka program akan meminta pengguna untuk memasukkan *input* berupa *node* sumber, *node* tujuan, dan parameter-parameter algoritma genetika.




*Gambar 1. Tampilan Command Window Data Masukan untuk Contoh 1*




*Gambar 2. Node yang Menggambarkan Rute Perjalanan Sebelum Dilakukan Proses Generasi*

Dari gambar node di atas, dapat kita ambil sebuah data pada tabel berikut dengan melakukan proses generasi awal sampai ke evaluasi nilai *fitness.*

*Tabel 1. Proses generasi awal sampai evaluasi nilai fitness.*

| Kromosom | Representasi Kromosom | Nilai *cost* (total jarak) | Nilai *Fitness* |
|---|---|---|---|
| 1 | **1-2-4-7-9-8-10**-3-6-5 | 15+8+5+10+10+6=54 | 0.0185 |
| 2 | **1-2-4-7-9-10**-5-8-6-3 | 15+8+5+10+8=46 | 0.0217 |
| 3 | **1-3-5-6-8-9-10-**4-7-2 | 10+9+6+12+10+8=55 | 0.0182 |
| 4 | **1-3-5-7-9-10**-4-6-2-8 | 10+9+2+10+8=39 | 0.0256 |
| 5 | **1-3-5-7-9-10-**6-4-8-2 | 10+9+2+10+8=39 | 0.0256 |
| 6 | **1-2-4-7-9-10**-5-3-8-6 | 15+8+5+10+8=46 | 0.0217 |
| 7 | **1-3-2-4-7-5-6-8-9-10** | 10+3+8+5+2+6+12+10+8=64 | 0.0156 |
| 8 | **1-2-4-6-8-10**-9-7-5-3 | 15+8+7+12+6=48 | 0.0208 |
| 9 | **1-2-4-7-9-10-**3-8-5-6 | 15+8+5+10+8=46 | 0.0217 |
| 10 | **1-2-4-7-5-6-8-9-10-**3 | 15+8+5+2+6+12+10+8=66 | 0.0152 |

Dari tabel diatas, tampak bahwa kromosom 4 dan 5 memiliki nilai *fitness* tertinggi, sedangkan kromosom 10 memiliki nilai *fitness* terendah. Kemudian kita proses data tabel di atas dengan seleksi *roda roulette.*

*Tabel 2. Populasi kromosom orang tua setelah proses roda roulette.*

| Bilangan Acak | Kromosom Orang Tua | Kromosom | Representasi Kromosom |
|---|---|---|---|
| 85.2042 | 1 | 9 | **1-2-4-7-9-10-**3-8-5-6 |
| 82.512 | 2 | 9 | **1-2-4-7-9-10-**3-8-5-6 |
| 90.505 | 3 | 9 | **1-2-4-7-9-10-**3-8-5-6 |
| 80.2401 | 4 | 8 | **1-2-4-6-8-10**-9-7-5-3 |
| 60.1012 | 5 | 6 | **1-2-4-7-9-10**-5-3-8-6 |
| 95.201 | 6 | 10 | **1-2-4-7-5-6-8-9-10-**3 |
| 39.2014 | 7 | 4 | **1-3-5-7-9-10**-4-6-2-8 |
| 7.5001 | 8 | 1 | **1-2-4-7-9-8-10**-3-6-5 |
| 62.5101 | 9 | 6 | **1-2-4-7-9-10**-5-3-8-6 |
| 63.2431 | 10 | 6 | **1-2-4-7-9-10**-5-3-8-6 |

Setelah terbentuk seperti tabel di atas, kita akan melakukan proses *crossover* yang memvalidasi jalur yang terkandung di dalamnya, karena bisa jadi *offspring* (anak) yang terbentuk mempresentasikan jalur yang tidak valid.

*Offspring* ketujuh sampai kesepuluh mengandung jalur yang valid, dalam hal ini adalah jalur yang dapat mencapai node tujuan yaitu node 10 dari node sumber yaitu node 1. Sementara *offspring* 1 sampai *offspring 6* mengandung jalur yang tidak valid. *Offspring* yang mengandung jalur tidak valid tidak akan digunakan untuk generasi selanjutnya (G. Nagib:2010). Setelah itu kita memilah bagian yang valid, kemudian *offspring* tersebut dapat menjadi bagian dari generasi berikutnya.

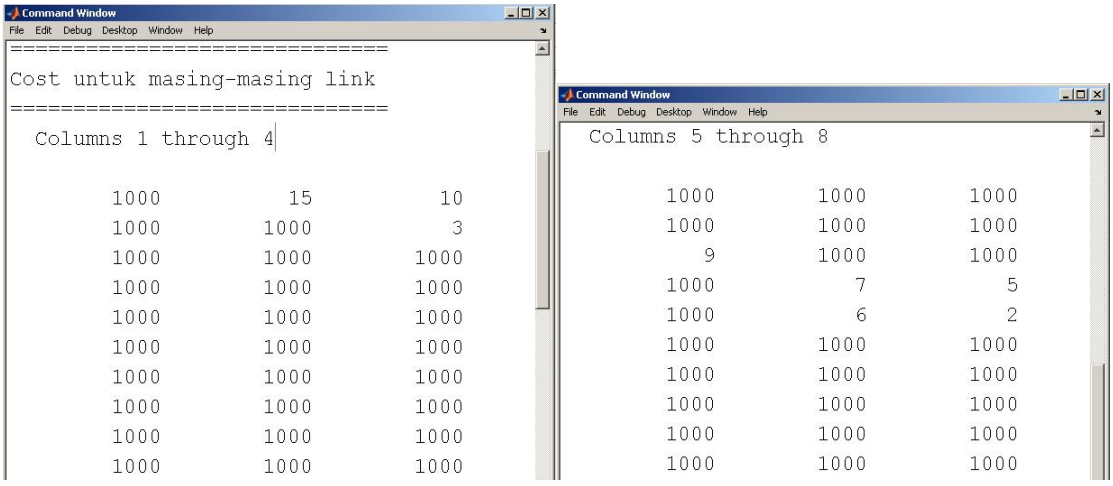
*Tabel 3. Populasi offspring hasil crossover*

| *Offspring* | Representasi Kromosom | Representasi Jalur |
|---|---|---|
| *1* | 1-4-6-9-7-10-3-8-5-2 | Tidak Valid |
| *2* | 1-2-4-3-8-10-9-7-5-6 | Tidak Valid |
| *3* | 1-3-4-6-8-10-9-5-2-7 | Tidak Valid |
| *4* | 1-8-4-7-9-10-3-5-2-6 | Tidak Valid |
| *5* | 1-9-10-7-5-6-3-8-2-4 | Tidak Valid |
| *6* | 1-5-6-7-9-10-8-3-2-4 | Tidak Valid |
| *7* | 1-2-4-7-9-8-10-3-6-5 | Valid |
| *8* | 1-3-5-7-9-10-4-6-2-8 | Valid |
| *9* | 1-2-4-7-9-10-5-3-8-6 | Valid |
| *10* | 1-2-4-7-9-10-5-3-8-6 | Valid |

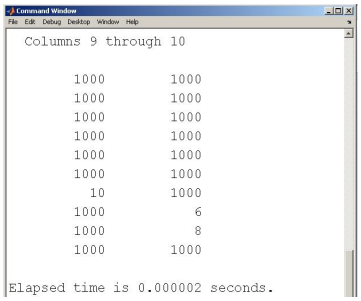
*Tabel 4. Populasi Offspring Hasil Mutasi*

| *Offspring* | Representasi Kromosom | Kromosom Hasil Mutasi | Representasi Jalur |
|---|---|---|---|
| **1** | 1-4-6-9-7-10-3-8-5-2 | 1-4-6-9-7-10-3-8-5-2 | Tidak Valid |
| ***2*** | 1-2-4-3-8-10-9-7-5-6 | 1-6-2-4-3-8-10-9-7-5 | Tidak Valid |
| ***3*** | 1-3-4-6-8-10-9-5-2-7 | 1-3-4-6-8-10-9-5-2-7 | Tidak Valid |
| ***4*** | 1-8-4-7-9-10-3-5-2-6 | 1-4-7-9-10-8-3-5-2-6 | Tidak Valid |
| ***5*** | 1-9-10-7-5-6-3-8-2-4 | 1-9-10-7-5-6-3-8-2-4 | Tidak Valid |
| ***6*** | 1-5-6-7-9-10-8-3-2-4 | 1-5-6-7-9-10-8-3-2-4 | Tidak Valid |
| ***7*** | 1-2-4-7-9-8-10-3-6-5 | 1-2-4-7-9-8-10-3-6-5 | Valid |
| ***8*** | 1-3-5-7-9-10-4-6-2-8 | 1-3-5-7-9-10-4-6-2-8 | Valid |
| ***9*** | 1-2-4-7-9-10-5-3-8-6 | 1-2-4-7-9-10-5-3-8-6 | Valid |
| ***10*** | 1-2-4-7-9-10-5-3-8-6 | 1-2-4-7-910-5-3-8-6 | Valid |

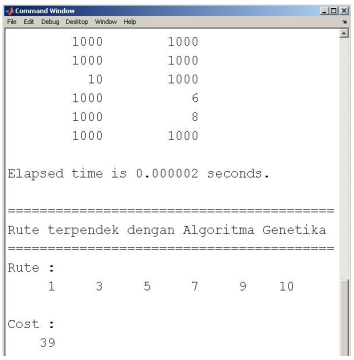
Adapun nilai yang digunakan untuk memasukkanparameter algoritma genetika adalah ukuran populasi 20, probabilitas *crossover* 0.7, probabilitas mutasi 0.25, batas generasi 30, sedangkan node sumber 1, dan node tujuan 10 (Gambar 1). Program kemudian mengeluarkan solusi akhir yang didapat dan grafik yang menampilkan perubahan nilai *fitness* terbaik terhadap pertambahan generasi (Gambar 3).



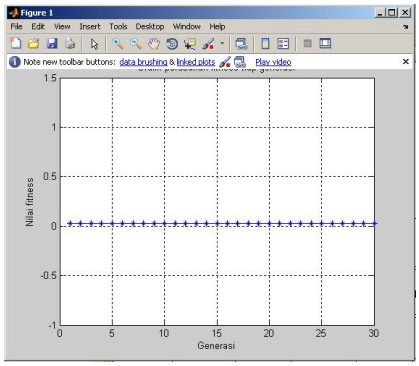

(a) (b)



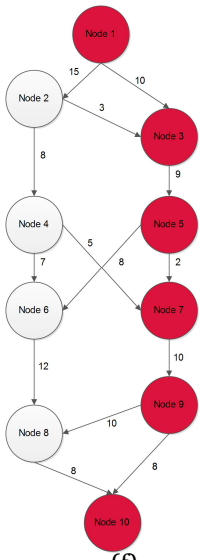

(b)




(d)




(e)




(f)

*Gambar 3. Solusi berupa perubahan nilai fitness*

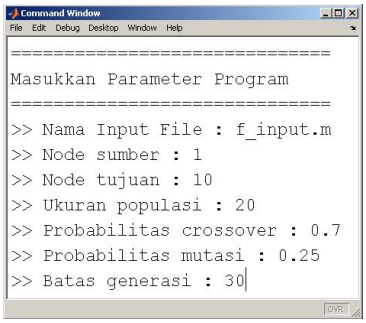
Dari gambar 3 terlihat bahwa solusi untuk contoh 1 adalah jalur 1-3-5-7-9-10 yang memiliki total *cost39.* Grafik yang ditujukkan pada gambar 3.e dan 3.f menunjukkan nilai *fitness* terbaik pada tiap generasi. Dari grafik tersebut terlihat bahwa solusi optimal telah diperoleh sejak generasi pertama yaitu dengan nilai *fitness* 0.0256. Selanjutnya menurut Sivanandam (2008) akan dibahas beberapa percobaan yang dilakukan untuk melihat pengaruh parameter-parameter genetik seperti ukuran populasi, probabilitas *crossover,* dan probabilitas mutasi terhadap kinerja algoritma genetika.

## 3. Hasil dan Pembahasan

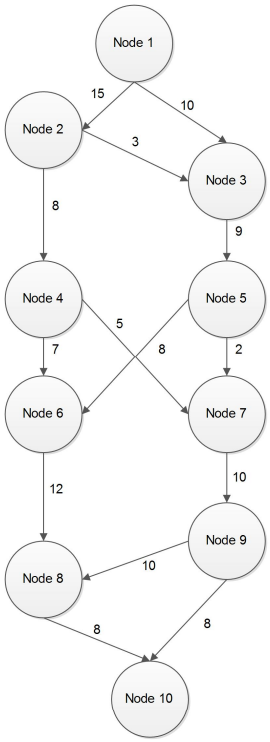
Pemanggilan program dilakukan dengan cara mengetik "gaspp" pada MATLAB *Command Window.* Program akan berhenti apabila telah mencapai batas generasi yang telah ditentukan oleh pengguna. Contoh pemanggilan program saat memasukkan data dan hasil keluaran yang diperoleh pada contoh.

Berikut ini contoh persoalan masalah jalur terpendek dari suatu rute perjalanan yang terdapat 10 simpul. *Node* sumber adalah *node* 1 dan *node* akhir adalah *node* 10, maka panjang kromosom atau banyaknya gen dalam satu kromosom adalah 10. Pada contoh ini, masalah akan diselesaikan menggunakan algoritma genetika. Ketik "gaspp" pada MATLAB *Command Window* dan tekan E*nter.* Maka program akan meminta

pengguna untuk memasukkan *input* berupa *node* sumber, *node* tujuan, dan parameter-parameter algoritma genetika.




*Gambar 4. Tampilan Command Window data masukan untuk contoh 1*




*Gambar 5. Node yang Menggambarkan Rute Perjalanan Sebelum Dilakukan Proses Generasi*

Dari gambar node diatas, dapat kita ambil sebuah data pada tabel berikut dengan melakukan proses generasi awal sampai ke evaluasi nilai *fitness.*

*Tabel 5. Proses Generasi Awal Sampai Evaluasi Nilai Fitness*

| Kromosom | Representasi Kromosom | Nilai *cost* (total jarak) | Nilai *Fitness* |
|---|---|---|---|
| 1 | **1-2-4-7-9-8-10**-3-6-5 | 15+8+5+10+10+6=54 | 0.0185 |
| *2* | **1-2-4-7-9-10**-5-8-6-3 | 15+8+5+10+8=46 | 0.0217 |
| *3* | **1-3-5-6-8-9-10-**4-7-2 | 10+9+6+12+10+8=55 | 0.0182 |
| *4* | **1-3-5-7-9-10**-4-6-2-8 | 10+9+2+10+8=39 | 0.0256 |
| *5* | **1-3-5-7-9-10-**6-4-8-2 | 10+9+2+10+8=39 | 0.0256 |
| *6* | **1-2-4-7-9-10**-5-3-8-6 | 15+8+5+10+8=46 | 0.0217 |
| *7* | **1-3-2-4-7-5-6-8-9-10** | 10+3+8+5+2+6+12+10+8=64 | 0.0156 |

| | | | |
|---|---|---|---|
| *8* | **1-2-4-6-8-10**-9-7-5-3 | 15+8+7+12+6=48 | 0.0208 |
| *9* | **1-2-4-7-9-10-**3-8-5-6 | 15+8+5+10+8=46 | 0.0217 |
| *10* | **1-2-4-7-5-6-8-9-10-**3 | 15+8+5+2+6+12+10+8=66 | 0.0152 |

Dari tabel diatas, tampak bahwa kromosom 4 dan 5 memiliki nilai *fitness* tertinggi, sedangkan kromosom 10 memiliki nilai *fitness* terendah. Kemudian kita proses data tabel di atas dengan seleksi *roda roulette.*

*Tabel 6. Populasi Kromosom Orang Tua Setelah Proses Roda Roulette*

| Bilangan Acak | Kromosom Orang Tua | Kromosom | Representasi Kromosom |
|---|---|---|---|
| 85.2042 | 1 | 9 | **1-2-4-7-9-10-**3-8-5-6 |
| 82.512 | 2 | 9 | **1-2-4-7-9-10-**3-8-5-6 |
| 90.505 | 3 | 9 | **1-2-4-7-9-10-**3-8-5-6 |
| 80.2401 | 4 | 8 | **1-2-4-6-8-10**-9-7-5-3 |
| 60.1012 | 5 | 6 | **1-2-4-7-9-10**-5-3-8-6 |
| 95.201 | 6 | 10 | **1-2-4-7-5-6-8-9-10-**3 |
| 39.2014 | 7 | 4 | **1-3-5-7-9-10**-4-6-2-8 |
| 7.5001 | 8 | 1 | **1-2-4-7-9-8-10**-3-6-5 |
| 62.5101 | 9 | 6 | **1-2-4-7-9-10**-5-3-8-6 |
| 63.2431 | 10 | 6 | **1-2-4-7-9-10**-5-3-8-6 |

Setelah terbentuk seperti tabel di atas, kita akan melakukan proses *crossover* yang memvalidasi jalur yang terkandung di dalamnya, karena bisa jadi *offspring* (anak) yang terbentuk mempresentasikan jalur yang tidak valid.

*Tabel 7. Populasi Offspring Hasil Crossover*

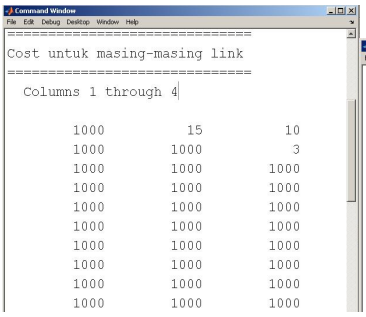
| *Offspring* | Representasi Kromosom | Representasi Jalur |
|---|---|---|
| 1 | 1-4-6-9-7-10-3-8-5-2 | Tidak Valid |
| *2* | 1-2-4-3-8-10-9-7-5-6 | Tidak Valid |
| *3* | 1-3-4-6-8-10-9-5-2-7 | Tidak Valid |
| *4* | 1-8-4-7-9-10-3-5-2-6 | Tidak Valid |
| *5* | 1-9-10-7-5-6-3-8-2-4 | Tidak Valid |
| *6* | 1-5-6-7-9-10-8-3-2-4 | Tidak Valid |
| *7* | 1-2-4-7-9-8-10-3-6-5 | Valid |
| ***8*** | 1-3-5-7-9-10-4-6-2-8 | Valid |
| ***9*** | 1-2-4-7-9-10-5-3-8-6 | Valid |
| ***10*** | 1-2-4-7-9-10-5-3-8-6 | Valid |

*Offspring* ketujuh sampai kesepuluh mengandung jalur yang valid, dalam hal ini adalah jalur yang dapat mencapai node tujuan yaitu node 10 dari node sumber yaitu node 1. Sementara *offspring* 1 sampai *offspring 6* mengandung jalur yang tidak valid. *Offspring* yang mengandung jalur tidak valid tidak akan digunakan untuk generasi selanjutnya. Setelah itu kita memilah bagian yang valid, kemudian *offspring* tersebut dapat menjadi bagian dari generasi berikutnya.

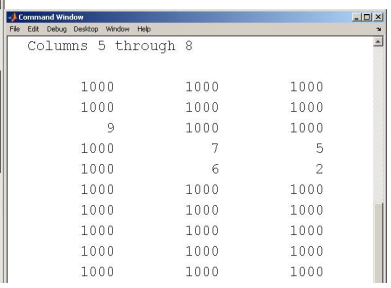
*Tabel 8. Populasi Offspring Hasil Mutasi*

| *Offspring* | Representasi Kromosom | Kromosom Hasil Mutasi | Representasi Jalur |
|---|---|---|---|
| 1 | 1-4-6-9-7-10-3-8-5-2 | 1-4-6-9-7-10-3-8-5-2 | Tidak Valid |
| *2* | 1-2-4-3-8-10-9-7-5-6 | 1-6-2-4-3-8-10-9-7-5 | Tidak Valid |
| *3* | 1-3-4-6-8-10-9-5-2-7 | 1-3-4-6-8-10-9-5-2-7 | Tidak Valid |
| *4* | 1-8-4-7-9-10-3-5-2-6 | 1-4-7-9-10-8-3-5-2-6 | Tidak Valid |
| *5* | 1-9-10-7-5-6-3-8-2-4 | 1-9-10-7-5-6-3-8-2-4 | Tidak Valid |
| *6* | 1-5-6-7-9-10-8-3-2-4 | 1-5-6-7-9-10-8-3-2-4 | Tidak Valid |
| *7* | 1-2-4-7-9-8-10-3-6-5 | 1-2-4-7-9-8-10-3-6-5 | Valid |
| *8* | 1-3-5-7-9-10-4-6-2-8 | 1-3-5-7-9-10-4-6-2-8 | Valid |
| *9* | 1-2-4-7-9-10-5-3-8-6 | 1-2-4-7-9-10-5-3-8-6 | Valid |
| *10* | 1-2-4-7-9-10-5-3-8-6 | 1-2-4-7-910-5-3-8-6 | Valid |

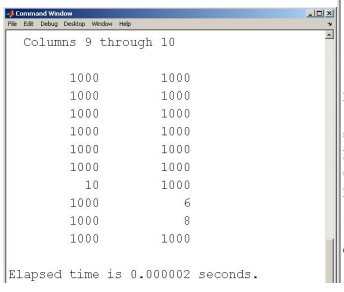
Adapun nilai yang digunakan untuk memasukkanparameter algoritma genetika adalah ukuran populasi 20, probabilitas *crossover* 0.7, probabilitas mutasi 0.25, batas generasi 30, sedangkan node sumber 1, dan node tujuan 10 (Gambar 4). Program kemudian mengeluarkan solusi akhir yang didapat dan grafik yang menampilkan perubahan nilai *fitness* terbaik terhadap pertambahan generasi (Gambar 6).



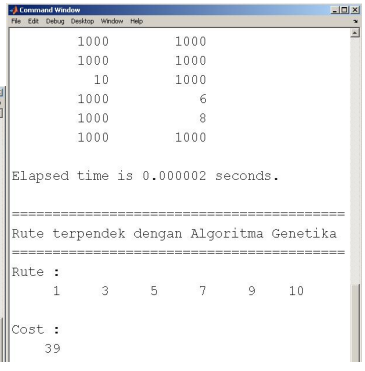

(a)



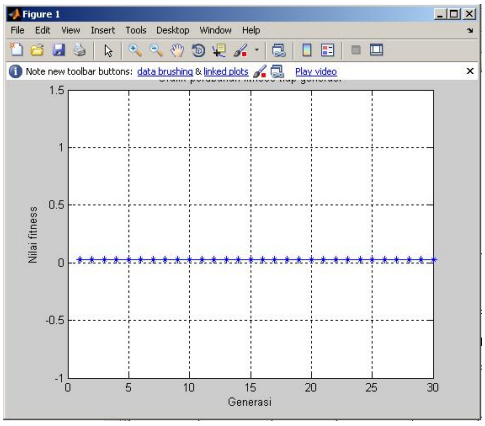

(b)



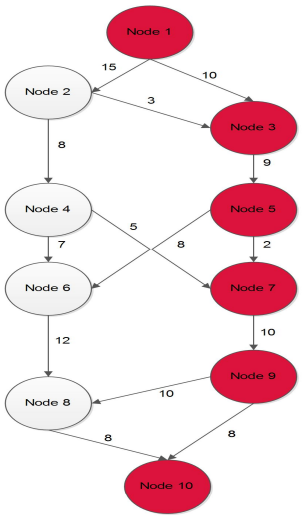

(c)




(d)




(e)




(f)

*Gambar 6. Solusi berupa perubahan nilai fitness*

Dari gambar 5 terlihat bahwa solusi untuk contoh 1 adalah jalur 1-3-5-7-9-10 Yang Memiliki Total *Cost39.* Grafik yang ditujukkan pada gambar 6.e dan 6.f menunjukkan nilai *fitness* terbaik pada tiap generasi. Dari grafik tersebut terlihat bahwa solusi optimal telah diperoleh sejak generasi pertama yaitu dengan nilai *fitness* 0.0256. Selanjutnya akan dibahas beberapa percobaan yang dilakukan untuk melihat pengaruh parameter-parameter genetik seperti ukuran populasi, probabilitas *crossover,* dan probabilitas mutasi terhadap kinerja algoritma genetika.

### 3.1 Hasil Percobaan

Percobaan dilakukan dengan mengubah nilai parameter probabilitas *crossover,* probabilitas mutasi, serta ukuran populasi dalam batas generasi yaitu, 50 untuk semua percobaan. Program dijalankan pada *personal Computer* (PC) dengan prosesor AMD A6-4400M, memori RAM 4GB, dan sistem operasi Microsoft Windows 7 Ultimate.

*3.1.1 Percobaan dengan Mengubah Nilai Parameter Ukuran Populasi (nind)*

Percobaan pertama dilakukan dengan menggunakan batas generasi=50, probabilitas *crossover Pc=0.75,* probabilitasi mutasi *Pm*=0.25, dan ukuran populasi *nind* yang berubah-ubah. Nilai *nind* yang digunakan adalah 10, 50, dan 200.

*Tabel 9. Tabel algoritma genetika berdasarkan ukuran populasi yang berbeda*

| **Masalah Pengujian** | **Banyak Percobaan** | ***Node* Sumber** | ***Node* Tujuan** | **Banyak Solusi Optimal untuk Tiap Ukuran Populasi** | | |
|---|---|---|---|---|---|---|
| | | | | **10** | **50** | **200** |
| **1** | 10 | 1 | 200 | 0 | 2 | 6 |

Terlihat pada tabel 9 bahwa algoritma genetika tersebut memberikan hasil yang cukup baik. Penambahan nilai parameter *nind* mengakibatkan solusi optimal yang diperoleh menjadi semakin banyak untuk kedua masalah pengujian. Hal ini dikarenakan ukuran populasi menentukan banyaknya kombinasi jalur yang ada dalam satu populasi, semakin besar ukuran populasi maka semakin banyak pula variasi dari kromosom atau kombinasi jalur, sehingga semakin besar pula kemungkinan ditemukannya kromosom dengan nilai *fitness* terbaik global atau jalur dengan total *cost* minimum yang dapat dipertahankan dan bahkan diperbaiki lagi hingga akhir generasi yang pada akhirnya dapat menjadi solusi optimal global (E. Satriyanto : 2009). Sehingga dapat dikatakan bahwa penambahan nilai parameter *nind* dapat membuat kinerja algoritma genetika lebih

baik dalam hal pencarian solusi optimal. Namun penentuan nilai parameter *nind* ini haruslah proporsional dengan banyaknya simpul pada jaringan node, karena menurut Tanjung (2010) jika nilai parameter *nind* relatif terlalu tinggi terhadap banyaknya simpul maka tentu saja akan memakan banyak waktu komputasi. Jika nilai parameter *nind* relatif terlalu rendah terhadap banyaknya simpul maka yang terjadi adalah kemungkinan didapat solusi optimal akan semakin rendah, seperti pada tabel 9 ditas. Selanjutnya akan dilihat pengaruh perubahan nilai parameter *Pc* terhadap kinerja algoritma genetika.

*3.1.2 Percobaan dengan Mengubah Nilai Parameter Probabilitas Crossover (Pc)*

Percobaan kedua dilakukan dengan menggunakan batas generasi=50, probabilitas mutasi *Pm=0.25,* ukuran populasi *nind=100*, dan probabilitas *crossover* Pc yang berubah-ubah. Nilai probabilitas *crossover* Pc yang digunakan adlah 0.3, 0.75, dan 1.0.

*Tabel 10. Tabel Algoritma Genetika Berdasarkan Probabilitas Crossover yang Berbeda*

| **Masalah Pengujian** | **Banyak Percobaan** | ***Node* Sumber** | ***Node* Tujuan** | **Banyak Solusi Optimal untuk Tiap Probabilitas *Crossover*** | | |
|---|---|---|---|---|---|---|
| | | | | **0.3** | **0.75** | **1.0** |
| **1** | 10 | 1 | 200 | 0 | 3 | 7 |

Terlihat pada tabel 4.6 bahwa semakin besar nilai probabilitas *crossover* maka semakin banyak pula solusi optimal yang tercapai dalam 10 kali percobaan. Sehingga dapat dikatakan parameter Pc dapat mempengaruhi kinerja algoritma genetika, hal ini dikarenakan semakin besar nilai probabilitas *crossover* maka semakin besar pula peluang melahirkan anak atau *offspring* dari kromosom orang tua yang unggul, sehingga N. Muniati (2009) pada akhir generasi akan terlahir *offspring* dengan inlai *fitness* terbaik. Selanjutnya akan diuji pengaruh perubahan nilai probabilitas mutasi Pm terhadap kinerja algoritma genetika.

*3.1.3 Percobaan dengan Mengubah Nilai Parameter Probabilitas Mutasi (Pm)*

Percobaan ketiga dilakukan dengan menggunakan batas generasi=50, probabilitas *crossover* Pc=0.75, ukuran populasi *nind*=100, dan probabilitas mutasi Pm yang berubah-ubah. Nilai probabilitas mutasi Pm yang digunakan adalah 0.01, 0.25, dan 0.5.

*Tabel 11. Tabel Algoritma Genetika Berdasarkan Probabilitas Mutasi yang Berbeda*

| **Masalah Pengujian** | **Banyak Percobaan** | ***Node* Sumber** | ***Node* Tujuan** | **Banyak Solusi Optimal untuk Tiap Probabilitas Mutasi** | | |
|---|---|---|---|---|---|---|
| | | | | **0.01** | **0.25** | **0.5** |
| **1** | 10 | 1 | 200 | 0 | 2 | 3 |

Terlihat pada tabel 11 bahwa semakin besar nilai probabilitas mutasi maka cenderung semakin banyak solusi optimal yang tercapai dalam 10 kali percobaan untuk kedua maslah pengujian. sehingga dapat dikatakan parameter Pm dapat mempengaruhi kinerja algoritma genetika bahwa proses mutasi dapat mencegah terjadinya optimum lokal. Karena itu meskipun pada kehidupan nyata mutasi memiliki dampak negatif, namun pada algoritma genetika ini mutasi harus tetap ada.

## 4. Kesimpulan

Pada contoh persoalan masalah jalur terpendek dari suatu rute perjalanan yang terdapat 10 simpul setelah melalui proses generasi, evaluasi nilai *fitness, crossover* diperoleh solusi 1-3-5-7-9-10 yang memiliki total *cost39.* Solusi optimal diperoleh dengan nilai *fitness* 0,02356. Setelah dilakukan uji coba didapatkan bahwa algoritma genetika tersebut memberikan hasil yang cukup baik.

## 5. Referensi


Pandjaitan L.W., (2007),'Dasar-Dasar Komputasi Cerdas',Yogyakarta, Andi.

Kusumadewi, (2003),'Arrificial Intelligence',Yogyakarta, Graha Ilmu

Kusumadewi S & Purnomo H., (2005),'Penyelesaian Masalah Optimasi dengan Teknik-Teknik Heuristik',Yogyakarta. Graha Ilmu.

G. Nagib dan W. G. Ali, (2010),'Network Routing Protocol using Genetic Algorithms', International Journal of Electrical & Computer Sciences IJECS-IJENS, vol. 10.

Sivanandam, S.N., Deepa, S.N, (2008) ,'Introduction to Genetic Algorithms', New York, Springer Berlin Heidelberg, 30-60.

E.Satriyanto, (2009), 'Algoritma Genetika', http://lecturer.eepis-its.edu/~kangedi/materi%20kuliah/Kecerdasan%20Buatan/Bab%207%20Algoritma%20Genetika.pdf,diakses tanggal 7 November 2014, pukul 10.28.

W.S.E. Tanjung, (2010), 'Kajian Algoritma Genetika pada Travelling Salesman Problem'. Skripsi: Universitas Sumatera Utara.

N. Muniati, (2009), 'Penerapan Algoritma Genetik pada DNA Sequencing by Hibridization'. Skripsi: Depok, Departemen Matematika, FMIPA, Universitas Indonesia.

S. Lukas, T. Anwar, dan W. Yuliani (2005), 'Penerapan Algoritma Genetika untuk Traveling Salesman Problem dengan Menggunakan Metode Order Crossover dan Insertion Mutation'. Seminar Nasional Aplikasi Teknologi Informasi 2005:1-2