PENERAPAN ALGORITMA GENETIKA PADA PERSOALAN PEDAGANG KELILING (TSP)

Aulia Fitrah[1], Achmad Zaky[2], Fitrasani[3]

Program Studi Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung

E-mail : if14058@students.if.itb.ac.id[1], if14076@students.if.itb.ac.id[2],
if14108@students.if.itb.ac.id[3]

Abstrak

Persoalan pedagang keliling (*Travelling Salesperson Problem*-TSP) merupakan persoalan optimasi untuk mencari perjalanan terpendek bagi pedagang keliling yang ingin berkunjung ke beberapa kota, dan kembali ke kota asal keberangkatan. TSP merupakan persoalan yang sulit bila dipandang dari sudut komputasinya. Beberapa metode telah digunakan untuk memecahkan persoalan tersebut namun hingga saat ini belum ditemukan algoritma yang mangkus untuk menyelesaikannya. Cara termudah untuk menyelesaikan TSP yaitu dengan mencoba semua kemungkinan rute dan mencari rute yang terpendek. Namun, pada zaman yang serba praktis sekarang ini dibutuhkan algoritma yang dapat menyelesaikan TSP dengan cepat sehingga diperoleh solusi yang mendekati solusi optimal. Oleh karena itu digunakan algoritma genetika untuk menentukan perjalanan terpendek yang melalui kota lainnya hanya sekali dan kembali ke kota asal keberangkatan. Algoritma genetika yaitu algoritma pencarian dan optimasi yang terinspirasi oleh prinsip dari genetika dan seleksi alam (teori evolusi Darwin). Algoritma ini sangat tepat digunakan untuk penyelesaian masalah optimasi yang kompleks dan sukar diselesaikan dengan metode konvensional.

***Kata kunci**: Algoritma Genetika, Travelling Salesperson Problem*

1. Pendahuluan

Persoalan pedagang keliling (*Travelling Salesperson Problem*-TSP) merupakan salah satu persoalan optimasi kombinatorial; jika diberikan sejumlah kota (atau tempat) dan biaya perjalanan dari satu kota ke kota lain. Deskripsi persoalannya adalah bagaimana menemukan rute perjalanan paling murah dari suatu kota dan mengunjungi semua kota lainnya, masing-masing kota hanya dikunjungi satu kali, dan harus kembali ke kota asal keberangkatan. Kombinasi dari semua rute perjalanan yang ada adalah faktorial dari jumlah kota. Biaya perjalanan bisa berupa jarak, waktu, bahan bakar, kenyamanan, dan sebagainya.

Tipe persoalan TSP banyak muncul di beberapa persoalan aplikasi teknik yang mencakupi optimasi tampilan dari pipa saluran udara, perancangan *antena feed system*, pengurutan objek untuk memperoleh susunan yang tepat.

[3]Sudah banyak metode pemecahan masalah yang dikerahkan untuk menyelesaikan persoalan TSP. Persoalan TSP adalah persoalan yang sulit (*hard problem*) dipandang dari sudut komputasinya. Hingga saat ini belum ditemukan algoritma yang mangkus (dalam orde polinomial) untuk menyelesaikannya. Jika algoritma dengan kompleksitas dalam orde polinomial ditemukan untuk TSP, maka banyak persoalan sulit juga dapat diselesaikan dengan algoritma tersebut.

2. Algoritma Genetika

Algoritma genetika merupakan teknik pencarian dan optimasi yang terinspirasi oleh prinsip dari genetika dan seleksi alam (teori evolusi Darwin). Algoritma ini digunakan untuk mendapatkan solusi yang tepat untuk masalah optimasi dari satu variabel atau multi variabel.

Berbeda dengan teknik pencarian konvensional, algoritma genetika bermula dari himpunan solusi yang dihasilkan secara acak. Himpunan ini disebut populasi. Sedangkan setiap individu dalam populasi disebut kromosom yang merupakan representasi dari solusi. Kromosom-kromosom berevolusi dalam suatu proses iterasi yang berkelanjutan yang disebut generasi. Pada setiap generasi, kromosom dievaluasi berdasarkan suatu fungsi evaluasi (Gen dan Cheng, 1997). Setelah beberapa generasi maka algoritma genetika akan konvergen pada kromosom terbaik, yang diharapkan merupakan solusi optimal (Goldberg,1989). Sebelum

Pertama kali, sebelum algoritma genetika dijalankan, maka perlu didefinisikan fungsi *fitness* sebagai masalah yang ingin dioptimalkan. Jika nilai *fitness* semakin besar, maka sistem yang dihasilkan semakin baik. fungsi *fitness* ditentukan dengan metode heuristik.

Algoritma genetika sangat tepat digunakan untuk penyelesaian masalah optimasi yang kompleks dan

sukar diselesaikan dengan menggunakan metode konvensional. Sebagaimana halnya proses evolusi di alam, suatu algoritma genetika yang sederhana umumnya terdiri dari tiga operasi yaitu: operasi reproduksi, operasi *crossover* (persilangan), dan operasi mutasi. Struktur umum dari suatu algoritma genetika dapat didefinisikan dengan langkah-langkah sebagai berikut:

- Membangkitkan populasi awal secara random.
- Membentuk generasi baru dengan menggunakan tiga operasi diatas secara berulang-ulang sehingga diperoleh kromosom yang cukup untuk membentuk generasi baru sebagai representasi dari solusi baru.
- Evolusi solusi yang akan mengevaluasi setiap populasi dengan menghitung nilai *fitness* setiap kromosom hingga kriteria berhenti terpenuhi. Bila kriteria berhenti belum terpenuhi maka akan dibentuk lagi generasi baru dengan mengulangi langkah 2. beberapa kriteria berhenti yang sering digunakan antara lain:
  - berhenti pada generasi tertentu
  - berhenti setelah dalam beberapa generasi berturut-turut didapatkan nilai *fitness* tertinggi/terendah (tergantung persoalan) tidak berubah.
  - berhenti bila dalam n generasi berikutnya tidak diperoleh nilai *fitness* yang lebih tinggi/rendah.

3. Persoalan Pedagang Keliling dengan Teknik Pencarian Konvensional

3.1 Algoritma Brute Force

Salah satu cara termudah untuk menyelesaikan TSP yaitu dengan menggunakan algoritma brute force. Hal yang dilakukan yaitu mencoba semua kombinasi dan mencari rute yang paling murah. Tetapi hal tersebut memerlukan waktu yang sangat lama karena banyaknya jumlah kombinasi yang ada. Sebagai contoh, jumlah kombinasi rute untuk 20 kota adalah $20! = 2{,}4 \times 10^{18}$. Jumlah yang sangat besar untuk suatu algoritma pencarian.

3.2 Algoritma Greedy

Metode konvensional lain dalam menyelesaikan TSP yaitu dengan menggunakan algoritma greedy. Hal yang dilakukan yaitu memilih kota yang belum dikunjungi yang mempunyai biaya paling rendah pada setiap langkah. Namun, dengan menggunakan algoritma greedy solusi yang dihasilkan tidak menjamin bahwa solusi tersebut optimal.

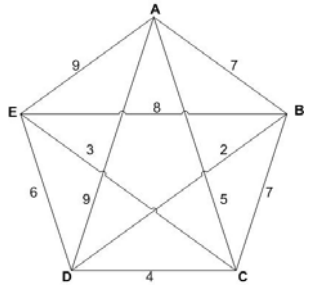
4. Persoalan Pedagang Keliling dengan Algoritma Genetika

Dalam dunia yang serba praktis, diperlukan suatu algoritma cepat untuk mencari suatu solusi yang mendekati solusi optimal, tetapi tidak memerlukan waktu yang lama. Algoritma genetika adalah salah satu algoritma alternatif yang dapat digunakan sebab prosesnya cepat dan memberikan hasil yang diinginkan. Selain itu, algoritma genetika juga mampu memberikan suatu solusi pada waktu kapanpun.

Bagaimana algoritma genetika dapat menyelesaikan TSP yaitu solusi direpresentasikan ke dalam suatu kromosom yang berisi dari nomor urut kota-kota selain kota asal. Masing-masing nomor urut tidak boleh muncul lebih dari satu kali di dalam kromosom sehingga satu kromosom merepresentasikan satu rute perjalanan (satu solusi) yang valid.

4.1 Contoh persoalan

Berikut contoh persoalan TSP yang diselesaikan dengan menggunakan algoritma genetika.
Terdapat 5 buah kota yang akan dilalui oleh seorang pedangang keliling, misalnya Kota A,B,C,D,E. Perjalanan dimulai dari kota A dan berakhir di kota A. Jarak antar kota diperlihatkan pada graf di bawah ini:



Persoalan TSP tersebut akan diselesaikan dengan menggunakan algoritma genetika. Kriteria berhenti ditentukan terlebih dahulu yaitu apabila setelah dalam beberapa generasi berturut-turut diperoleh nilai *fitness* yang terendah tidah berubah. Pemilihan nilai *fitness* yang terendah sebagai syarat karena nilai tersebut yang merepresentasikan jarak terdekat yang dicari pada persoalan TSP ini.
Ada 4 kota yang akan menjadi gen dalam kromosom yaitu kota-kota selain kota asal.

a. Inisialisasi
Misalkan kita menggunakan 6 buah populasi dalam satu generasi, yaitu
Kromosom[1] = [B D E C]
Kromosom[2] = [D B E C]
Kromosom[3] = [C B D E]
Kromosom[4] = [E B C D]
Kromosom[5] = [E C B D]
Kromosom[6] = [C D E B]

b. Evaluasi kromosom
Kita akan menghitung nilai *fitness* dari tiap kromosom yang telah dibangkitkan:

Fitness[1] = AB+BD+DE+EC+CA
= 7 + 2 + 6 + 3 + 5 = 23
Fitness[2] = AD+DB+BE+EC+CA
= 9 + 2 + 8 + 3 + 5 = 27
Fitness[3] = AC+CB+BD+DE+EA
= 5 + 7 + 2 + 6 + 9 = 29
Fitness[4] = AE+EB+BC+CD+DA
= 9 + 8 + 7 + 4 + 9 = 37
Fitness[5] = AE+EC+CB+BD+DA
= 9 + 3 + 7 + 2 + 9 = 30
Fitness[6] = AC+CD+DE+EB+BA
= 5 + 4 + 6 + 8 + 7 = 30

c. Seleksi kromosom
Oleh karena pada persoalan TSP yang diinginkan yaitu kromosom dengan *fitness* yang lebih kecil akan mempunyai probabilitas untuk terpilih kembali lebih besar maka digunakan inverse.

$$\mathbf{Q[i]} = {}^{1}\!/\!_{Fitness[i]} \quad (1)$$

Q[1] = $1/23$ = 0,043
Q[2] = $1/27$ = 0,037
Q[3] = $1/29$ = 0,034
Q[4] = $1/37$ = 0,027
Q[5] = $1/30$ = 0,033
Q[6] = $1/30$ = 0,033
Total = 0,043+0,037+0,034+0,027+0,033+0,033
= 0,207
Untuk mencari probabilitas kita menggunakan rumus berikut :

$$\mathbf{P[i]} = {}^{Q[i]}\!/\!_{Total} \quad (2)$$

P[1] = $0,043/0,207$ = 0,208
P[2] = $0,037/0,207$ = 0,179
P[3] = $0,034/0,207$ = 0,164
P[4] = $0,027/0,207$ = 0,130
P[5] = $0,033/0,207$ = 0,159
P[6] = $0,033/0,207$ = 0,159

Dari probabilitas di atas dapat terlihat bahwa kromosom ke-1 mempunyai *fitness* paling kecil mempunyai probabilitas untuk terpilih pada generasi selanjutnya lebih besar dari kromosom lainnya.

Untuk proses seleksi kita menggunakan *roulete-wheel*, untuk itu kita terlebih dahulu mencari nilai kumulatif dari probabilitasnya.
C[1] = 0,028
C[2] = 0,028+0,179 = 0,387
C[3] = 0,387+0,164 = 0,551
C[4] = 0,551+0,130 = 0,681
C[5] = 0,681+0,159 = 0,840
C[6] = 0,840+0,159 = 1

Proses *roulete-wheel* adalah membangkitkan nilai acak R antara 0-1. Jika **R[k]<C[k]** maka kromosom ke-k sebagai induk, selain itu pilih kromosom ke-k sebagai induk dengan syarat **C[k-1] < R[k] < C[k]**. Kita putar roulete-wheel sebanyak jumlah kromosom yaitu 6 kali (membangkitkan bilangan acak R).
R[1] = 0,314
R[2] = 0,111
R[3] = 0,342
R[4] = 0,743
R[5] = 0,521
R[6] = 0,411

Setelah itu, populasi baru akan terbentuk yaitu :
Kromosom[1] = [2] = [D B E C]
Kromosom[2] = [1] = [B D E C]
Kromosom[3] = [3] = [C B D E]
Kromosom[4] = [5] = [E C B D]
Kromosom[5] = [4] = [E B C D]
Kromosom[6] = [6] = [C D E B]

d. *Crossover* (pindah silang)
Pindah silang pada TSP dapat diimplementasikan dengan skema *order crossover*. Pada skema ini, satu bagian kromosom dipertukarkan dengan tetap menjaga urutan kota yang bukan bagian dari kromosom tersebut. Kromosom yang dijadikan induk dipilih secara acak dan jumlah kromosom yang dicrossover dipengaruhi oleh parameter *crossover probability* (ρc).

Misal kita tentukan ρc = 25%, maka diharapkan dalam 1 generasi ada 50% (3 kromosom) dari populasi mengalami *crossover*.

Pertama kita bangkitkan bilangan acak R sebanyak jumlah populasi yaitu 6 kali.
R[1] = 0,451
R[2] = 0,211
R[3] = 0,302
R[4] = 0,877
R[5] = 0,771
R[6] = 0,131

Kromosom ke-k yang dipilih sebagai induk jika R[k] < ρc. Maka yang akan dijadikan induk adalah kromosom[2], kromosom[3], dan kromosom[6].

Setelah melakukan pemilihan induk, proses selanjutnya adalah menentukan posisi *crossover*. Hal tersebut dilakukan dengan membangkitkan bilangan acak antara 1 sampai dengan panjang kromosom-1. Dalam kasus TSP ini bilangan acaknya adalah antara 1-3. Misal diperoleh bilangan acaknya 1, maka gen yang ke-1 pada kromosom induk pertama diambil kemudian ditukar dengan gen pada kromosom induk kedua yang belum ada pada induk pertama dengan tetap memperhatikan urutannya.

Bilangan acak untuk 3 kromosom induk yang akan di-*crossover* :
C[2] = 2
C[3] = 1
C[6] = 2

Proses *crossover* :
Kromosom[2] = **Kromosom[2]** >< **Kromosom[3]**
= [B D E C] >< [C B D E]
= [B D C E]
Kromosom[3] = **Kromosom[3]** >< **Kromosom[6]**
= [C B D E] >< [C D E B]
= [C D E B]
Kromosom[6] = **Kromosom[6]** >< **Kromosom[2]**
= [C D E B] >< [B D E C]
= [C D B E]

Populasi setelah di-*crossover* :
Kromosom[1] = [D B E C]
Kromosom[2] = [B D C E]
Kromosom[3] = [C D E B]
Kromosom[4] = [E C B D]
Kromosom[5] = [E B C D]
Kromosom[6] = [C D B E]

e. Mutasi
Pada kasus TSP ini skema mutasi yang digunakan adalah *swapping mutation*. Jumlah kromosom yang mengalami mutasi dalam satu populasi ditentukan oleh parameter *mutation rate(*ρm*)*. Proses mutasi dilakukan dengan cara menukar gen yang dipilih secara acak dengan gen sesudahnya. Jika gen tersebut berada di akhir kromosom, maka ditukar dengan gen yang pertama.

Pertama kita hitung dulu panjang total gen yang ada pada satu populasi:
Panjang total gen=jumlah gen dalam 1 kromosom * jumlah Kromosom (3)
= 4 * 6
= 24

Untuk memilih posisi gen yang mengalami mutasi dilakukan dengan membangkitkan bilangan acak antara 1 – Panjang total gen yaitu 1- 24. Misal kita tentukan ρm = 20 %. Maka jumlah gen yang akan dimutasi adalah = 0,2*24
= 4,8
= 5

5 buah posisi gen yang akan dimutasi, setelah diacak adalah posisi 3, 7, 10, 20, 24.

Proses mutasi :
Kromosom[1] = [D B **C** E]
Kromosom[2] = [B D **E** C]
Kromosom[3] = [C **E** D B]
Kromosom[4] = [E C B D]
Kromosom[5] = [D B C **E**]
Kromosom[6] = [E D B **C**]

Proses algoritma genetik untuk 1 generasi telah selesai. Maka nilai *fitness* setelah 1 generasi adalah:
Fitness[1] = AD+DB+BC+CE+EA
= 9 + 2 + 7 + 3 + 9 = 30
Fitness[2] = AB+BD+DE+EC+CA
= 7 + 2 + 6 + 3 + 5 = 23
Fitness[3] = AC+CE+ED+DB+BA
= 5 + 3 + 6 + 2 + 7 = 23
Fitness[4] = AE+EC+CB+BD+DA
= 9 + 3 + 7 + 2 + 9 = 30
Fitness[5] = AD+DB+BC+CE+EA
= 9 + 2 + 7 + 3 + 9 = 30
Fitness[6] = AE+ED+DB+BC+CA
= 9 + 6 + 2 + 7 + 5 = 29

Sebelumnya telah ditentukan kriteria berhenti yaitu bila setelah dalam beberapa generasi berturut-turut diperoleh nilai *fitness* yang terendah tidah berubah. Pada 1 generasi telah terlihat bahwa terdapat nilai *fitness* terkecil yang tidak berubah. Apabila perhitungan dilanjutkan hingga ke generasi ke-N maka diyakinkan bahwa nilai *fitness* yang terendah tetap tidak akan berubah. Walaupun perhitungan cukup dijabarkan hingga generasi ke-1 saja namun solusi yang mendekati optimal telah didapatkan. Oleh karena itu, terbukti bahwa algoritma genetika dapat menyelesaikan persoalan TSP.

4.2 Pseudo-code Algoritma Genetika

Berikut pseudo-code yang dibuat untuk menyelesaikan persoalan TSP di atas dengan menggunakan algoritma genetika.

```
function Fitness (Kromosom[i])→integer
{menghitung nilai fitness dari masing-masing kromosom}
Deklarasi
        Jum : integer
        j : integer
        Kromosom[][] : array of integer of integer
function Jarak(input A, B : integer) → integer
{menghasilkan jarak antara dua kota A dan B}
Algoritma
```

```
        Jum ← Jarak(A,Kromosom[i][1])
        for j←2 to 4 do
                Jum ← Jum +
Jarak(Kromosom[i][j-1],Kromosom[i][j])
        endfor
        Jum ← sum + Jarak(Kromosom[i][4],A)
        → Jum
```

```
procedure Crossover (input populasi: integer, ρc:
real)
{melakukan pemilihan induk pada proses cross
over}
Deklarasi
        k : integer
        R[] : array of integer
        function random (input a-b :
integer)→integer
{menghasilkan bilangan random bilangan a hingga
b}

Algoritma
        k= 0
        While k <= populasi do
                R[k] ← random(0-1)
                if R[k] < ρc then
                        pilih Kromosom[k][]
sebagai induk
                endif
                k ← k+1
        endwhile
```

```
function JumlahMutasi(input JumGen,
JumlahKromosom: integer, ρm: real) → integer
{menghitung jumlah proses mutasi}
Deklarasi
        TotalGen : integer
        JumMutasi : integer
Algoritma
        TotalGen ← JumGen * JumlahKromosom
        ρm ← 0.2
        JumMutasi ← 0.2*TotalGen
        → JumMutasi
```

5. Kesimpulan

Persoalan pedagang keliling (TSP) dapat diselesaikan dengan menggunakan algoritma genetika. Walaupun solusi TSP yang dihasilkan oleh algoritma ini belum tentu merupakan solusi paling optimal (misalnya apabila yang dilalui sangat banyak), namun algoritma genetika akan menghasilkan solusi yang lebih optimal pada setiap generasinya. Hal tersebut terlihat dari nilai *fitness* tiap generasi. Kelebihan algoritma genetika dibandingkan metode pencarian konvensiona pada TSP yaitu pertama, solusi dapat diperoleh kapanpun karena solusi dihasilkan pada generasi ke berapapun, kedua, algoritma genetika tidak harus membutuhkan waktu yang lama karena tidak semua kemungkinan dicoba, tergantung pada kriteria berakhirnya.

Daftar Pustaka

[1] Haupt, Randy L and Sue Ellen Haupt. *Practical Genetic Algorithms,* 2nd edition, John Wiley & Sons, Inc, Hoboken, New jersey, 2004.

[2] Hermawanto, Denny. *Tutorial Algoritma Genetika*.
http://cspi.istecs.org/Papers1/csp546.pdf.gz
Diakses pada tanggal 12 Mei 2006 pukul 15:35 WIB

[3] Munir, Rinaldi. *Strategi Algoritmik*, Program Studi Informatika STEI Institut Teknologi Bandung, Bandung, 2006.

[4] Sukmawan, Budi. *Sekilas Tentang Algoritma Genetika dan Aplikasinya pada Optimasi Jaringan Pipa Air Bersih*,
http://bdg.centrin.net.id/~budskman/ga.htm
Diakses pada tanggal 12 Mei 2006 pukul 15:34 WIB

[5] Suyanto. *Algoritma Genetika dalam MATLAB*, ANDI, Yogyakarta, 2005.

[6]
http://en.wikipedia.org/wiki/Selection_%28genetic_algorithm%29
Diakses pada tanggal 15 Mei 2006 pukul 14:43 WIB