**LECTURE NOTES**

**Selected Topics in Computational Intelligence I**

**Session 2**

**Genetic Algorithm**

**LEARNING OUTCOMES**

**Tujuan Instruksional Umum :**

1. Mahasiswa mampu menjelaskan fungsi dari Genetic Algorithm
2. Mahasiswa dapat membuat aplikasi kecerdasan buatan dengan Genetic Algorithm

**Tujuan Instruksional Khusus :**

1. Mahasiswa dapat menyebutkan sejarah dan definisi dari Genetic Algorithm
2. Mahasiswa mampu menjelaskan metode dan penerapan Genetic Algorithm
3. Mahasiswa mampu membuat aplikasi berbasis Genetic Algorithm

**2.1 Introduction Genetic Algorithm (GA)**

Optimasi adalah proses menyelesaikan suatu masalah tertentu supaya berada pada kondisi yang paling menguntungkan dari suatu sudut pandang. Optimasi adalah sautu proses yang berhubungan dengan penyesuaian masukan, pemilihan karakteristik peralatan, proses matematis dan pengujian yang dilakukan untuk menemukan keluaran optimum (Haupt dan Haupt, 2004). Klasifikasi optimasi terdiri dari: fungsi, pencarian harga optimum, dengan kendala, tanpa kendala, noncontinue, statis, dinamis, variabel tunggal, multivariable, trial dan error.

Metode heuristic merupakan sautu metode untuk menemukan penyelesaian masalah optimasi sebatas dalam kadar cukup baik dan masuk akal untuk diterima. Walaupun penyelesaian yang ditemukan bukanlah penyelesaian terbaik, tetapai sudah dapat untuk diterim akarena sudah mencapai kadar 90pesen daripada penyelesaian optimum (Turban dan Aronson, 1998). Beberapa metode heuristic yang biasanya digunakan adalah Metode Hill climbing, tabu search, simulated annealing, ant colony, GA dan Artificial Immune System.

Setiap organisme memiliki seperangkat aturan, peraturan-peraturan ini dikodekan dalam gen dari organisme, yang pada gilirannya dihubungkan bersama-sama untaian string yang disebut kromosom. Gen masing-masing mewakili sifat tertentu dari organisme, seperti warna mata atau warna rambut, dan memiliki pengaturan yang berbeda. Misalnya, pengaturan untuk gen warna rambut mungkin pirang, hitam atau kemerahan. Gen dan pengaturan mereka biasanya disebut sebagai *genotipe* organisme. Ekspresi fisik *genotipe* organisme itu sendiri yang disebut *fenotipe*. Ketika dua organisme kawin mereka berbagi gen mereka. Keturunan yang dihasilkan mungkin berakhir memiliki setengah gen dari satu orang tua dan setengah dari yang lain. Proses ini disebut rekombinasi. Kadang-kadang gen dapat bermutasi. Kehidupan di bumi telah berevolusi untuk menjadi seperti itu adalah melalui proses alami, rekombinasi seleksi dan mutasi.

Istilah *Genetic Algorithm* ini diusulkan pertama kali oleh John Holland pada tahun 1975 dalam bukunya berjudul “*Adaptation in Natural and Artificial Systems”*. Salah satu penerapannya ialah *pathplanning* pada *mobile robot* sehingga robot bergerak dengan efisien ke tempat tujuan. Oleh karena itu teknik pemrograman evolusi, berdasarkan *genetic algorithm*, berlaku untuk banyak masalah optimasi, seperti optimasi fungsi dengan kendala linear dan nonlinear, masalah *travelling salesman*, dan masalah penjadwalan, partisi, dan kontrol.

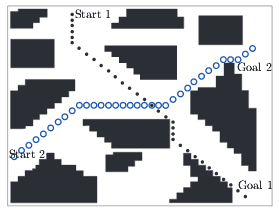
*Genetic Algorithm* merupakan metode adaptif, dimulai dengan satu set solusi (diwakili oleh kromosom) disebut populasi. Solusi dari satu populasi yang diambil dan digunakan untuk membentuk populasi baru. Hal ini didorong oleh harapan, bahwa populasi baru akan lebih baik dari yang lama. Solusi yang dipilih untuk membentuk solusi baru (keturunan) yang dipilih sesuai dengan kebugaran mereka yang lebih cocok mereka adalah lebih banyak kesempatan mereka harus mereproduksi. Secara teori, *Gnome* memiliki beberapa gen, dan kumpulan lebih dari 1 *gnome* menjadi populasi.

*Genetic Algorithm* merupakan suatu metode heuristika yang dikembangkan berdasarkan prinsip genetika dan proses seleksi alamiah teori evolusi Darwin. Algoritma genetika merupakan metode pencarian yang didasarkan pada proses evolusi alamiah (Wolfgang Ertel, 2011), yaitu terbentuknya populasi awal secara acak yang teridir dari individu dengan sifat yang tergantung pada

gen –gen dalam kromosomnya.Proses komputasi yang terjadi dalam algoritma ini analog dengan proses seleksi mahluk hidup dalam sebuah populasi. Sejumlah penyelesaian yang mungkin dikenal sebagai populasi. Populasi dalam GA juga terdiri atas sejumlah individu yang disebut kromosom. Kromosom merupakan representasi penyelesaian masalah yang masih berbentuk symbol.

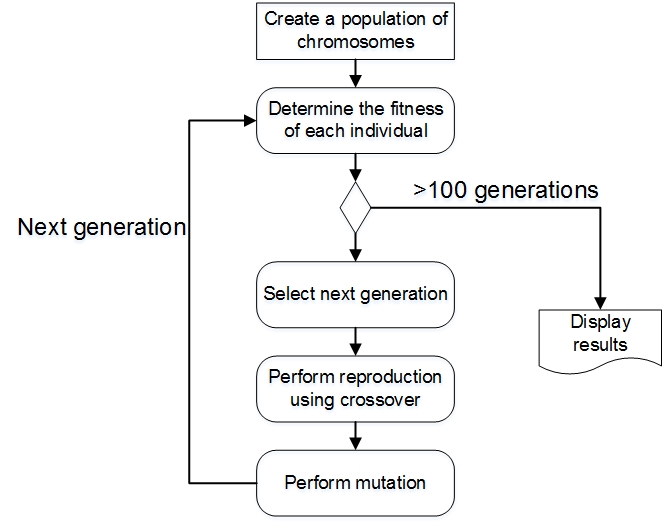
Populasi awal dalam *Genetic Algorithm* dibentuk secara acak, sedangkan populasi berikutnya dibentuk oleh operator-operator Algoritma genetika selama beberapa generasi. Pada setiap generasi, kromosom –kromosm akan mengalami proses evaluasi dengan menggunakan alat ukur yang disebut dengan fungsi fitness(kebugaran). Kromosm-kromosomg yang dibentuk dari kromosom generasi sebelumnya disebut seabga anak(offspring). Demikian juga dengan apsangan kromosmn generasi sebelumnya disebut juga sebagai induk (parents). Penyilangan memungkinkan anak yang mewarisi sifat kedua induknya. Dalam GA proses reproduksi dikenal seabgai oerator penyilangan (crossover). Dalam GA, dikenal operator yang dapat mengubah gen-gen dalam kromosom, operator ini disebut sebagai operator mutasi (mutation). Menurut Haupt (2004), struktur dasar Algoritma Genetika terdiri atas beberapa langkah:

1. Inisialisasi populasi
2. Evaluasi populasi
3. Seleksi populasi yang akan dikenai operator genetika
4. Proses penyilangan pasangan kromosom tertentu
5. Proses mutasi kromosomg tertentu
6. Evaluasi populasi baru
7. Ulangi dari langka 3 jika syarat berhenti belum terpenuhi

****

**Gambar 2.1 Path Planning pada Mobile Robot**

Gambar 2.2 menampilkan blok diagram *Genetic Algorithm*, dimana perlu membuat sebuah populasi dari kromoson untuk selanjutnya menentukan fitness dari tiap individu.



Gambar 6.2 Blok Diagram Genetic Algorithm

**2.2 Implementasi Program Utama**

Berikut ini kode utama program:

int main()

{

int Generasi;

//perintah untuk memberi nilai parameter algoritma

UkuranPopulasi=10;

ProbabilitasPenyilangan=0.5;

ProbabilitasMutasi=0.1;

BanyakGenerasi=100;

//Perintah untuk memberi nilai masukan masalah

BanyakGen=6;

Srand(time (NULL));

Initialization(BanyakGen);

MaxObjective=0;

//iterasi untuk melakukan komputasi

//sebanyak parameter banyak generasi

for(Generasi=1;Generasi<=BanyakGenerasi’Generasi++)

{

//perhitungan nilai fungsi evaluasi

Evaluasi (Generasi);

//simpan historis iterasi seperti solusi terbaik,

//kromosom terbaik dan sebagainya

Historis (Generasi);

//Selection process

Selection(Generasi);

//Crossover process

Crossover();

//Mutation process

Mutation();

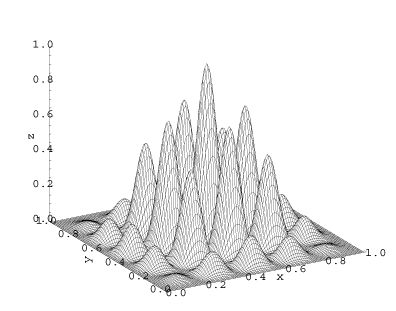
}

Return 0;

}

**2.3 Aplikasi Genetic Algorithm**

Berikut ini contoh program algoritma genetik dengan C# dengan kelas GA dan Genomedan kelas helper bernama GenomeComparer untuk yang menggunakan fungsi sin dan cos untuk membuat plot di bawah dengan solusi optimal ialah (0.5,0.5):



Gambar 2.4 Plot Percobaan Genetic Algorithm

Misalnya kita ingin membuat parameter crossover rate, mutation rate, population size, jumlah generasi, dan jumlah parameter sebagai berikut :

GA ga = new GA(0.8,0.05,100,2000,2);

ga.FitnessFunction = new GAFunction(theActualFunction);

*Pembuatan objek diatas memungkinkan kita mendeklarasikan FitnessFunction*

publicstaticdouble theActualFunction(double[] values)

{

if (values.GetLength(0) != 2)

thrownew ArgumentOutOfRangeException("hanya memiliki 2 argumen");

double x = values[0];

double y = values[1];

double n = 9;

double f1 = Math.Pow(15\*x\*y\*(1-x)\*(1-y)\*Math.Sin(n\*Math.PI\*x)

\*Math.Sin(n\*Math.PI\*y),2);

return f1;

}

Kemudian untuk class Test-nya sendiri yang digunakan sebagai class untuk menguji isi program adalah sebagai berikut:

using System;

using btl.generic;

public class Test

{

// solusi optimal adalah (0.5,0.5)

public static double theActualFunction(double[] values)

{

if (values.GetLength(0) != 2)

throw new ArgumentOutOfRangeException("gunakan 2 argumen");

double x = values[0];

double y = values[1];

double n = 9;

double f1 = Math.Pow(15\*x\*y\*(1-x)\*(1y)

\*Math.Sin(n\*Math.PI\*x)\*Math.Sin(n\*Math.PI\*y),2);

return f1;

}

public static void Main()

{

// Crossover = 80%

// Mutation = 5%

// Population size = 100

// Generations = 2000

// Genome size = 2

GA ga = new GA(0.8,0.05,100,2000,2);

ga.FitnessFunction = new GAFunction(theActualFunction);

ga.Elitism = true;

ga.Go();

double[] values;

double fitness;

ga.GetBest(out values, out fitness);

System.Console.WriteLine("Best ({0}):", fitness);

for (int i = 0 ; i < values.Length ; i++)

System.Console.WriteLine("{0} ", values[i]);

ga.GetWorst(out values, out fitness);

System.Console.WriteLine("\nWorst ({0}):", fitness);

for (int i = 0 ; i < values.Length ; i++)

System.Console.WriteLine("{0} ", values[i]);

}

}

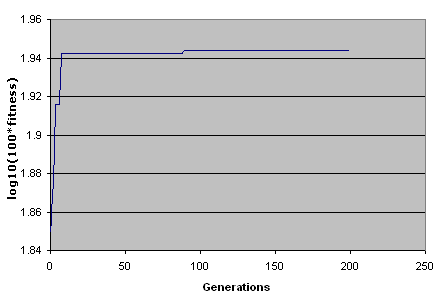
Pada penerapan program di atas, *crossover* membutuhkan 3 *gnome*, dipisahkan secara random dan menghasilkan 2 *gnome* baru dengan ditukar pada ujungnya, seperti pada ilustrasi berikut ini:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10 20 30 40 50 60 70 | |  | | --- | | 80 90 00 | |  | 10 20 30 40 50 60 70 | |  | | --- | | 30 20 10 | |
|  |  | 🡪 |  |  |
| 00 90 80 70 60 50 40 | |  | | --- | | 30 20 10 | |  | 00 90 80 70 60 50 40 | |  | | --- | | 80 90 00 | |

Pada mutasi, terjadi lebih jarang, tiap gen dengan *gnome* diuji untuk melihat apakah diijinkan untuk mutasi. Jika diijinkan, maka diganti dengan bilangan acak :

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 10 | 20 | 30 | 40 | 50 | 60 | |  | | --- | | ***70*** | | 80 | 90 | ===> | 10 | 20 | 30 | 40 | 50 | 60 | |  | | --- | | ***22*** | | 80 | 90 |

Tampilan perkembangan dari *Genetic Algorithm* sebagai berikut:



Gambar 2.5 Progress Perkembangan Genetic Algorithm pada Program

Dengan contoh di atas, maka solusi optimal ialah (0.5, 0.5), dan kita temukan setelah generasi 250 ditemukan solusi yang mendekati.

**SIMPULAN**

*Genetic Algorithm merupakan metode adaptif, dimulai dengan satu set solusi (diwakili oleh kromosom) disebut populasi. Solusi dari satu populasi yang diambil dan digunakan untuk membentuk populasi baru. Hal ini didorong oleh harapan, bahwa populasi baru akan lebih baik dari yang lama.*

**DAFTAR PUSTAKA**

* Adries P. Engelbrect. (2007), ***Computational Intelligence An Introduction***. 2nd ed. John Wiley & Sons. USA.
* Haupt, R.L dan Hupt., 2004, Practical Genetic Algorithms Ner Jersy: John Wiley and Sons, Inc.
* http://visualstudiomagazine.com/articles/2014/02/01/evolutionary-optimization-using-c.aspx