Dasar teori

Algoritma Genetika merupakan metode yang biasa digunakan untuk memecahkan suatu pencarian nilai dalam sebuah masalah optimasi. Algoritma ini didasarkan pada proses genetik yang ada di dalam makhluk hidup; yaitu perkebangan generasi dalam sebuah populasi yang alami, secara lambat laun mengikuti prinspip seleksi alam. Dengan meniru teori evolusi ini, algoritma genetika dapat digunakan untuk mencari solusi permasalahan dalam dunia nyata.

Beberapa hal yang harus dilakukan dalam algoritma genetika adalah:

* Menentukan proses pembangkitan populasi awal, pembangkitan dilakukan secara acak
* Mendefinisikan nilai fitness, yang merupakan ukuran baik/tidaknya sebuah individu atau baik/tidaknya solusi yang didapatkan
* Menentukan proses seleksi yang akan digunakan
* Menentukan proses pindah silang(cross-over) dan mutasi gen yang akan digunakan

Implementasi

* Proses pembangkitan populasi awal

Populasi ini terdiri atas individu – individu yang merupakan sebuah jadwal dengan alokasi mata kuliah yang dilakukan secara random

* Menghitung nilai fitness

Nilai fitness akan dihitung untuk tiap-tiap individu. Nilai fitness disini merupakan jumlah mata kuliah yang tidak bentrok

* Melakukan proses selection

Mata kuliah – mata kuliah yang bentrok akan dipilih pada salah satu individu, yang akan di cross-over nantinya dengan individu pasangannya

* Melakukan proses cross-over

Melakukan pindah silang antar 2 individu terhadap mata kuliah yang sudah dipilih tadi

* Melakukan proses mutation

Jika setelah melakukan proses cross-over dan masih terdapat mata kuliah yang bentrok, maka salah satu mata kuliah yang bentrok tersebut akan diganti hari/ruangannya supaya tidak bentrok

Proses iterasi akan berhenti jika terdapat individu yang memiliki nilai fitness sebesar 100% atau jumlah iterasi maksimum tercapai

Dasar Teori

<<Alson yang buat>>

Beberapa hal yang harus dilakukan dalam algoritma hill climbing adalah:

* Membuat random untuk starting point (starting awal algoritma yang mengassign variable secara *complete* namun belum (tentu) *consistent* dengan constraint)

(diimplementasi pada procedure startRandom)

* Membuat next step yang merupakan neighbour dari kondisi sebelumnya (diimplementasi pada procedure nestStep)
* Menentukkan jumlah langkah (jika dia berhenti di local maksimum, harus dibatasi berapa langkah dia mencari tetangganya)

Implementasi

1. Proses dimulai dengan memanggil fungsi startRandom untuk mengenerate assignment pada variable.

startRandom melakukan assignment semua matkul pada ruangan dan waktu yang random (fungsi ini pun sudah memastikan bahwa ruangan dan waktu terebut diperbolehkan oleh constraint)

1. Kemudian, proses dilanjutkan dengan mencari tetangganya

Pilih satu matkul secara random. Kemudian, cari lokasi ruangan dan waktu yang masih mungkin (ini adalah representasi tetangga dari state sebelumnya) secara random pula

1. Apabila tetangga nya lebih baik (jumlah matkul yang bentrok menurun atau minimal sama, pindah ke tetangga, ulangi increment jumlah langkah dari 0 kembali
2. Apabila tidak, increment jumlah langkah
3. Apabila jumlah langkah belum mencapai batas dan jumlah matkul yang bentrok belum 0, lanjutkan pencarian (pada step 2)
4. Apabila matkul yang bentrok sudah 0, (kasus terbaik) stop pencarian.
5. Apabila matkul yang bentrok masih ada namun jumlah langkah sudah mencapai batas, stop pencarian (untuk keluar dari loop apabila tidak ditemukan tetangga yang lebih baik)