**IF 3170 INTELEGENSIA BUATAN**

**TUGAS BESAR I PENJADWALAN KEGIATAN MENGUNAKAN ALGORITMA *LOCAL SEARCH***

**Oleh:**

**13514040 Devin Lukianto**

**13514045 Elvina R. K. Situmorang**

**13514058 Jason Jeremy Iman**

**13514077 Atika Azzahra Akbar**



**PROGRAM STUDI TEKNIK INFORMATIKA**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**2016**

**BAB I**

**DESKRIPSI PERSOALAN**

1. **Deskripsi Persoalan**

Jadwal yang akan digunakan dalam tugas memiliki spesifikasi berikut:

1. Jadwal adalah sebuah nama kegiatan, waktu, dan tempat
2. Setiap jadwal memiliki durasi tertentu
3. Terdapat beberapa jadwal yang ruangannya ditentukan dan beberapa jadwal yang ruangannya bebas
4. Pencarian dilakukan dengan mencocokan jadwal terhadap waktu dan ruang dengan tepat

Format jadwal:

Format teks di bawah adalah contoh jadwal yang akan digunakan.

****

Dalam pengerjaan tugas ini terdapat beberapa algoritma yang perlu diterapkan yaitu:

* 1. Hill Climbing Search
  2. Simulated Annealing
  3. Genetic Algorithm

**BAB II**

**TEORI ALGORITMA LOCAL SEARCH**

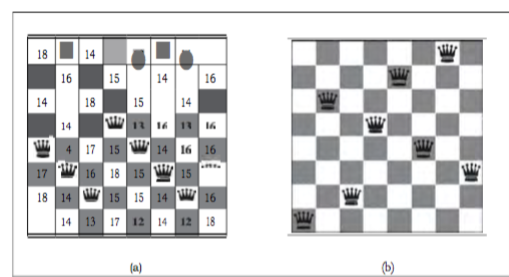
1. ***Hill Climbing Search***

Algoritma Hill Climbing Search adalah salah satu pencarian solusi *local search* yang tidak diperlu dibentuk pohon pencarian. Node pada algoritma hanya perlu menyimpan state dan nilai fungsi objektifnya (nilai *heuristic).*

Algoritma ini juga sering disebut sebagai *greedy local search* karena hanyamemilih *state* tetangga yang menuju ke solusi lebih baik (*good neighbor)* tanpa memikirkan *state* setelahnya yang harus dipilih. Jika tidak lagi ditemukan tetangga yang lebih baik, *current* node akan menjadi solusi paling optimal. Di bawah ini adalah pseudocode untuk fungsi Hill Climbing:

|  |
| --- |
| **function** Hill-Climbing(problem)  state local maximum  current  Make-Node(problem, Initial-State)  loop do  neighbor  Highest-valued successor of current  if neighbor.Value < current.Value then   current.STATE  current  neighboor |

Contoh pemanfaatan algoritma *Hill Climbing* dengan penyelesaian soal 8 *Queens* seperti di gambar bawah ini. Pada gambar (a) nilai *heuristic*-nya adalah 17, tetapi dengan menggunakan algoritma Greedy Descent, hanya dengan 5 tahapan penempatan para ratu berubah menjadi seperti di gambar (b) dengan nilai *heuristic* persoalan berkurang menjadi 1.



Algoritma ini terbukti mampu mendapatkan hasil yang mendapatkan solusi, namun sayangnya solusi tersebut sering kali terjebak contohnya pada *local maximum,* seperti soal 8 *Queens* sebelumnya yang mendapatkan solusi dimana nilai herusiticnya adalah 1*, ridges,* atau *plateaux.*

1. ***Simulated Annealing***

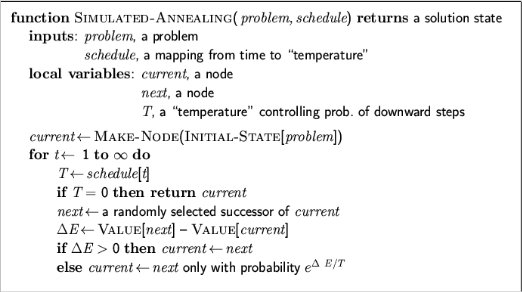
*Hill climbing* adalah algoritma yang tidak pernah mengambil langkah *downhill* atau *bad move*, sehingga dapat stuck pada *local maximum*. Algoritma *purely random walk* dapat selalu menyelesaikan masalah dengan tepat, namun algoritma ini tidak efektif. Sehingga untuk menyelesaikan masalah ini diciptakan algoritma *Simulated Annealing*. Algoritma *Simulated Annealing* adalah sebuah algoritma yang menggabungkan algoritma *hill climbing* dan *random walk*.

Algoritma ini memperbolehkan untuk melakukan *bad move*. Penentuan dari boleh atau tidak melakukan *bad move* dipengaruhi oleh probabilitas. Probabilitas sendiri terdiri dari 3 macam, yaitu

1. Probabilitas tetap, contoh
2. Probabilitas berkurang dipengaruhi oleh faktor waktu
3. Probabilitas berkurang dipengaruhi oleh faktor waktu dan yang meningkat.

Pada tipe ini, probabilitas diperoleh dari persamaan :

Dari tipe 2 dan 3, temperatur juga mempengaruhi probabilitas. Jika temperatur tinggi, maka akan cenderung memperbolehkan semua *moves* (*Random Walk*). Sementara ketika temperatur rendah akan melakukan algortiam *Stochastic Hill-Climbing*. Di bawah ini adalah *pseudocode* untuk algoritmta *Simulated Annealing* :



1. ***Genetic Algorithm***

*Genetic Algorithm* adalah algoritma yang menyelesaikan suatu problema dengan pendekatan yang menyerupai proses *natural selection* dan evolusi biologi. Algoritma ini akan terus menciptakan dan mengubah solusi-solusi baru (spesies). Tiap spesies akan menghasilkan populasi baru yang berevolusi ke solusi yang lebih baik.

Implementasi dari algoritma ini terdiri dari beberapa langkah diantaranya,

1. Membuat populasi baru, menciptakan beberapa solusi *random*

2. *Selection*, memilih solusi terbaik dari populasi tersebut

3. *Crossing*, melakukan perkawinan (penggabungan solusi) dari solusi-solusi yang sudah terbentuk dari hasil *selection*

4. *Mutation*, merubah bagian dari solusi untuk menghasilkan solusi yang baru

Di bawah ini adalah *pseudocode* untuk *Genetic Algorithm*,

|  |
| --- |
| function GeneticAlgorithm()  // start with an initial time  t := 0;  // initialize a usually random population of  // individuals  initpopulation P (t);  // evaluate fitness of all initial individuals of  // population  evaluate P (t);  // test for termination criterion (time, fitness, etc.)  while not done do  // increase the time counter  t := t + 1;  // select a sub-population for offspring  // production  P' := selectparents P (t);  // recombine the "genes" of selected parents  recombine P' (t);  // perturb the mated population stochastically  mutate P' (t);  // evaluate it's new fitness  evaluate P' (t);  // select the survivors from actual fitness  P := survive P,P' (t);  end GA. |

**BAB III**

**SOLUSI PERSOALAN**

1. **Pendifinisian Variabel, Domain, dan *Constraints***

**2.a Variabel**

Variabel pada persoalan ini adalah Matkul[i]. Dimana tiap variabel akan dilakukan penetapan kelas, jam mulai, dan hari mata kuliah. Contoh pendefinisian variabel adalah sebagai berikut:

Matkul[1] = { Kelas: y, Jam mulai: z, Hari: q}

**2.b Domain**

Setiap variable memilik tiga domain yaitu domain kelas (Domain\_kelasi), domain jam mulai (Domain\_jam\_mulaii), dan domain hari (Domain\_harii). Anggota domain untuk tiap variabel berbeda-beda sesuai dengan data yang diberikan. Contoh berikut adalah domain untuk variabel Matkul[1]:

Domain\_kelas1 = {7602}

Domain\_jam\_mulai1 = {07.00, 08.00, 09.00, 10.00, 11.00, 12.00}

Domain\_hari1 = {1, 2, 3, 4, 5}

**2.c Constraints**

Untuk tiap variabel terdapat constraints:

1. Jika Matkul[i].hari = Matkul[j].hari, Matkul[i].Jam\_mulai = Matkul[j].Jam\_mulai maka Matkul[i].kelas ≠ Matkul[j].kelas.
2. Jika Matkul[i].hari = Matkul[j].hari, Matkul[i].kelas = Matkul[j].kelas maka :

* Matkul[i].Jam mulai >= Matkul[j].Jam mulai
* Matkul[i].Jam mulai < Matkul[j].Jam mulai + durasi Matkul j
* Matkul[j].Jam mulai >= Matkul[i].Jam mulai
* Matkul[j].Jam mulai < Matkul[i].Jam mulai + durasi Matkul i

1. Matkul[i].hari = hari dimana Matkul[i].kelas diperbolehkan
2. Matkul[i].Jam mulai >= Jam Matkul[i].kelas boleh mulai dipakai
3. Matkul[i].Jam mulai + durasi Matkul i <= Jam terakhir Matkul[i].kelas sebelum ditutup
4. **Source Code**
5. **Hasil Program**

**BAB IV**

**DAFTAR REFERENSI**

* http://kuliah.itb.ac.id
* Struat, Russell, Peter Norvig. 2003. *Artificial Intelligence A Moderen Approach (Thrid Edition). New Jersey: Pearson Education, Inc.*