Laporan Tugas Besar I
IF3170 Inteligensia Buatan
Penggunaan *Local Search* untuk Penjadwalan Kegiatan

FANDA YULIANA PUTRI	**13514023**
DHARMA KURNIA SEPTIALOKA	**13514028**
MUHAMMAD KAMAL NADJIEB	**13514054**
HISHSHAH GHASSANI	**13514056**
ARNETTHA SEPTINEZ	**13514093**

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2016

A. Deskripsi Persoalan

Local search adalah algoritma yang melakukan pencarian solusi dengan metode pembelajaran. Pada tugas kali ini, akan dilakukan pengaturan jadwal kegiatan menggunakan algoritma-algoritma *local search*.

Jadwal yang akan diatur memiliki spesifikasi sebagai berikut:

a. Jadwal adalah nama kegiatan (mata kuliah), waktu (jam mulai dan jam selesai), dan ruangan
b. Setiap kegiatan memiliki durasi
c. Terdapat kegiatan yang ruangannya sudah ditentukan dan ada juga yang bebas
d. Pencarian dilakukan dengan mencocokkan jadwal terhadap waktu dan ruangan yang tepat.

Dalam pengerjaan tugas kali ini, digunakan tiga algoritma *local search*, yaitu *Hill Climbing*, *Simulated Annealing*, dan *Genetic Algorithm*. Setiap hasil program penjadwalan program akan dibandingkan dan dihitung seberapa baik algoritma berdasarkan jumlah bentrokan antar jadwal kegiatan.

B. Teori Dasar

1. Hill Climbing

Ide dasar dari algoritma ini adalah menyimpan *current state* dan memperbaikinya secara lokal. Secara umum, langkah algoritma *Hill Climbing* adalah sebagai berikut:

a. Mulai keadaan awal. Jika merupakan solusi, maka berhenti. Jika bukan, maka keadaan ini adalah solusi sementara.
b. Cari solusi lain (tetangga) dan evaluasi nilainya.
c. Jika solusi tetangga lebih baik daripada solusi sementara, set solusi sementara menjadi solusi tetangga tersebut.
d. Jika solusi tetangga tidak lebih baik daripada solusi sementara, abaikan.
e. Ulangi terus sampai n kali iterasi.

Kelemahan algoritma ini adalah adanya kemungkinan solusi yang ditemukan tidak optimal. Hal ini terjadi karena iterasi hanya dilakukan n kali, padahal solusi optimal mungkin saja ditemukan setelah n kali.

Pada tugas ini, solusi dikatakan lebih baik apabila jumlah konfliknya lebih sedikit. Jumlah konflik adalah jumlah mata kuliah yang tabrakan (jam dan ruangannya sama).

2. Simulated Annealing

Algoritma ini mirip dengan algoritma *Hill Climbing*, hanya saja apabila ada solusi yang lebih buruk, solusi tersebut masih mungkin diterima asal temperaturnya masih tinggi. Algoritma ini didasarkan pada teknik dalam bidang metalurgi, yakni proses *annealing.* Agar dapat terbentuk susunan kristal yang sempurna, diperlukan pemanasan sampai suatu tingkat tertentu, kemudian dilanjutkan dengan pendinginan yang perlahan-lahan dan terkendali dari materi tersebut. Pemanasan materi di awal proses annealing, memberikan kesempatan pada atom-atom dalam materi itu untuk bergerak secara bebas, dalam hal ini menerima solusi yang lebih buruk, mengingat tingkat energi dalam kondisi panas ini cukup tinggi. Proses pendinginan yang perlahan-lahan memungkinkan

atom-atom yang tadinya bergerak bebas itu, pada akhirnya menemukan tempat yang optimum, di mana energi internal yang dibutuhkan atom itu untuk mempertahankan posisinya adalah minimum.

Algoritma ini memiliki nilai probabilitas untuk menentukan apakah solusi yang lebih buruk bisa diterima atau tidak. Secara umum, langkah algoritma *Simulated Annealing* adalah sebagai berikut:

a. Mulai keadaan awal. Jika merupakan solusi, maka berhenti. Jika bukan, maka keadaan ini adalah solusi sementara.
b. Cari solusi lain (tetangga) dan evaluasi nilainya.
c. Jika solusi tetangga lebih baik daripada solusi sementara, set solusi sementara menjadi solusi tetangga tersebut.
d. Jika solusi tetangga tidak lebih baik daripada solusi sementara, periksa nilai probabilitas. Apabila nilai tersebut masih wajar, terima solusi tetangga tersebut. Apabila tidak, abaikan solusi tetangga.
e. Ulangi terus sampai nilai probabilitas hampir sama dengan nol dan sudah mencapai temperatur minimum.

Algoritma ini lebih memungkinkan untuk mendapatkan solusi optimal daripada algoritma *Hill Climbing* karena masih menerima solusi yang lebih buruk, sehingga kemungkinan untuk terjebak dalam maksimum lokal lebih kecil. Namun, masih ada kemungkinan tidak ditemukan solusi optimal. Hal ini terjadi jika nilai probabilitasnya sudah nol padahal solusi masih berhenti di maksimum lokal.

3. Genetic Algorithm

Algoritma ini memproyeksikan proses genetik yang terjadi di alam pada permasalahan yang ingin diselesaikan. Terdapat 4 tahapan yaitu:

a. Inisialisasi

   Memulai keadaan awal dengan membentuk populasi dari solusi secara acak. Masing-masing solusi dalam populasi dihitung *Fitness Value*-nya (nilai yang merepresentasikan seberapa baik solusi tersebut) dan probabilitas terpilihnya solusi tersebut di tahap berikutnya.

   Nilai probabilitas diambil dari akumulasi *Fitness Value* yang telah dinormalisasi. Normalisasi dilakukan dengan cara membagi *Fitness Value* dari sebuah individu dengan jumlah dari seluruh *Fitness Value* individu yang ada.

b. Seleksi

   Melakukan seleksi pada solusi yang dilakukan berdasarkan nilai probabilitas yang telah dihitung sebelumnya. Semakin besar nilai probabilitasnya, semakin besar kemungkinan solusi tersebut terpilih untuk dibawa ke tahap selanjutnya.

c. Operasi genetik

   i. *Cross-over*

      Untuk melakukan *cross-over*, diambil dua buah solusi dari populasi solusi. Kemudian kedua solusi tersebut dipartisi. Jika solusi memiliki panjang n dan solusi pertama dipartisi pada panjang ke-p, maka solusi kedua dipartisi pada panjang ke-(n-p). Hasil partisi dari solusi pertama kemudian digabung dengan hasil partisi solusi kedua.

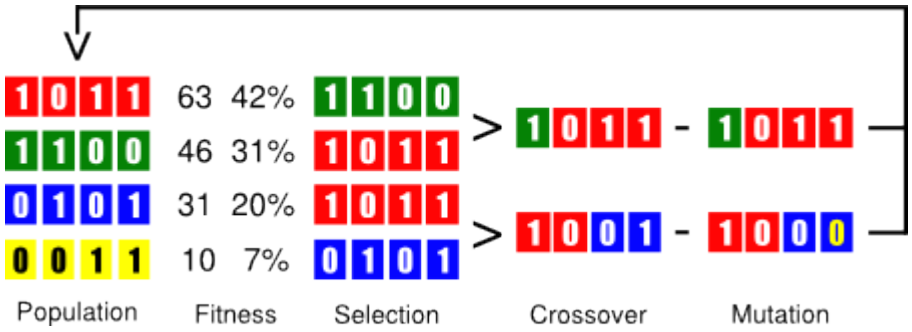
ii. Mutasi

Mutasi dilakukan dengan cara memilih sebagian kecil dari solusi, dilanjutkan dengan mengacak nilainya.

d. Terminasi

Terminasi merupakan tahap dilakukannya pemeriksaan terhadap setiap individu sehingga ditemukan hasil minimal. Jika nilai minimal (konflik) yang ditemukan sama dengan 0, maka algoritma diberhentikan, sebaliknya, jika nilai minimal yang ditemukan bukan sama dengan 0, maka algoritma akan terus diulang sampai batas yang ditentukan.

Ilustrasi proses *Genetic Algorithm*



(Sumber : http://ai-maker.atrilla.net/the-%EF%BB%BFgenetic-algorithms/)

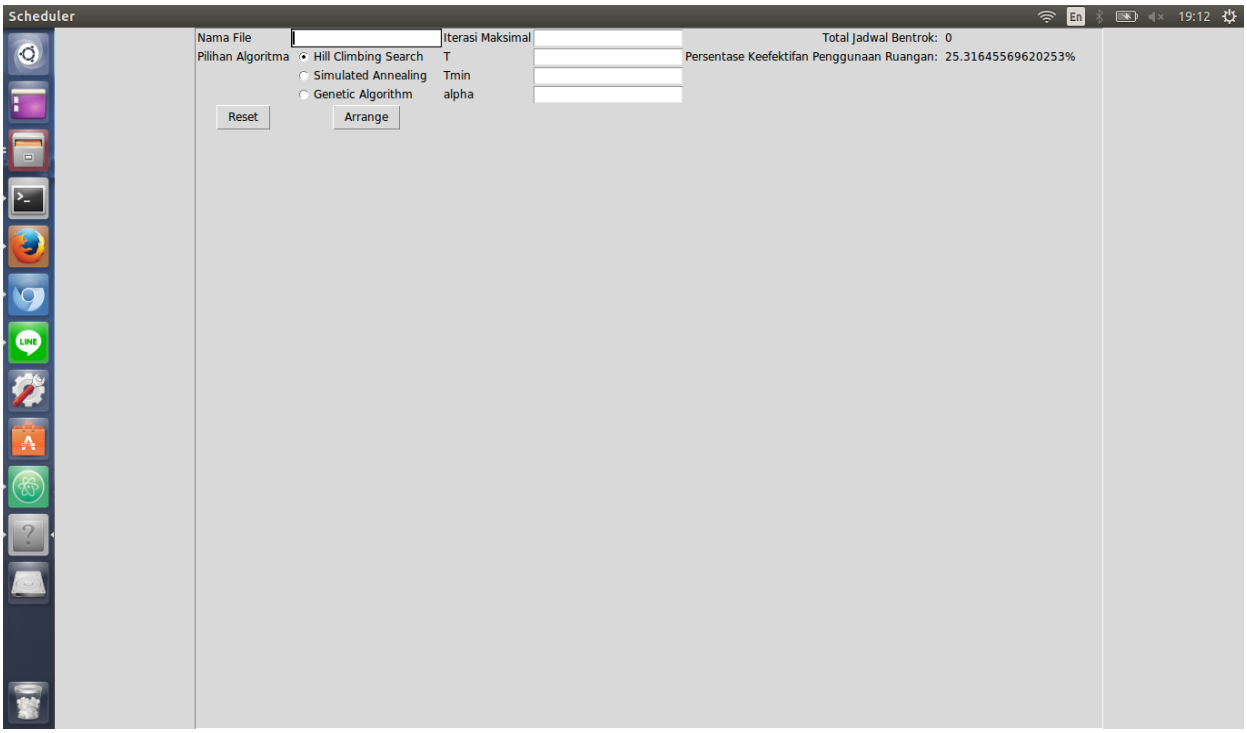
C. Penyusunan Jadwal sebagai CSP

- Variabel : Nama kegiatan, ruangan, hari, jam mulai, jam selesai, dan jumlah SKS
- Domain :
  - Nama kegiatan = {seluruh mata kuliah yang ingin dilakukan pengaturan jadwal}
  - Ruangan = {seluruh ruangan yang terdapat di daftar ruangan}
  - Hari = {0, 1, …, 4}; 0 = Senin, 1 = Selasa, … , 4 = Jumat
  - Jam mulai = {7, 8, 9, …, 17}
  - Jam selesai = {8, 9, 10, …, 18}
  - Jumlah SKS = {1, 2, …, n}; n = integer
- Konstrain :
  - Selisih jam selesai dan jam mulai suatu kegiatan adalah jumlah SKS kegiatan tersebut
  - Hari untuk suatu kegiatan harus merupakan subset dari hari yang tersedia dari ruangan yang dipilih dan hari yang dapat digunakan kegiatan tersebut
  - Jam mulai untuk suatu kegiatan harus lebih dari atau sama dengan jam mulai ruangan yang dipilih untuk kegiatan tersebut
  - Jam selesai untuk suatu kegiatan harus kurang dari atau sama dengan jam selesai ruangan yang dipilih untuk kegiatan tersebut
  - Kegiatan pada ruangan dan hari yang sama harus memiliki jam mulai yang berbeda
  - Dua buah kegiatan pada ruangan dan hari yang sama, jika salah satunya memiliki jam mulai kurang dari jam mulai kegiatan lainnya, jam selesai kegiatan tersebut harus kurang dari jam mulai kegiatan lainnya

- Dua buah kegiatan pada ruangan dan hari yang sama, jika salah satunya memiliki jam mulai lebih dari jam mulai kegiatan lainnya, jam mulai kegiatan tersebut harus lebih dari jam selesai kegiatan lainnya

D. Hasil Pengerjaan

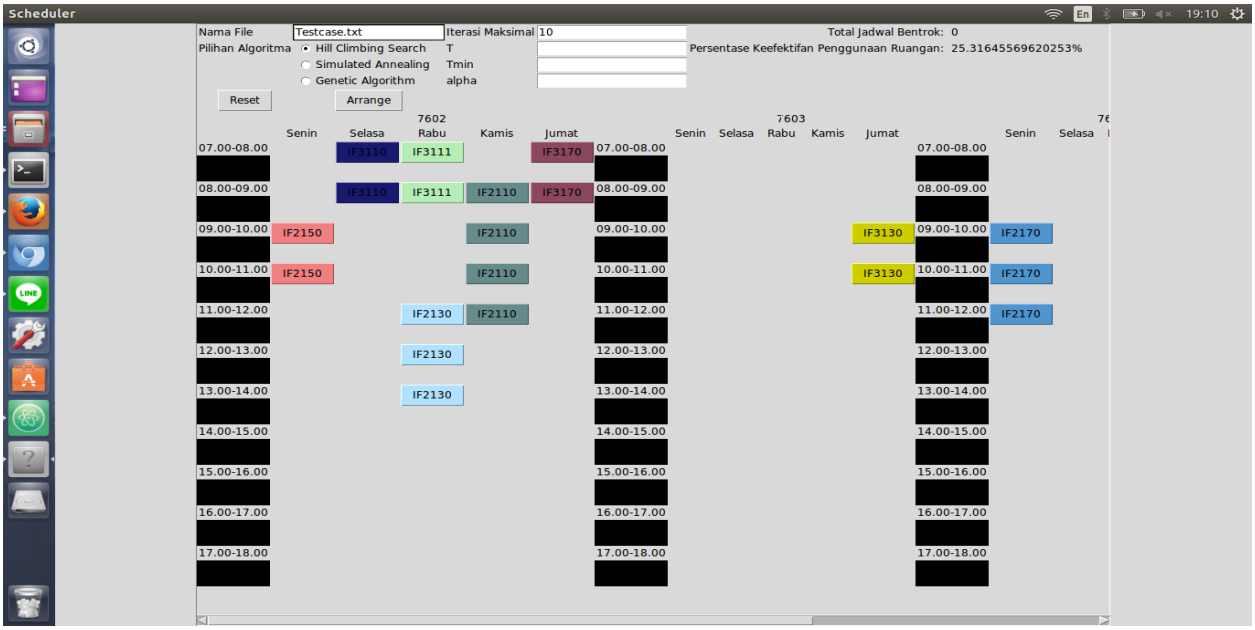
(1)



Penjelasan:

Ini merupakan tampilan awal saat program main dijalankan. Masukan nama file adalah kasus jadwal dan ruangan yang diberikan. Lalu “iterasi maksimal” digunakan untuk hill-climbing dan genetic algorithm, sedangkan “T, Tmin, dan alpha” adalah masukan untuk algoritma simulated annealing. Program akan mengeluarkan jumlah konflik dengan “total jadwal bentrok” dan persentase keaktifan dengan rumus jumlah slot yang diisi / jumlah slot yang disediakan.

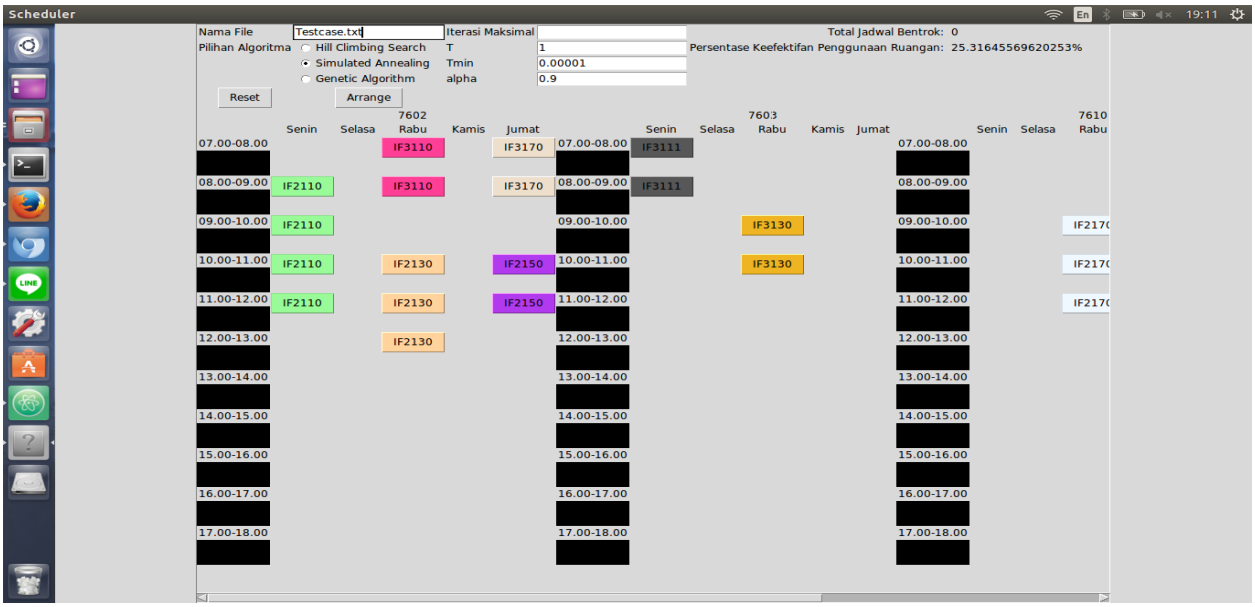
(2)



Penjelasan:

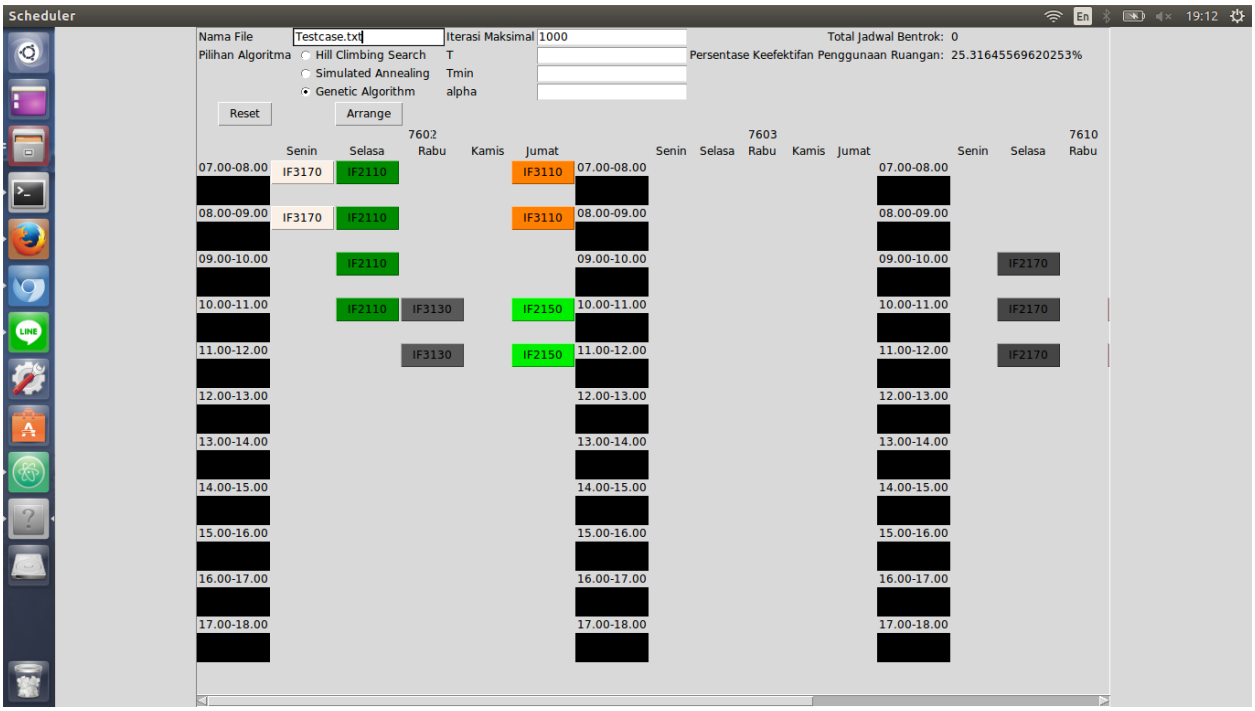
Kami menggunakan persoalan pada testcase yang diberikan asisten, yakni 4 ruangan dengan slot waktu sebanyak 79 jam dan 8 mata kuliah yang membutuhkan waktu kuliah sebanyak 20 SKS atau 20 jam. Dengan menggunakan hill climbing, dengan jumlah halt sebanyak 10, program akan terus men-generate solusi selama konflik yang ditemukan lebih baik atau belum mencapai jumlah percobaan yang diminta, dalam kasus ini 10. Dengan menggunakan hill-climbing implementasi kami, konflik atau jadwal bentrok berjumlah 0 alias tidak ada bentrok, maka persentase keefektifan adalah 20/79 = 25.31%.

(3)



Penjelasan:
Kami menggunakan persoalan pada testcase yang diberikan asisten, yakni 4 ruangan dengan slot waktu sebanyak 79 jam dan 8 mata kuliah yang membutuhkan waktu kuliah sebanyak 20 SKS atau 20 jam. Dengan menggunakan simulated annealing, dengan masukan T= 1, Tmin= 0.00001, dan alpha= 0.9, program akan terus men-generate solusi selama konflik yang ditemukan belum 0, lebih baik atau boleh lebih buruk dengan probabilitas = exp(Ebaru-Eawal / T). Dengan menggunakan simulated-annealing implementasi kami, konflik atau jadwal bentrok berjumlah 0 alias tidak ada bentrok, maka persentase keefektifan adalah 20/79 = 25.31%.

(4)



Penjelasan:
Gambar diatas merupakan hasil penyelesaian test case “Testcase.txt” dengen rincian berupa 4 ruangan dengan slot waktu sebanyak 79 jam dan terdapat 8 mata kuliah dengan total 20 sks atau setara dengan 20 jam waktu pertemuan. Persoalan diselesaikan dengan menggunakan genetic algorithm, masukan lainnya pada permulaan proses adalah jumlah iterasi maksimal. Pada kasus ini iterasi maksimalnya adalah 1000. Iterasi maksimal dimaksudkan untuk menghentikan iterasi jika pencarian solusi terjebak pada local minimal atau lokal maksimal. Setelah program dieksekusi, hasilnya adalah tidak ada jadwal yang bentrok (0 konflik) dengan keefektifan penggunaan ruangan sebesar 25,31%.