Hill Climbing

Dasar Teori

Hill Climbing adalah salah satu teknik optimasi yang termasuk dalam Local Search. Ide utama dari Hill Climbing adalah untuk mengubah nilai suatu variabel yang lebih dekat ke solusi optimum, yang diiterasi terus menerus.

Langkah-langkah hill climbing adalah sebagai berikut:

1. Assign nilai random kepada variabel-variabel yang ada
2. Assign nilai kepada variabel, yang dapat menghasilkan hasil yang lebih dekat kepada solusi optimum
3. Ulang langkah 2 sampai solusi optimum ditemukan atau iterasi maksimal dicapai

Masalah utama Hill Climbing adalah kecenderungannya untuk terjebak di solusi maksimum lokal, karena pada maksimum lokal tidak ada hasil yang lebih dekat ke solusi optimum dengan hanya mengubah satu elemen.

Masalah lain Hill Climbing adalah grafik “datar” yang menyebabkan algoritma tidak dapat memutuskan ke mana hasil yang akan dipilih akan bergerak, dan memungkinkan algoritma untuk memilih hasil yang mungkin tidak menjurus ke solusi optimum.

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)