DESKRIPSI MASALAH

Mencari nilai minimum dari suatu fungsi bisa dilkakukan dengan berbagai cara. Dengan batasan parameter fungsi yang ada, bisa dicoba sebanyak z-kali untuk menentukan niali minimum dari suatu fungsi. Dimana z positif dan nilainya tidak terdefinisi. Tugas kali ini dikerjakan menggunakan algorimta *Simulated Annealing*. Dengan definisi fungsi sebagai berkut :

dengan batasan nilai dan .

Intinya algoritma *Simulated Annealing* yang digunakan pada masalah ini akan membangkitkan setiap titik percobaan dengan metode randomization untuk mencari nilai minimum dari suatu fungsi. Program akan merandom nilai *x1,x2* dan dicari berapa nilai f(*x1,x2*) dari fungsi diatas. Setelah itu program akan melakukan  *looping* sebanyak z-kali untuk mendapatkan nilai minimum dari f(*x1,x2*) dengan *x1,x2* didapatkan dari proses randomization.

ANALISYS DAN STRATEGI

Strategi yang digunakan untuk menyelesaikan permasalahan diatas adalah membuat program dengan algoritma *Simulated Annealing*.

Dalam algoritma *Simulated Annealing* terdapat beberapa parameter yaitu:

T=sebagai salah satu faktor perhitungan probability.

∆T=sebagai parameter untuk merevisi nilai t

perulangan=sebagai salah satu faktor jumlah pemilihan calon kandidat.

Dalam laporan pengerjaan program. T bebas ditentukan nilainya. Dalam program yang dibuat T disebut tawal. T diisi dengan angka yang antara 0-100 dan di program ini di tulis dengan hardcode 91. ∆T pada program ini diinisialisasi dengan nilai 0.167. ∆T digunakan untuk mengubah nilai T dengan cara Tnext = Tsekarang - ∆T. Proses mengubah nilai T ini dilakukan pada perulangan ke-n dimana n mod 21 = 0 dan nilai T>∆T. Pada program ini perulangan dilakukan sebanyak z-kali, z pada program ini dihardcode dengan nilai 1000000 (satu juta). Dilakukan pengisian nilai-nilai diatas untuk mendapatkan nilai yang optimum.

Penjabaran algoritma *Simulated Annealing* pada program yang dibuat untuk mencari nilai minimum.

1. Inisialisasi nilai T(tawal)=91, ∆T(deltat)=0.167, jumlah perulangan(i)=1000000
2. Pindahkan nilai T ke dalam variable (t).
3. Generate titik awal serta dihitung nilai f(*x1,x2*) dan dimasukan kedalam variable awal dalam bentuk array ([*x1,x2*,f(*x1,x2*)])
4. Generate calon kandidat serta dihitung nilai f(*x1,x2*) dan dimasukan kedalam variable next dalam bentuk array ([*x1,x2*,f(*x1,x2*)])
5. Lakukan pengecekan apakah nilai f(*x1,x2*) dari next < nilai f(*x1,x2*) dari awal **atau** nilai exp(-(next[2]-awal[2])/t)>dari random(0,1). Jika salah satu kondisi terpenuhi maka next berubah menjadi awal. Artinya variable awal diisi dengan next.

\*catatan. next[2] = nilai f(*x1,x2*) dari next // begitu juga awal.

1. Lakukan pengecekan jika nilai I mod 21 = 0 dan nilai t>∆T maka revisi t menjadi t=t-∆T.
2. Lakukan decrement nilai i.
3. Lakukan penecekan nilai I jika i>0 lakukan kembali langkah 3-8, jika i<=0, lakukan langkat 9.
4. Tampilkan nilai-nilai yang ada pada variable awal, yaitu awal[0] sebagai *x1*, awal[1] sebagai *x2*, awal[2] sebagai f(*x1,x2*), serta nilai dari parameter t sebagai t akhir dan tawal sebagai t awal.

HASIL PENGUJIAN

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | X1 | X2 | Hasil | T |
| 1 | 7.93195943531514 | 9.50442747777642 | -18.8078454941774 | 91 |
| 2 | -7.83464542760435 | 9.7604722177762 | -18.6588501071636 | 91 |
| 3 | 8.026201238925562 | -9.774536514836388 | -19.07719433317454 | 91 |
| 4 | -8.247710175771164 | -9.645338890515431 | -18.830806081999878 | 91 |
| 5 | 8.05506831719964 | -9.6641436546277 | -19.208500811328 | 91 |
| 6 | 8.095178463356941 | -9.648457928664637 | -19.19005865947056 | 99 |
| 7 | -8.192948673842007 | -9.444296710729327 | -18.559448322265364 | 99 |
| 8 | 8.123552218881493 | -9.808820669920458 | -18.94560701527032 | 99 |
| 9 | 7.90960450849084 | -9.767712908993449 | -18.900230396730098 | 99 |
| 10 | 8.091705265767807 | -9.518089174903183 | -18.985796696089988 | 99 |

SCREEN SHOOT

