LAPORAN TUGAS PROGRAM 1

ARTIFICIAL INTELIGENCE

SEMESTER GENAP 2017/2018

SIMULATED ANEALLING

NAFASA MUTH MA'NAH

1301150441

IF 39-07

S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

UNIVERSITAS TELKOM

BANDUNG

A. Deskripsi Masalah

Pada tugas program 1 ini diberikan fungsi sebagai berikut:

$$f(x_1, x_2) = \left(4 - 2{,}1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2$$

dengan batasan $-10 \leq x1 \leq 10$ dan $-10 \leq x2 \leq 10$.

Setelah itu, kita disuruh untuk mencari nilai minimum yang akan dihasilkan oleh fungsi tersebut. Untuk mencari nilai minimum diberikan batasan untuk x_1 dan x_2 yaitu dari -10 hingga 10. Dalam mencari nilai minimum tersebut kita membutuhkan metode yang dapat digunakan agar solusi yang didapat bisa akurat, konstan, serta tepat. Ada beberapa macam metode yang dapat digunakan untuk mencari nilai minimum yaitu *Hill climbing, Simulated Annealing, Generate and Test, Brute Force, Knapsack,* dll. Dalam persoalan ini metode yang digunakan adalah Simulated Annealing.

B. Rancangan Metode

Metode *Simulated Annealing* ini juga memanfaatkan analogi cara pendinginan dan pembekuan metal atau logam menjadi sebuah struktur kristal dengan energi yang minimal. Pada proses ini pertama-tama logam dipanaskan hingga mencapai suhu yang tinggi. Pada saat logam mencapai suhu yang tinggi maka molekul-molekul di dalam logam akan bergerak dengan bebas. Kemudian ketika logam sudah mencapai suhu tertinggi, logam didinginkan dengan cara menurunkan suhu logam secara perlahan dan stabil. Dengan cara pendinginan ini molekul-molekul logam dapat berpindah-pindah secara bebas dengan waktu yang lama sehingga diasumsikan dengan perpindahan logam secara bebas tersebut pada akhirnya akan menemukan tempat dimana sebuah molekul mengeluarkan energI terkecil untuk mempertahankan tempatnya. Pada metode ini diasumsikan bahwa kita akan berpindah pindah dari titik satu ke titik lainnya secara perlahan dari Temperature Awal yang telah kita tentukan hingga Temperature Akhir sehingga dalam perpindahan tersebut, kita bisa menemukan titik terendah atau nilai minimum dari fungsi yang telah diberikan. Apabila menggunakan *Hill climbing*(HC) bisa terjebak pada minimum atau maksimum local. Terdapat banyak cara yang bisa digunakan untuk mengatasi masalah tersebut, diantaranya:

- Mencoba algoritma *hill climbing* dengan menggunakan beberapa titik awal yang berbeda.
- Meningkatkan ukuran ketetanggaan sehingga terdapat lebih banyak ruang pencarian pada setiap langkah.

Bagaimanapun kedua cara tersebut tidak efektif ketika menggunakan algoritma simple *hill climbing*. Cara yang membolehkan langkah menuju *state* yang lebih buruk daripada *current state*. Cara efektif agar tidak terjebak pada minimum local adalah dengan cara membolehkan langkat menuju *state* yang memiliki biaya(*cost*) lebih besar daripada *current*

state. Metode yang melakukan langkah seperti itu adalah metode SA, dimana dalam pemilihan *new state* dilakukan secara acak dengan probabilitas tertentu. Jika *new state* lebih baik dibandingkan *current state* maka metode SA akan memilih *new state*.

Berikut ini langkah-langkah untuk mencari nilai minimum dari persoalan program di atas:

1. Melakukan inisiasi terhadap temperature awal, temperature akhir, dan alfa yang dianalogikan sebagai titik didih logam, titik beku logam, dan penurunan terhadap logam dari titik didih ke titik beku.
2. Mengenerate nilai terendah dengan cara mencari x_1 dan x_2 yang telah dirandom dengan batasan -10 sampai dengan 10 dan disubtitusikan ke dalam fungsi berikut:

   $f(x_1, x_2) = \left(4 - 2{,}1x_1^2 + \frac{x_1^4}{3}\right) x_1^2 + x_1 x_2 + (-4 + 4x_2^2)x_2^2$

3. Setelah mendapat nilai terendah, kemudian nilai tersebut kita anggap saja sebagai *minimum state*.
4. Lakukanlah perulangan apabila temperature awal masih lebih besar daripada temperature akhir.
5. Untuk kelangkah selanjutnya kita lakukan di dalam perulangan untuk pindah ke *state* lain lakukan langkah 2. Kemudian bandingkan *minimum state* yang sudah kita simpan dengan *new state*. Terdapat 2 kondisi dalam pencarian nilai terendah saat dilakukan perbandingan yaitu:
   a. Jika nilai fungsi yang baru lebih kecil daripada *minimum state* yang disimpan, maka nilai x_1 dan x_2 akan diganti dengan nilai x_1 dan x_2 yang baru, kemudian *minimum state* diganti dengan *new state*.
   b. Jika nilai *new state* lebih besar dari *minimum state* yang disimpan maka kita akan membuat nilai random baru dari interval 0 sampai 1 yang kemudian akan dibandingkan dengan probabilitas yang diperoleh dari rumus:

      $$p(\Delta E) = e^{-\Delta E/T}$$

      Keterangan:
      ΔE = delta energy(menyatakan fungsi biaya atau evaluasi)
      T = temperature saat ini(titik didih)
      E = bilangan eksponen
      Terdapat 2 kondisi dalam perbandingan nilai random dengan nilai probabilitas yaitu:
      - Jika nilai probabilitas lebih besar dari nilai random, maka nilai x_1 dan x_2 yang lama akan diganti dengan nilai x_1 dan x_2 yang baru.
      - Jika nilai probabilitas lebih kecil dari nilai random maka tidak usah melakukan apapun.
6. Kemudian mengalikan temperature awal dengan nilai alfa sehingga nilai temperature akan berkurang sedikit demi sedikit.

C. User manual dari penggunaan aplikasi berupa screenshot

```
-1.0306766700788417
-1.0306766700788417
-1.0306766700788417
-1.0306766700788417
-1.0306766700788417
[Finished in 13.1s]
```

G:\NEW\Tugas AI\SimulatedAnnealing.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

SimulatedAnnealing.py

```
1  import random
2
3  def HeuristicFunction(x1,x2):
4      left = (4-(2.1*(x1**2))+((x1**4)/3))*(x1**2)+(x1*x2)+(-4+(4*(x2**2)))*(x2**2)
5      return left
6
7  def Random():
8      return random.uniform(-10,10)
9  def probability(min,new,awal):
10     pangkat = -1*(new-min)/awal
11     Probabilitas = 2.7182818284**(pangkat)
12     return Probabilitas
13
14 Tawal = 100000
15 Takhir = 0.00001
16 x1 = Random()
17 x2 = Random()
18 minstate = HeuristicFunction(x1,x2)
19 while (Tawal>Takhir):
20     for i in range(1,180):
21         nx1 = Random()
22         nx2 = Random()
23         newstate = HeuristicFunction(nx1,nx2)
24         if (minstate>newstate):
25             minstate=newstate
26             nx1 = x1
27             nx2 = x2
28         else:
29             if (probability(minstate,newstate,Tawal)>random.random()):
30                 nx1 = x1
31                 nx2 = x2
32                 minstate = newstate
33     Tawal = Tawal*0.999
34 for i in range(0,5):
35     print(minstate)
```

```
-1.031557485666105
```

References

Suyanto. (2010). Algoritma Optimasi.

Suyanto. (2014). Artificial Intelligence.