PENCARIAN NILAI FUNGSI MINIMUM DENGAN ALGORITMA *SIMULATED ANNEALING*

LAPORAN TUGAS PROGRAM I

Disusun untuk memenuhi nilai mata kuliah Kecerdasan Buatan (CCH3F3)

Disusun oleh :

Raden Rizky Falih P

(1301154211)

IF-39-01

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

UNIVERSITAS TELKOM

BANDUNG

2017

Daftar Isi

1 Pendahuluan

1.1 Deskripsi Masalah

Permasalahan yang diselesaikan pada dokumentasi ini adalah permasalahan ketika anda mendapatkan suatu fungsi kuadrat, kemudian disuruh mencari nilai optimasi minimumnya dengan suatu solusi x1 dan x2 dan dengan menggunakan metoda *Simulated Annealing (SA)* yang dimana solusi nilai x1 dan x2 didapatkan secara *random*. Untuk kasus ini fungsi yang diselesaikan adalah:

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

Dengan batasan yang sudah ditentukan sebagai berikut : $-10 \leq x_1 \leq 10$ dan $-10 \leq x_2 \leq 10$, dan menggunakan algoritma Simulated Annealing(SA).

1.2 Deskripsi Umum Sistem

Perangkat yang akan diuji adalah sistem sederhana dengan mengimplementasikan metode simulated annealing. Perangkat hanya menampilkan solusi yang didapat, cost (nilai fungsi minimum dari solusi tersebut) dan akurasi model.

1.3 Referensi

- Geltman, K. E. (2014, 2 20). *simulated-annealing*. Retrieved 9 20, 2017, from katrinaeg: http://katrinaeg.com/simulated-annealing.html

2 Rancangan Metode

2.1 Metode Simulated Annealing

Simulated annealing (SA) adalah algoritma yang dapat digunakan untuk mencari pendekatan terhadap solusi optimum global dari suatu permasalahan. Masalah yang membutuhkan pendekatan SA adalah masalah-masalah optimisasi kombinatorial, di mana ruang pencarian solusi yang ada terlalu besar, sehingga hampir tidak mungkin ditemukan solusi eksak terhadap permasalahan itu. Kali ini algoritma Simulated Annealing akan digunakan untuk mencari nilai minimum dari suatu fungsi.

Simulated Annealing berjalan berdasarkan analogi dengan proses annealing yang kurang lebih adalah proses pembentukan Kristal . Pada awal proses SA, dipilih suatu solusi awal, yang merepresentasikan kondisi materi sebelum proses dimulai. Pada awal proses SA, saat parameter suhu (T) diatur tinggi, solusi sementara yang sudah ada diperbolehkan untuk mengalami modifikasi secara bebas.
Kebebasan ini secara relatif diukur berdasarkan nilai fungsi tertentu yang mengevaluasi seberapa optimal solusi sementara yang telah diperoleh. Bila nilai fungsi evaluasi hasil modifikasi ini membaik (dalam masalah optimisasi yang berusaha mencari minimum berarti nilainya lebih kecil/downhill) solusi hasil modifikasi ini akan digunakan sebagai solusi selanjutnya.

Bila nilai fungsi evaluasi hasil modifikasi ini memburuk, pada saat temperatur annealing masih tinggi, solusi yang lebih buruk (uphill) ini masih mungkin diterima, sedangkan pada saat temperatur annealing sudah relatif rendah, solusi hasil modifikasi yang lebih buruk ini mungkin tidak dapat diterima. Dalam tahapan selanjutnya saat temperatur sedikit demi sedikit dikurangi, maka kemungkinan untuk menerima langkah modifikasi yang tidak memperbaiki nilai fungsi evaluasi semakin berkurang. Sehingga kebebasan untuk memodifikasi solusi semakin menyempit, sampai akhirnya diharapkan dapat diperoleh solusi yang mendekati solusi optimal.

2.2 Algoritma Dasar

Untuk algoritma dasar/umum simulated annealing adalah sebagai berikut (Geltman, 2014) :

1. *First, generate a random solution*
2. *Calculate its cost using some cost function you've defined*
3. *Generate a random neighboring solution*
4. *Calculate the new solution's cost*
5. *Compare them:*
    - *If $c_{new} < c_{old}$: move to the new solution*
    - *If $c_{new} > c_{old}$: maybe move to the new solution*
6. *Repeat steps 3-5 above until an acceptable solution is found or you reach some maximum number of iterations.*

Contoh *code* dasar menggunakan *python* (Geltman, 2014):

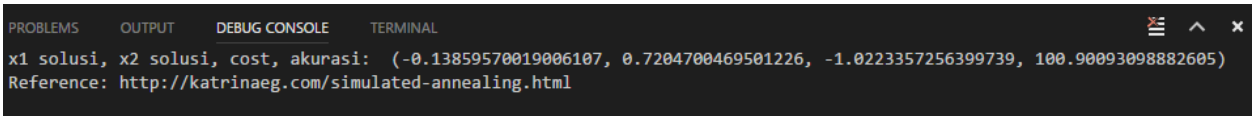
```
from random import random

def anneal(solution):
    old_cost = cost(solution)
    T = 1.0
    T_min = 0.00001
    alpha = 0.9
    while T > T_min:
        i = 1
        while i <= 100:
            new_solution = neighbor(solution)
            new_cost = cost(new_solution)
            ap = acceptance_probability(old_cost, new_cost, T)
            if ap > random():
                solution = new_solution
                old_cost = new_cost
            i += 1
        T = T*alpha
    return solution, old_cost
```

2.3 Implementasi Metode

Source code implementasi:

```python
from random import uniform
import math
import random

#cost = fungsi
def cost(x1,x2):
    fungsiK = (4-(2.1*(x1**2))+((x1**4)/3))*(x1**2) + x1*x2 + (-4+(4*(x2**2)))*(x2**2)
    return fungsiK

def acceptance_probability(old_cost, new_cost, temperature):
    if new_cost < old_cost:
        return 1.0
    else:
        return math.exp((old_cost - new_cost) / temperature)

def akurasi(solution):
    akurasis = (1-((solution-(-1.03163))/(-1.03163)))*100
    return akurasis

def SA(x1,x2):
    alpha = 0.9
    temperature = 1.0
    min_temperature = 0.00001
    cost1 = cost(x1,x2)
    while temperature > min_temperature:
        i = 1

        while i<=100:
            #solution2 =  random.random()
            x1_new = uniform(-10,10)
            x2_new = uniform(-10,10)
            cost2 = cost(x1_new,x2_new)
            ap = acceptance_probability(cost1,cost2,temperature)
                if ap > random.random():
                    x1 = x1_new
                    x2 = x2_new
                    cost1 = cost2
                i +=1

            temperature = temperature * alpha
        return x1, x2, cost1 , akurasi(cost1)

x1 = uniform(-10,10)
x2 = uniform(-10,10)
print("x1 solusi, x2 solusi, cost, akurasi: ", SA(x1,x2))
print("Reference: http://katrinaeg.com/simulated-annealing.html")
```

Hasil running program:



Didapat :
X1 = -0.1385...
X2 = 0.7204...
Cost = -1.0223...
Akurasi = 100,900...%

Note: Nilai acuan fr = -1.031... (Global Minimum)