TUGAS: Kecerdasan Buatan (CCH3F3)
2019
DOSEN: NJFA
KELAS: IFIK – 40 – 03

Analisis, desain, dan implementasi algoritma *Simulated Annealing* untuk menemukan nilai minimum dari suatu fungsi

Alanu Dinasti Permana **(NIM: 1302160195)**[1]
Informatic Engineering, School of Computing,
Telkom University, Indonesia.

Email: (alanupermana@gmail.com)

Abstract.
Paper ini memaparkan analisis untuk menemukan nilai minimum dari suatu fungsi menggunakan algoritma *Simulated Annealing*. Dalam paper ini dijelaskan bagaimana mendesain algoritma *Simulated Annealing* baik dari *pseudocode* maupun programnya. Hasil *numeric* yang ditemukan akan dianalisi dengan hasil *exact*. Bahasa yang digunakan adalah *Ruby*.

Kata Kunci: *Simulated Annealing, Ruby, temperature, random*

1. Latar Belakang

Simulated Annealing (SA) adalah salah satu algoritma untuk untuk optimisasi yang bersifat generik. Berbasiskan probabilitas yang digunakan untuk mencari pendekatan terhadap solusi optimum global dari suatu permasalahan. Dalam kasus ini kita akan menggunakan *Simulated Annealing* untuk menemukan nilai minimum dari sebuah fungsi.

2. Deskripsi Masalah

Algoritma *Simulated Anneaaling* menggunakan mekanisme probabilitas dalam membandingkan nilai yang akan dijadikan solusi dengan beberapa ketentuan. Menurut *Kirkpatrick,* ada tiga hal utama yang perlu diperhatikan dalam penggunaan *Simulated annealing* untuk dimemodelkan:

1. Nilai Awal untuk Temperatur yang ditetapkan cukup besar.
2. Kriteria yang dipakai untuk memutuskan apakah *temperature* sistem seharusnya dikurangi atau tidak.
3. Berapa besarnya pengurangan temperature dalam setiap waktu

Pada kasus ini ada beberapa parameter yang harus diperhatikan, berikut adalah parameternya.

- *Temperature* : Suhu yang di *set* dengan nilai 100
- *Final Temperature* : Suhu yang di *set* dengan nilai 0,00001
  Alpha : Menghitung penentu Iterasi Temperature di *set* 0,9999
- X1,X2 : variable yang di *set* Random dengan *range* (-10,10)
- *State* Awal : *State* awak yang berisi X1,X2

2.1 Perancangan Algoritma

Hal yang harus dilakukan agar algoritma terlihat rapih dan efisien adalah dengan menentukan fungsi/method, dan parameter yang dibutuhkan.

- Fungsi yang akan diminimumkan

$$f(x_1, x_2) = -\left(sin(x_1)cos(x_2) + \frac{4}{5}exp\left(1 - \sqrt{x_1^2 + x_2^2}\right)\right)$$

dengan batasan $-10 \leq x_1 \leq 10$ dan $-10 \leq x_2 \leq 10$.

- Parameter
  - **xx1,xx2** : Variabel yang di set 0,0 untuk menghitung nilai *exact*
  - ***Temp*** : Suhu yang di *set* dengan nilai 100
  - ***Final_Temp*** : Suhu yang di *set* dengan nilai 0,00001
  - ***Alpha*** : Menghitung penentu Iterasi *Temperature* di *set* 0,9999
  - **X1,X2** : variable yang di *set Random* dengan *range* (-10,10)
  - **is** : *State* awal yang berisi X1,X2
  - **cS** : *State* sekarang
  - **cC** : Fungsi sekarang
  - **bSolution** : *State* terbaik
  - **bCost** : Fungsi terbaik
  - **nS** : *State* terbaru
  - **nC** : Fungsi terbaru
  - **solusiTemp** : solusi *temperature* yang mendekati 0
  - **solusiBestSolution** : solusi x1, x2
  - **solusiBestCost** : solusi fungsi yang minimum

- *Method*/Fungsi
  - **fungsi(x1,x2),** *Method* untuk mengembalikan nilai dari fungsi
  - **cost(state)**, Method untuk menerima inputan X1,X2 dengan mengembalikan nilai dari **fungsi(x1,x2)**
  - **prob(nC,cC,temp),** *Method* untuk menerima inputan **nC, cC, temp** serta perhitungan probabilitas dengan mengembalikan nilai probabilitas
  - **fungsiRandom(),** *Method* untuk mengembalikan nilai random dengan range (-10,10)
  - **fungsiRandom01(),** *Method* untuk mengembalikan nilai random dengan range (0,1)
  - **simulatedAnnealing(is,cS,cC,bSolution,bCost,temp,final_temp,alpha),** Method untuk menerima inputan **is, cS, cC, bSolution, bCost, temp, final_temp, alpha** serta algortima *Simulated Annealing* dengan mengembalikan nilai **solusiTemp, solusiBestSolution, solusiBestCost**

- Prosedur penerapan algoritma *Simulated Annealing*
  - Inisialisasi minimal 2 buah variabel yang nilainya dipilih secara *random.*
  - Menetapkan suhu awal *temperature.*
  - Menetapkan nilai pendinginan *temperature.*
  - Menetapkan Fungsi hitung
  - Membuat Pengkondisian *Looping* ketika *Temperature* Masih Tinggi, ada 2 kondisi,
    - Jika proses sudah mencapai kriteria, program terhenti, dan keluar dari looping
    - Jika Proses belum mencapai kriteria, maka program akan melakukan perulangan terus dan membandingkan nilai terus.
  - Menginisialisasi kembali 2 buah variabel yang bernilai (ini akan menjadi nilai baru pembanding selama perulangan).
  - Menetapkan Fungsi hitung
  - Membuat pengkondisian hasil fungsi terkecil, terdapat 3 kondisi :
    - Jika Nilai Fungsi Baru ternyata Lebih Kecil dari nilai Fungsi Sekarang,maka Nilai Baru di masukkan ke variabel Fungsi Sekarang. (Mendapatkan nilai terkecil).
    - Jika Nilai Fungsi Baru ternyata Lebih Besar dari nilai Fungsi Sekarang, maka Nilai Baru akan dihitung Probabiltias nya, apakah dia nilai yang optimum atau tidak dengan kondisi apakah exp(-ΔE) > |0…1|* . Jika Benar, ,maka Nilai Baru di masukkan ke variabel Fungsi Sekarang. (Mendapatkan nilai optimum).
    - Jika Nilai Fungsi Baru ternyata Lebih Besar dari nilai Fungsi Sekarang dan Tidak Optimum, maka Nilai Terkecil dan teroptimum tetap di pegang oleh Fungsi Sekarang.
  - Menghitung penentu Iterasi *Temperature*, dengan mendinginkan Temperatur (temperature * pendingin).
  - Mengeluarkan nilai terkecil/teroptimum.
  - ΔE adalah (Nilai Terbesar – Nilai Terkecil/Temp)

- Pseudocode

```
Input: ProblemSize, iterations_max, temp_max
Output: S_best
S_current ← CreateInitialSolution(ProblemSize)
S_best ← S_current
For (i = 1 To iterations_max)
  S_i ← CreateNeighborSolution(S_current)
  temp_curr ← CalculateTemperature(i, temp_max)
  If (Cost(S_i) ≤ Cost(S_current))
    S_current ← S_i
    If (Cost(S_i) ≤ Cost(S_best))
      S_best ← S_i
    End
  ElseIf (Exp( (CostS_current − CostS_i) / temp_curr ) > Rand())
    S_current ← S_i
  End
End
Return (S_best)
```

Gambar 2.1.0 Pseudocode

- Source Code Program

```
def fungsi (x1,x2)
  return -((Math.sin(x1) * Math.cos(x2)) + ((4.0/5.0) * Math.exp(1 - Math.sqrt((x1 **
2) + (x2 ** 2)))))
end

def cost(state)
  return fungsi(state[0], state[1])
end

def prob(nC,cC,temp)
  deltaE = nC - cC
  return Math.exp(-(deltaE)/temp)
end

def fungsiRandom()
  return rand(-10.0..10.0)
end

def fungsiRandom01()
  return rand(0..1)
end

def simulatedAnnealing(is,cS,cC,bSolution,bCost,temp,final_temp,alpha)
  solusiTemp = {}
  solusiBestSolution = {}
  solusiBestCost= {}
  j=1
```

```
  while temp > final_temp do
    for i in 1..100
      x1 = fungsiRandom()
      x2 = fungsiRandom()
      nS = [x1,x2]
      nC = cost(nS)
    end
    if nC < cC then
      cS = nS
      cC = nC
      if cC < bCost then
        bSolution = cS
        bCost = cC
      end
    else
      if (prob(nC,cC,temp) > fungsiRandom01()) then
        cS = nS
        cC = nC
      end
      solusiTemp[j] = [temp]
      solusiBestSolution[j] = [bSolution]
      solusiBestCost[j] = [bCost]
    end
    j = j+1
    temp = temp * alpha
  end
  return solusiTemp, solusiBestSolution, solusiBestCost
end

if __FILE__ == $0
  xx1,xx2 = 0.0,0.0
  temp = 100
  final_temp = 0.00001
  alpha = 0.9999

  x1 = fungsiRandom()
  x2 = fungsiRandom()

  is = [x1,x2]
  cS = is
  cC = cost(is)

  bSolution = is
  bCost = cC

  puts "TUGAS AI SIMULATED ANNEALING"
  puts "  Nama  : Alanu Dinasti Permana"
```

```
  puts "  NIM   : 130116-774"
  puts "  Kelas : IFIK - 40 - 03"
  puts "\nMenentukan nilai minimum dari sebuah fungsi"
  puts "menggunakan metode Simulated Annealing\n"

  puts "\nINISIALISASI AWAL"
  puts "  Temperature                    : #{temp}"
  puts "  Final Temperature              : #{final_temp}"
  puts "  Alpha                           : #{alpha}"
  puts "  X1 awal (Random)                : #{x1}"
  puts "  X2 awal (Random)                : #{x2}"
  puts "  Initial State                   : #{is}"
  puts "  State saat ini                  : #{cS}"
  puts "  f(X1,X2) awal                   : #{cC}"
  puts "  State terbaik awal              : #{bSolution}"
  puts "  Fungsi Minimum f(X1,X2) awal  : #{bCost}"

  bestTemp,bestState,bestCost =
  simulatedAnnealing(is,cS,cC,bSolution,bCost,temp,final_temp,alpha)

  puts "\nSETELAH ALGORITMA DIJALANKAN"
  puts "\nSOLUSI NUMERIC"
  puts "  Temperature          : #{bestTemp[bestTemp.length]}"
  puts "  State Minimum (X1,X2)    : #{bestState[bestState.length]}"
  puts "  Fungsi Minimum f(X1,X2)   : #{bestCost[bestCost.length]}"
  puts "\nSOLUSI EXACT f(0,0)        : [#{fungsi(xx1,xx2)}]"
end
```

2.2 Spesifikasi Hardware dan Software

Dalam memodelkan algoritma *Simulated Annealing* dibutuhkan spesifikasi yang besar dalam menemukan solusi paling optimum, alangkah baiknya jika menggunakan *High Peformance Computing* untuk momodelkanya. Dalam paper ini berikut hardware dan software yang digunakan.

- Hardware
  - MacBook Pro (13-inch, Mid 2012)
  - Processor 2,5 Ghz Intel Core i5
  - Ram 4GB 1600 MHz DDR3
  - Startup Disk SSD 128GB

- Software
  - Atom (*Editor*)
  - Ruby (*Program Language)*
  - Atom Ruuner (*Compiler*)

2.3 Hasil Running Program

P	State Minimum	Fungsi Minimum	Fungsi Exact	Waktu Running
1.	[0.007716399152421616, -0.033916158130843144]	- 2.1079979790440624	- 2.1746254627672363	31.812 detik
2.	[0.008682199086768705, -0.02704141517782155]	- 2.1224115699645227]		34.134 detik
3.	[0.01406017060305409, -0.02506495286862176]	- 2.127073322764901		36.112 detik
4.	[-0.021886824129108362, -0.036528764301596794]	- 2.0620950989751057		30.232 detik
5.	[0.0326226301265109, 0.018061372754834437]	- 2.127641206430553		32.133 detik
6.	[0.016475406261029946, -0.00934636865726901]	- 2.1502956477039814		35.157 detik
7.	[0.004966238175846627, 0.019476196217159014]	- 2.1363184098106283		31.689 detik
8.	[-0.0004554323230010482, -0.0074578754166427785]	- 2.15798229597994		32.769 detik
9.	[-0.006775427592899064, -0.005721393751132453]	- 2.1486509549467767		32.351 detik
n.	[0.0015990110889748359, -0.002685037610579144]	- 2.169439148626254		29.657 detik

Tabel solusi 2.3.1

Keterangan

P = Percobaan

```
TUGAS AI SIMULATED ANNEALING
Nama  : Alanu Dinasti Permana
NIM   : 130116-774
Kelas : IFIK - 40 - 03

Menentukan nilai minimum dari sebuah fungsi
menggunakan metode Simulated Annealing

INISIALISASI AWAL
Temperature       : 100
Final Temperature : 1.0e-05
Alpha             : 0.9999
X1 awal (Random)  : 7.5674432849513025
X2 awal (Random)  : 6.472108904018555
Initial state     : [7.5674432849513025, 6.472108904018555]

Solusi Exact f(0,0)   : -2.1746254627672363
Solusi Numeric
Temperature           : [0.0008097248793707022]
Best Solution (X1,X2) : [[0.0015990110889748359, -0.002685037610579144]]
Best Cost f(X1,X2)    : [-2.169439148636254]

Running: ruby (cwd=/Volumes/DATA/SEMESTER 6/Kecerdasan Buatan/TUGAS/RUBY/SA.rb
pid=2558)..... Exited with code=0 in 29.657 seconds.
```

Screenshoot hasil running 2.3.2

3. Kesimpulan

Kesimpulannya adalah metode *Simulated* Annealing sangat baik digunakan untuk menentukan nilai minimum dari sebuah fungsi. Semakin kecil nilai perbandingan *temperature* (**final_temp**) dan nilai dalam penentu iterasi *temperature* (**alpha**) maka semakin besar kemungkinan untuk mendapatkan nilai paling minimum. Tetapi membutuhkan membutuhkan spesifikasi memory yang tinggi. Berikut ilustrasinya.

Semakin kecil (Alpha dan final_temp) ➔ Solusi hampir mendekati, tetapi
- Jika memory besar ➔ kompleksitas waktu efisien
- Jike memory kecil ➔ kompeksitas waktu tidak efisien

Dalam paper ini banyak percobaan yang dilakukan mendapatkan fungsi minimum dengan nilai -2.169439148626254 yang mendekati nilai exact yaitu -2.1746254627672363. dengan rentan waktu kurang lebih 35 detik.

4. References

[1]Jason Brownlee (2016). *Clever Algorithms : Nature-Inspired Programming Recipes*
[2]Ikhdarisnan (2017). *Algoritma Optimasi Simulated Annealing*