Penggunaan Algoritma Genetika dan Graf Berbobot pada Perancangan Organisme Sederhana

Gabriel Bentara Raphael 13516119
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13516119@std.stei.itb.ac.id

***Abstrak*—Makalah ini membahas penggunaan algoritma genetika dan graf berbobot pada perancangan suatu organisme sederhana yang dapat berevolusi. Algoritma genetika merupakan salah satu algoritma evolusi yang berlandaskan prinsip yang sama dengan seleksi alam. Graf berbobot merupakan struktur data yang dapat merepresentasikan sebuah artificial neural network yang membuat organisme tersebut dapat berpikir sendiri atau seringkali kita dengar sebagai inteligensia buatan. Dengan mengetahui prinsip dari algoritma genetika dan graf berbobot sebagai neural network, kita dapat menyelesaikan masalah dalam kehidupan yang sangat sulit untuk mencari jawaban pastinya dengan algoritma biasa karena kompleksitasnya yang eksponensial.**

***Kata kunci*—algoritma genetika, algoritma evolusi, graf berbobot, seleksi alam**

I. PENDAHULUAN

Dewasa ini, seringkali kita menjumpai bahwa terdapat banyak sekali masalah yang tidak dapat kita selesaikan menggunakan algoritma-algoritma biasa karena kompleksitasnya yang eksponensial. Padahal, masalah-masalah ini sangatlah mempengaruhi kehidupan kita sehari-hari. Contohnya adalah *Traveling Salesman Problem* dan *Chineese Postman Problem.*

Kedua masalah ini selalu kita jumpai dalam kehidupan kita, misalnya ketika kita berpergian atau menerima paket dari kota lain. Oleh karena itu, kita memerlukan algoritma lain yang dapat menghasilkan solusi efisien. Di sinilah algoritma evolusi berperan, salah satunya adalah algoritma genetika yang akan kita bahas.

Prinsip dari algoritma genetika adalah seleksi alam di mana yang kuatlah yang bertahan. Kita memandang solusi sebagai sebuah individu. Terdapat 3 inti proses sederhana dalam proses algoritma ini berjalan, yaitu seleksi, kawin silang, dan mutasi. Seleksi akan memilih para elit dengan nilai terbaik, kawin silang akan menyilangkan dua elit untuk membentuk suatu individu baru, dan mutasi akan mengubah sedikit hal dari individu yang ada.

Dengan algoritma genetika, kita akan mendapatkan solusi yang cukup efisien atau paling efisien. Efisiensi solusi ini dapat diatur dari berapa lama kita "melatih" solusi kita dan kompleksitas struktur data yang kita miliki.

Untuk mempermudah penjelasan penggunaan algoritma ini, penulis memilih untuk merancang suatu organisme buatan sederhana, agar pembaca dapat lebih menangkap apa yang dimaksud dengan seleksi alam dalam algoritma genetika.

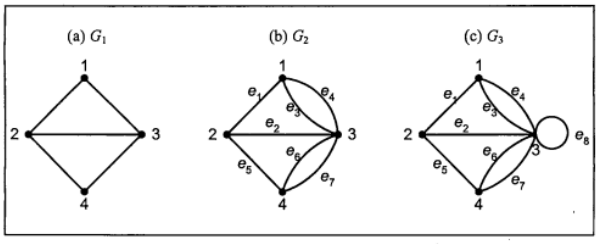
II. LANDASAN TEORI

2.1 Definisi Graf

Secara matematis, sebuah graf G didefinisikan sebagai pasangan himpunan (V,E) dalam notasi G = (V,E) di mana V himpunan tidak kosong dari simpul (*vertices/node*) dan E adalah himpunan sisi (*arcs/edges*) yang menghubungkan sepasang *vertices* [1].

Graf dapat memiliki himpunan sisi kosong, namun sebuah graf harus memiliki setidaknya satu buah simpul. Bila sebuah graf hanya memiliki satu buah simpul tanpa ada sisi, maka graf tersebut dinamakan *graf trivial*.

Cara umum dalam menggambarkan sebuah graf adalah dengan menggambarkan titik-titik sebagai *vertex* dan garis yang menghubungkan dua *vertex* sebagai *edge*. Dalam penggambaran graf, *vertex* dan *edge* yang ada biasanya diberi nama untuk memudahkan pembacaan.



Gambar 2.1 (a) Graf Sederhana (b) Graf Ganda (c) Graf Semu[1]

2.2 Jenis Graf

Graf dikategorikan menjadi beberapa jenis tergantung pada sudut pandang pengkategorian tersebut. Jika kita mengkategorikan sebuah graf berdasarkan ada tidaknya sisi ganda atau gelang, maka terdapat 2 jenis graf yaitu :

1. Graf Sederhana
   Graf sederhana tidak memiliki gelang ataupun sisi ganda. Contoh graf sederhana dapat dilihat pada gambar 2.1 (a) di atas.
2. Graf Tak Sederhana
   Graf tidak sederhana adalah graf yang memiliki sisi ganda atau gelang. Graf tak-sederhana kemudian dibagi menjadi dua jenis yaitu graf ganda (memiliki sisi ganda) dan graf semu (memiliki gelang). Gelang didefinisikan

sebagai sebuah sisi yang menghubungkan sebuah simpul dengan dirinya sendiri. Contoh graf tak-sederhana dapat dilihat pada gambar 2.1 (a) dan (b).

Jika kita mengkategorikan sebuah graf berdasarkan arahnya, maka terdapat 2 jenis graf yaitu :

1. Graf tak-berarah
   Graf tak-berarah sisinya tidak memiliki orientasi arah yang berarti urutan dari pasangan simpul yang dihubungkan oleh sebuah sisi tidak diperhatikan. Sisi yang menghubungkan pasangan simpul (u,v) dan (v,u) adalah sama. Contoh aplikasi dari graf tak-berarah adalah jaringan telepon, dimana hubungan dari dua penelpon tidak diperhatikan.
2. Graf berarah
   Graf berarah sisinya memiliki orientasi arah yang berarti sisi yang menghubungkan pasangan simpul (u,v) belum tentu menghubungkan pasangan simpul (v,u) sehingga (u,v) tidak sama dengan (v,u). Pada pasangan simpul (u,v) pada graf berarah G, u disebut sebagai simpul asal, dan v disebut sebagai simpul terminal.
   Pada graf berarah

Jenis-jenis graf tersebut kemudian dapat diberi bobot pada masing-masing sisinya sehingga menjadi graf berbobot. Bobot tiap sisi dapat berbeda, tergantung pada permasalahan yang ingin digambarkan. Misalnya bobot dapat berupa jarak antara dua kota.

2.3 Terminologi Graf

Dalam graf, terdapat beberapa terminologi dasar yang harus ktia ketahui. Berikut ini adalah beberapa terminologi dasar yang umum :

1. Bertetangga (*Adjacent*)
   Dua simpul dari graf tidak berarah bertetangga jika ada sebuah sisi yang menghubungkan keduanya. Dalam notasinya, u bertetangga dengan v jika ada (u,v) sebagai sisi dari sebuah graf.
   Pada graf berarah, u dikatakan bertetangga jika ada (u,v) sebagai sisi (busur) dari graf tersebut dan v dikatakan sebagai tetangga dari u.

2. Bersisian (*Incident*)
   Untuk setiap sisi e = (u,v) dalam graf G, e dikatakan bersisian dengan u dan v.

3. Simpul Terpencil
   Sebuah simpul dikatakan sebagai simpul terpencil apabila simpul tersebut tidak memiliki sisi dan tidak bertetangga dengan simpul lainnya.

4. Graf Kosong (*null graf*)
   Graf kosong adalah graf yang himpunan sisinya merupakan himpunan kosong.

5. Derajat (*Degree*)
   Derajat dari suatu simpul adalah banyaknya jumlah sisi yang bersisian dengan simpul tersebut. Derajat dinotasikan sebagai *d(v)*. Pada graf berarah, derajat dibagi menjadi dua, yaitu derajat keluar dan derajat masuk. Derajat keluar merupakan jumlah sisi yang bersisian dengan sebuah simpul dan arahnya mengarah ke dalam simpul tersebut, sedangkan derajat keluar adalah jumlah sisi yang bersisian dengan sebuah simpul, namun mengarah ke tetangga dari simpul tersebut.
   Derajat masuk dinyatakan sebagai $d_{in}(v)$ dan derajat keluar dinyatakan sebagai $d_{out}(v)$. Derajat total dinyatakan sebagai $d(v) = d_{in}(v) + d_{out}(v)$.

6. Lintasan (*Path*)
   Lintasan dengan panjang n yang berasal dari v_0 ke v_n adalah sebuah barisan selang-seling antara simpul dan sisi yang menghubungkan simpul sebelumnya dengan simpul setelahnya. Lintasan berbentuk barisan seperti v_0, e_1, v_1, e_2, v_2, ….v_{n-1}, e_n, v_n.
   Terdapat pengecualian dalam penulisan lintasan pada graf sederhana. Karena hanya terdapat 1 sisi yang menghubungkan sepasang simpul, kita cukup menuliskan lintasannya sebagai barisan dari simpul saja. Misalnya 1,2,4,3 adalah sebuah lintasan yang sama dengan 1, (1,2), 2, (2,4), 4, (4,3), 3.
   Lintasan pada graf dapat diklasifikasikan menjadi beberapa jenis. Berdasarkan keunikan simpul yang dilewati, sebuah lintasan disebut lintasan sederhana jika lintasan tersebut hanya melewati sebuah simpul tepat satu kali saja. Berdasarkan simpul awal dan akhir, sebuah lintasan dikatakan sebagai lintasan tertutup jika simpul awal dan akhirnya adalah sama, jika tidak demikian maka lintasan tersebut dikatakan sebagai lintasan terbuka.

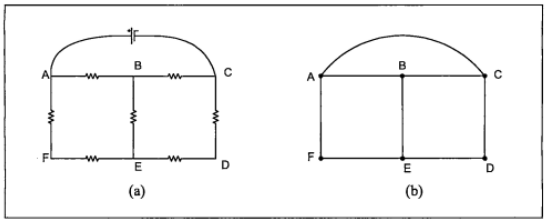
7. Terhubung (*connected*)
   Sepasang simpul (u,v) dikatakan terhubung apabila terdapat lintasan yang berawalan dari u dan berakhir di v. Jika setiap pasang simpul dari graf G terhubung, maka graf G disebut sebagai graf terhubung.

2.4 Aplikasi Graf

Aplikasi graf pada dunia nyata sangatlah luas. Graf dapat digunakan pada banyak hal dalam kehidupan sehari-hari, baik dalam cabang ilmu komputer ataupun disiplin ilmu lainnya. Berikut ini adalah beberapa contoh penerapan graf dalam kehidupan sehari-hari.

2.4.1 Rangkaian listrik

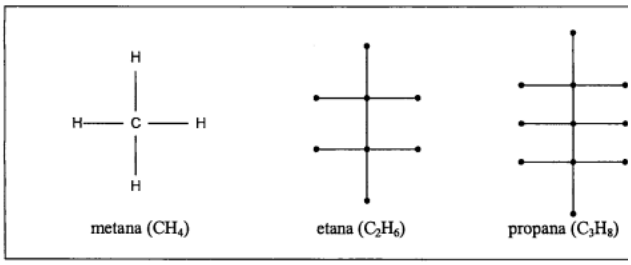
Graf digunakan oleh Kirfchoff untuk memodelkan rangkaian listrik. Berdasarkan graf tersebut, Kirchoff menurunkan persamaan arus listrik yang keluar masuk dari tiap simpul.



Gambar 2.2 (a) Rangkaian listrik (b) graf yang menyatakan rangkaian listrik [1]

2.4.2 Isomer senyawa kimia karbon

Graf juga digunakan untuk memodelkan molekul senyawa alkana dengan rumus kimia C_nH_{2n+2} untuk menghitung jumlah isomer. Atom-atom dalam alkana tersebut (Carbon (C) dan Hidrogen (H)) dinyatakan sebagai sebuah simpul, sedangkan ikatan antara atom C dan H dinyatakan sebagai sebuah sisi. (Gambar 2.3)



Gambar 2.3 Graf Senyawa Alkana [1]

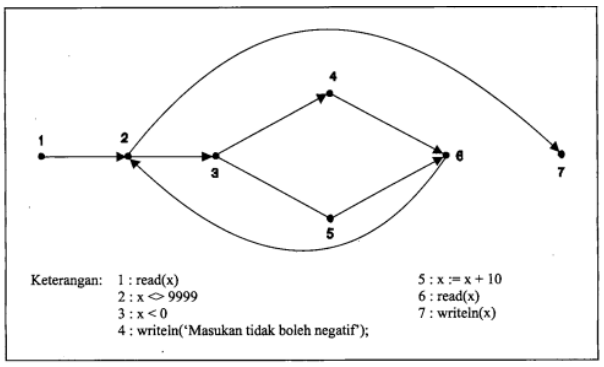
2.4.3 *Flowchart* Program

Alur bagaimana sebuah program berjalan dapat digambarkan sebagai sebuah graf berarah. Sebuah *flowchart* digambarkan dengan tujuan memperjelas tingkah laku sebuah program dan tahapan-tahapan yang dilakukan sehingga mempermudah dalam proses pencarian kesalahan atau *bug*.

Contoh program dengan *flowchart* nya :

```
Read(x);
While x <> 9999 do
Begin
  If x < 0 then
        Writeln('Masukan tidak boleh negatif')
  Else
   X := x + 10;
  Read(x);
End;
Writeln(x);
```

Gambar 2.4 Contoh program pascal [1]

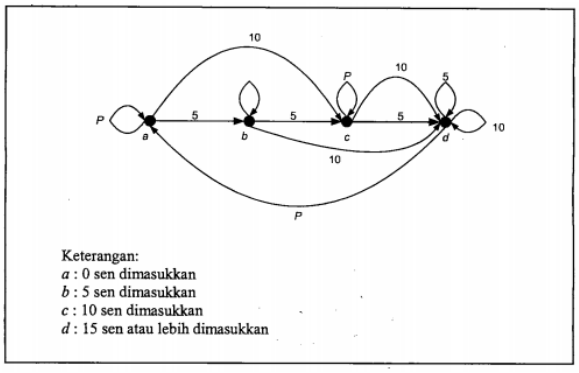


Gambar 2.5 *Flowchart* program pascal 2.4[1]

2.4.4 Graf dalam Teori Otomata

Teori otomata digunakan dalam pemodelan sebuah mesin dengan status terhingga (*Finite State Automata*). Graf digunakan untuk menggambarkan status dan perubahannya. Status digambarkan sebagai sebuah simpul dan perubahan status digambarkan sebagai sebuah sisi yang memiliki bobot. Bobot pada FSM adalah input yang menyebabkan perubahan status pada FSM. Contoh sederhana dari FSM adalah mesin jaja (*vending machine*). Perilaku mesin jaja dapat digambarkan sebagai sebuah FSM.

Misalkan mesin jaja tersebut menjual sebuah coklat dengan harga 15 sen. Terdapat 3 jenis koin, yaitu 5 sen, 10 sen, dan 15 sen. Mesin tidak mengembalikan uang yang tersisa. Maka FSM yang terbentuk jika digambarkan dengan graf akan terlihat sebagai berikut.

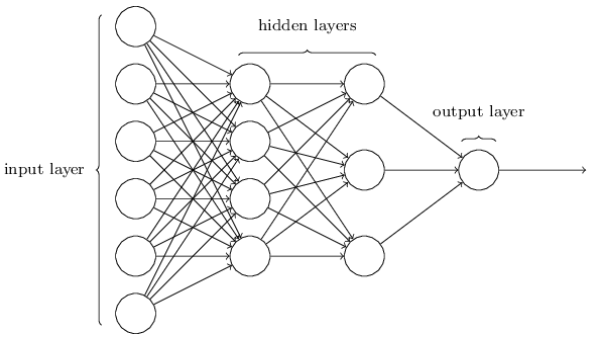


Gambar 2.6 FSM Mesin Jaja [1]

2.4.5 Artificial Neural Network

Artificial Neural Network (Jaringan saraf buatan) merupakan salah satu sistem koneksi komputer yang terdiri atas beberapa elemen proses yang terkoneksi erat, yang memproses informasi berdasarkan status dinamis yang berubah dari input eksternal. [2]

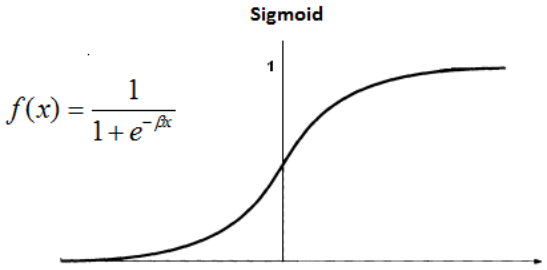
Jaringan saraf buatan dapat digambarkan sebagai sebuah graf 1 arah yang memiliki bobot. Setiap simpul menyimpan sebuah nilai yang disebut *Activation value* (nilai aktivasi). Sedangkan setiap bobot dari sisi merupakan faktor pengali. Bobot biasanya bernilai diantara -1 hingga 1.



Gambar 2.7 Contoh Neural Network. (sumber : http://neuralnetworksanddeeplearning.com/chap1.html)

Dalam sebuah jaringan saraf buatan, terdapat beberapa lapisan (*layer*). Pertama adalah lapisan masukan (*input layer*). Masing-masing simpul dari lapisan masukan memiliki nilai berupa bilangan real antara 0 sampai 1. Nilai ini didapatkan dari masukan user.

Lapisan kedua adalah lapisan tersembunyi (*hidden layer*).Setiap simpul pada lapisan ini menerima masukan dari setiap simpul pada lapisan sebelumnya dikalikan dengan bobot pada sisi yang menghubungkan antara kedua simpul tersebut kemudian dijumlahkan. Agar nilai tetap berada di antara 0 sampai 1, sebuah fungsi lain digunakan. Salah satu fungsi yang paling sering digunakan adalah fungsi sigmoid.



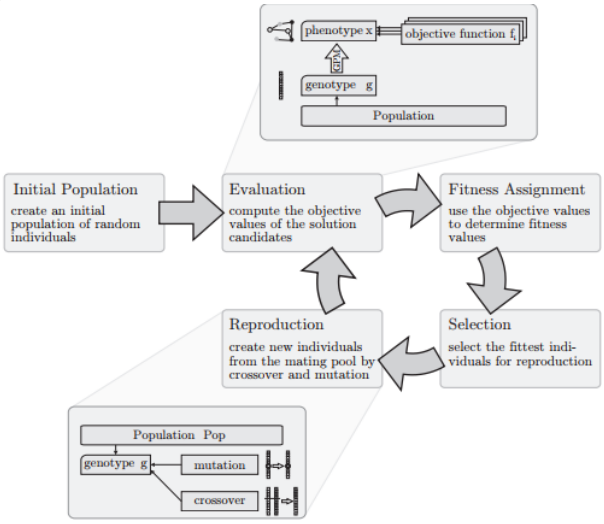
Gambar 2.8 Grafik Fungsi Sigmoid (sumber : https://www.linkedin.com/pulse/logistic-regression-sigmoid-function-explained-plain-english-hsu)

Lapisan tersembunyi ini dapat terdiri atas 1 atau lebih lapisan. Lapisan ini dikatakan tersembunyi karena kita tidak mengetahui apa yang terjadi pada lapisan tersebut dan fungsi dari masing-masing simpul.

Lapisan terakhir adalah lapisan keluaran *(output layer*). Lapisan ini menerima input sama seperti lapisan tersembunyi. Nilai setiap simpul dari lapisan ini menggambarkan keluaran yang dihasilkan. Nilai ini juga bernilai 0-1, semakin tinggi nilai tersebut, semakin tinggi juga probabilitas bahwa output tersebut benar (jika kita ingin mengkategorikan sesuatu).

2.5 Algoritma Genetika

Algoritma Genetika (*Genetic Algorithm*) adalah sub-kelas dari algoritma evolusi di mana element dari luas pencarian G adalah string biner atau array dari element bertipe standar lainnya. Berikut adalah cara kerja algoritma genetika



Gambar 2.9 Proses Algoritma Genetika [3]

Algoritma genetika bekerja dalam 5 langkah spesifik, yaitu :

1. Pembuatan Populasi
   Populasi diisi dengan individiu-individu secara acak. Di sini individu dapat berupa *neural network* ataupun solusi dalam bentuk string atau array linear.

2. Evaluasi
   Setiap individu diberikan nilai berdasarkan performanya terhadap aturan-aturan yang kita definisikan sebelumnya.Semakin tinggi nilai tersebut, semakin besar kemungkinan individu tersebut bertahan.

3. Pemberian Nilai
   Saat proses evaluasi berlangsung, kita dapat memberikan masing-masing nilai performa berdasarkan tindakan apa saja yang telah ia lakukan.

4. Seleksi
   Pada proses ini, individu dengan nilai performa terbaik akan dipertahankan dan sebagian besar sisa individu akan dibuang kecuali beberapa individu yang beruntung berdasarkan probabilitas yang ada.

5. Reproduksi
   Pada proses ini, individu-individu unggulan akan melakukan reproduksi. Individu baru akan dihasilkan dengan gen campuran antara individu asalnya.

Selain 5 proses di atas, terdapat proses mutasi yang dapat menghasilkan perubahan acak terhadap sebuah individu, tentunya dengan probabilitas yang telah ditentukan.

III. METODOLOGI

3.1 Batasan Penelitian

Terdapat beberapa batasan dalam perancangan organisme sederhana dengan menggunakan algoritma genetika dan graf berbobot, diantaranya adalah :

1. Organisme tidak merepresentasikan solusi yang paling efisien, namun hanya berupa salah satu solusi yang cukup efisien untuk kondisi yang penulis tentukan.
2. Organisme sederhana yang dibuat mengabaikan organisme lainnya pada ekosistem yang sama.
3. Organisme hanya dapat melihat makanan dan batasan wilayah pada arah tertentu, sisanya diabaikan.

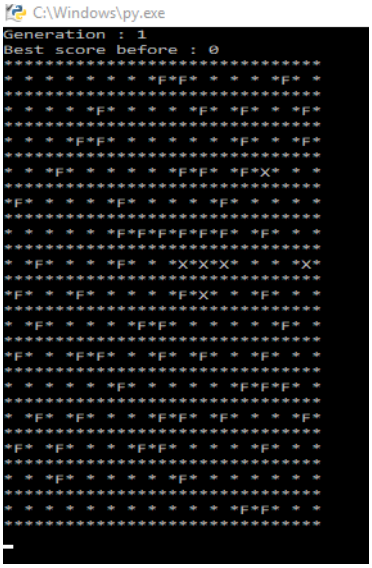
3.2 Langkah-Langkah Pembuatan Organisme Sederhana

Dalam pembuatan organisme sederhana ini, penulis memecah-mecah persoalan dan menyusunnya secara bertahap agar sebuah ekosistem dapat berbentuk. Berikut ini adalah tahapan yang penulis lakukan dalam pembuatan organisme sederhana ini.

1. Perancangan Ekosistem

   Penulis merancang ekosistem pengujian sebagai sebuah matriks NxN di mana sebuah elemen (sel) memiliki tepat satu buah makanan, organisme, atau tidak keduanya. Ketika salah satu organisme berada di kotak yang sama dengan makanan, maka makanan tersebut "dimakan" oleh organisme tersebut dan diacak kembali di kotak lainnya.

   Ketika sebuah organisme memakan sebuah makanan, maka organisme tersebut akan memperoleh satu poin performa. Namun jika organisme tersebut "keluar" dari ekosistem tersebut dengan cara menabrak batasan, maka organisme tersebut dianggap mati dan nilai performanya menjadi 0.



Gambar 3.1 Contoh Ekosistem, F melambangkan makanan dan X melambangkan organisme.

2. Perancangan Organisme

   Penulis merancang organisme ini agar memiliki 8 input, yaitu jarak antara organisme tersebut ke makanan dan dinding dalam 4 arah yang berbeda. Hasilnya kemudian dibagi dengan ukuran matriks ekosistem yang telah ditentukan sebelumnya. Organisme memiliki 4 tindakan yang dapat dilakukan yaitu bergerak ke kiri, kanan, atas, atau bawah.

   Setiap organisme memiliki beberapa properti yaitu NN (Neural Network), posisi baris (rowPos), posisi kolom, (colPos), status mati, nilai performa (fit), dan nyawa (health).

```
class Organism():
    def __init__(self,rowPos, colPos, NN = NN()):
        self.NN = NN
        self.rowPos = rowPos
        self.colPos = colPos
        self.dead = False
        self.fit = 0
        self.health = 10
```

Gambar 3.2 Definisi Organisme dengan Paradigma Objek dalam Bahasa Python 3 oleh Penulis

Setiap organisme dapat dibuat secara acak atau sebagai hasil kawin silang dari dua organisme lainnya. Setiap organisme juga memiliki probabilitas sebesar 5% untuk mengalami mutasi terhadap neural network yang dimiliki.

Neural network yang dimiliki setiap organisme berstruktur sama yaitu sebuah graf yang memiliki sebuah lapisan masukan yang terdiri atas 8 simpul, satu lapisan tersembunyi yang teridiri atas 6 simpul, dan lapisan keluaran yang terdiri atas 4 simpul, masing-masing melambangkan arah gerak organisme tersebut selanjutnya.

Penulis mendefinisikan struktur graf neural network tersebut dalam dua array multi dimensi. Array pertama merupakan array 2 dimensi dengan elemennya misalkan a_{ij} yang berarti node ke j dari lapisan ke i.

```
>>> nodes = [[1,2,3,4,5,6,7,8], [9,10,11,12,13,14], [15
,16,17,18,19,20], [21,22,23,24]]
>>> nodes[0][0]
1
```

Gambar 3.3 Struktur Array 2 Dimensi untuk Simpul

Sedangkan untuk sisi antara sepasang simpul, penulis tidak mendefinisikannya secara eksplisit dikarenakan sifat dari neural network yang memiliki bentuk yang pasti yaitu setiap simpul dari lapisan n-1 pasti memiliki satu sisi ke masing-masing simpul dari lapisan ke n. Oleh karena itu, penulis hanya mendefinisikan bobot dari masing-masing sisi dengan memasukkan masing-masing bobot ke dalam array 3 dimensi. Misalkan A_{ijk} adalah elemen dari array tersebut maka A_{ijk} adalah bobot untuk sisi yang menghubungkan simpul ke j dari lapisan ke-i kepada simpul kek dari lapisan ke-i+1.

Simpul kemudian dibungkus dalam objek untuk mencegah adanya nilai melebihi 1 atau kurang dari 0. Neural network yang ada juga didefinisikan sebagai objek

yang memiliki properti kedua array simpul dan bobot serta banyaknya lapisan yang ada.

```
class Node():
    def __init__(self, value = 0):
        self._value = value

    @property
    def value(self):
        return self._value

    @value.setter
    def value(self, value):
        if (value < -1 or value > 1):
            raise Exception('Value must be in inclusive range 0 - 1')
        self._value = value


class NN():
    def __init__(self, layerCount = 0, layers = [], weights = []):
        self.layerCount = layerCount
        self.layers = layers
        self.weights = weights
```

Gambar 3.4 Implementasi Simpul dan Graf Neural Network dalam Bahasa Python 3

3. Pelatihan Organisme dalam Ekosistem

Pelatihan dilakukan dengan aturan-aturan yang penulis sudah tentukan sebelumnya yaitu :

1. Organisme memperoleh satu poin ketika memakan makanan dengan cara menghampiri sel dimana makanan tersebut berada.
2. Organisme mati ketika ia keluar dari batasan ekosistem dan nilai performanya otomatis menjadi 0.
3. Organisme dapat bergerak sejauh 200 sel sebelum satu sesi (generasi) selesai.
4. 50% organisme terbaik akan lolos ke generasi selanjutnya, 50% sisanya akan terdiri atas campuran organisme hasil kawin silang dari organisme yang baik dan organisme acak baru.
5. Terdapat 5% kemungkinan terjadi mutasi pada setiap sesi untuk masing-masing organisme.

Ekosistem yang penulis pilih berukuran 15x15 dengan jumlah makanan 60 dan organisme 20. Pelatihan dilakukan selama 5000 generasi di mana pada masing-masing generasi apabila terdapat organisme yang mendapat nilai performa lebih dari 20 maka organisme tersebut akan disimpan ke dalam file untuk diobservasi lebih lanjut.

IV. Hasil dan Pembahasan

Dari 5000 generasi yang telah penulis lakukan, terdapat 887 organisme unggul yang dapat mencapai skor lebih dari 20 dalam sebuah generasi. Penulis kemudian mengamati tingkah laku yang dihasilkan oleh tiap organisme dan hasilnya cukup bervariasi.

1. Organisme beruntung
   Organisme ini hanya bergerak satu petak ke kiri dan kanan atau atas bawah secara beruntun. Namun karena makanan secara acak berada di sel tersebut, organisme ini dapat mencapai nilai performa hingga di atas 20.

2. Organisme dengan gerak horizontal
   Organisme ini bertahan hidup dengan cara bergerak ke arah makanan secara horizontal. Beberapa organisme selamat dengan cara ini pada generasi-generasi awal dikarenakan ketika sebuah makanan dimakan oleh organisme lain, terdapat kemungkinan bahwa makanan tersebut akan ditaruh secara acak di barisan yang sama seperti organisme yang bergerak horizontal, sehingga organisme tersebut dapat mencapai nilai performa di atas 20.

3. Organisme dengan gerak berliku menurun
   Organisme ini sedikit lebih pintar dari organisme sebelumnya. Organisme ini pertama melihat makanan pada baris yang sama. Apabila organisme ini menjumpai makanan, ia akan bergerak memakannya hingga seluruh makanan pada baris itu habis. Setelah itu, organisme akan bergerak turun ke baris berikutnya dan melakukan hal yang sama.

4. Organisme dengan gerak berliku menyamping
   Sedikit berbeda dengan organisme gerak berliku menurun, organisme ini pertama mencari makan secara vertikal. Ketika organisme ini sudah memakan seluruh makanan hingga habis, ia akan bergerak ke kanan dan mencari makanan kembali dengan cara yang sama. Namun dalam beberapa kasus, terdapat saat-saat di mana organisme tersebut dapat bergerak ke kiri juga.

Organisme yang dihasilkan oleh algoritma ini berhasil bertahan untuk 1 generasi sesuai dengan aturan-aturan yang ada. Organisme-organisme yang dihasilkan tidaklah sempurna karena kurangnya opsi arah yang ada, sehingga ada kasus-kasus di mana organisme tidak bergerak melainkan terjebak di antara dua sel secara terus menerus. Organisme yang dihasilkan juga terlihat lebih “manusiawi” jika dibandingkan dengan organisme yang di *hardcode*. Organisme ini dapat dikatakan menyerupai organisme hidup sesungguhnya. Berikut adalah salah satu contoh array bobot hasil latihan salah satu organisme gerak berliku menyamping.

```
Weight Layer 0
    Node 0 : -0.0639 0.1070 -0.9159 -0.8468 -
  0.8862 -0.2605
    Node 1 : -0.4896 -0.0961 -0.5768 -0.3841
  0.6375 0.3164
    Node 2 : -0.1301 -0.0357 0.1377 -0.7577
  0.8175 0.4325
    Node 3 : 0.8365 -0.2024 0.2852 0.7801 -0.9065
  -0.0448
    Node 4 : 0.0373 -0.0139 0.8416 -0.2321 0.9672
  -0.7089
    Node 5 : 0.3017 0.3216 -0.3160 -0.2976 -
  0.9741 -0.4025
    Node 6 : -0.1156 0.0662 0.7247 -0.0181 0.2254
  -0.7873
    Node 7 : -0.4131 0.7452 -0.9058 -0.7840
  -0.9929 0.2050
```

```
Weight Layer 1
    Node 0 : 0.0072 -0.3546 0.0399 0.8164
    Node 1 : 0.6580 -0.1672 -0.1124 -0.3863
    Node 2 : 0.3334 0.5016 -0.9269 0.2004
    Node 3 : 0.2179 0.6697 0.4520 0.8403
   Node 4 : -0.7314 0.4414 -0.6011 -0.0058
   Node 5 : 0.1429 0.4107 -0.5466 -0.2195
```

Dimana masing-masing nilai dibaca menjadi "Bobot dari *Node* (simpul) X di *layer* (lapisan) Y ke pada *Node* Z di *layer* Y+1" dimana Z adalah posisi nilai real (bobot) di masing-masing *node* (nilai dibulatkan menjadi 4 dibelakang koma untuk memudahkan pembacaan).

VI. SIMPULAN

Dari hasil eksperimen pada bab sebelumnya, dapat disimpulkan bahwa graf berbobot dapat digunakan sebagai struktur data *neural network* suatu organisme sederhana yang dapat berevolusi hanya dengan mengubah konfigurasi bobot yang sesuai menggunakan algoritma genetika.

VI. UCAPAN TERIMAKASIH

Penulis ingin mengucapkan terimakasih pada semua pihak yang telah membantu penulis dalam membuat makalah berjudul "Penggunaan Algoritma Genetika dan Graf Berbobot pada Perancangan Organisme Sederhana". Tidak lupa, penulis juga berterimakasih kepada Pak Rinaldi Munir dan Bu Harlili selaku pembimbing mata kuliah Matematika Diskrit.

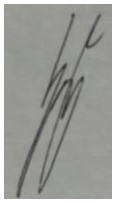
REFERENCES

[1] Munir, Rinaldi. Matematika Diskrit Edisi 3, Penerbit Informatika, 2010.
[2] Maureen Caudill , "Neural Network Primer: Part I", AI Expert, 1989
[3] http://www.it-weise.de/projects/book.pdf diakses pada tanggal 2 Desember 2017 pukul 13.00 WIB

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 2 Desember 2017



Gabriel Bentara Raphael
13516119