Implementasi Metode Breadth First Search danVikor pada Aplikasi Diagnosa Kerusakan Hardware Komputer

Halim Agung
Faculty of Technology and Design
Bunda Mulia University
Jl. Lodan Raya No. 2 Jakarta 14430, Indonesia
hagung@bundamulia.ac.id

Nico Yunus Marselinus
Faculty of Technology and Design
Bunda Mulia University
Jl. Lodan Raya No. 2 Jakarta 14430, Indonesia

Abstrak- Kerusakan pada komputer memiliki berbagai macam kerusakan dari kerusakan ringan maupun berat yang menyebabkan banyak *service center* yang banyak dibuka. Akan tetapi seiring banyaknya *service center* yang dibuka, banyak pula *service center* yang melakukan penipuan pada pelanggan. Untuk mengatasi hal tersebut dibangunlah aplikasi yang digunakan dalam pencarian diagnosa kerusakan komputer yang menggunakan metode *breadth first search* dan vikor. Pengujian yang dilakukan adalah dengan melakukan uji coba aplikasi dengan menggunakan metode *black box*. Dari hasil pengujian yang dilakukan dapat diambil kesimpulan bahwa algoritma *Breadth First Search* dapat melakukan proses pencarian solusi melalui pemilihan gejala-gejala yang ada, kemudian metode Vikor dapat melakukan proses rating untuk merekomendasi solusi yang akan dilakukan oleh *user* dengan tingkat keberhasilan 99% sesuai dengan rekomendasi ahli.

Kata Kunci: aplikasi diagnosa, breadth first search, hardware komputer, vikor.

I. INTRODUCTION

Komputer merupakan teknologi mutakhir yang digunakan banyak orang untuk melakukan banyak kegiatan. Komputer terbagi dalam tiga komponen yaitu *hardware*, *software*, dan *brainware*. Pada komputer sering terjadi kerusakan pada *software* dan *hardware*, pada kerusakan ini biasa banyak pengguna atau *brainware* yang tidak mengerti solusi dari kerusakan tersebut. Biasanya penggunakan pihak ketiga yang lebih ahli dalam persoalan tersebut dan membayar mahal untuk memperbaiki kerusakan itu.

Di dalam penggunaannya, *hardware* komputer tidak luput dari kerusakan atau masalah, meskipun kerusakan itu adalah kerusakan kecil dan peranan seorang teknisi pun sangat di butuhkan terutama bagi para pengguna atau pemilik komputer yang tidak mengetahui penyebab–penyebab kerusakan dan cara memperbaiki di saat komputer mengalami kerusakan. Sangat disayangkan jika kerusakan yang terjadi hanyalah kerusakan kecil yang semestinya dapat di perbaiki sendiri. Sementara, waktu untuk menunggu perbaikan cukup lama dan membutuhkan biaya yang cukup besar..

Aplikasi ini akan menentukan solusi terbaik dari beberapa solusi yang diberikan untuk dilakukan *user*, dimana setiap solusi akan diberikan *rating*. Dalam proses *rating* tersebut aplikasi ini menggunakan metode *Breadth First Search* dan metode VIKOR. VIKOR (*VlseKriterijumska Optimizacija I Kompromisno Resenje* dalam bahasa Serbia, yang artinya *Multicriteria Optimization and Compromise Solution*) adalah metode *ranking* dengan menggunakan indeks peringkat multikriteria berdasarkan ukuran tertentu dari kedekatan dengan solusi yang ideal. Metode VIKOR merupakan salah satu metode yang dapat dikategorisasikan dalam *Multicriteria Decision Analysis*. Metode VIKOR dikembangkan sebagai metode *multicriteria decision making* untuk menyelesaikan pengambilan keputusan bersifat diskrit pada kriteria yang bertentangan dan non-*commensurable*(Opricovic and Tzeng 2007).VIKOR adalah sebuah metode untuk optimisasi/optimalisasi kriteria majemuk dalam suatu sistem yang kompleks (Khezrian, Wan Kadir et al., 2011).Konsep dasar VIKOR adalah menentukan *ranking* dari sampel-sampel yang ada dengan melihat hasil dari nilai-nilai sesalan atau *regrets* (R) dari setiap sampel. Metode VIKOR telah digunakan oleh beberapa peneliti dalam MCDM, seperti dalam pemilihan vendor (Datta, Mahapatra et al., 2010).

Sedangkan pencarian dengan *Breadth First Search* menggunakan teknik di mana langkah pertamanya adalah *root node* diekspansi. Setelah itu dilanjutkan semua *successor* dari *root node* juga diperluas. Hal ini terus dilakukan berulang-ulang hingga node pada *level* paling bawah yang sudah tidak mempunyai *successor* lagi (Budiharto dkk,2014). Keuntungan dari metode ini tidak menemui jalan buntu dalam pencarian, jika ada satu solusi maka *Breadth First Search* akan menemukannya, dan jika ada lebih dari satu solusi, metode *Breadth First Search* akan menemukan solusi

minimum (Budiharto dkk,2014). Kekurangan metode ini memerlukan memori yang cukup banyak dan membutuhkan waktu yang cukup lama karena metode ini mencari dari node ke node atau dari pohon ke pohon (Budiharto dkk,2014).

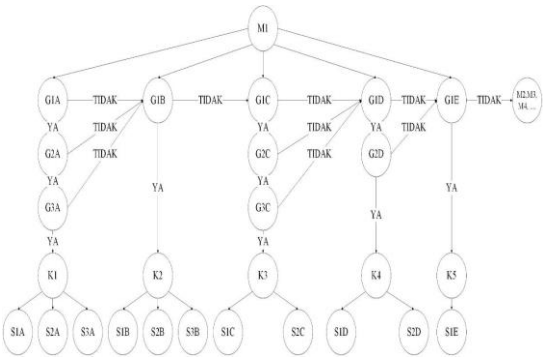
Algoritma BFS menggunakan algoritma tunggal yang memanggil setiap node pada pohon pencarian. Algoritma BFS melakukan *looping* dari setiap node ke node yang lain sampai tidak menemukan titik yang belum di kunjungi (C.E. Leiserson and T.B. Schardl, 2010).

II. METHODOLOGY

Dalam penelitian ini metode yang akan digunakan adalah metode *Breadth First Search* dan Vikor. Metode *Breadth First Search* adalah suatu metode pencarian yang mencari pada setiap pohon menuju akarnya sampai akar-akar pada setiap pohon itu habis, dengan kata lain tidak memiliki akar lagi.

Metode Vikor adalah metode peringkatan dengan menggunakan indeks peringkat multikriteria berdasarkan ukuran tertentu dari kedekatan dengan solusi yang ideal. Konsep dasar Vikor adalah menentukan ranking dari sampel-sampel yang ada dengan melihat hasil dari nilai-nilai sesalan atau *regrets* (R) dari setiap sampel.

Berikut ini adalah contoh pencarian kesimpulan pada sistem pakar *hardware* komputer dengan metode *breadth first search*. Proses pencarian dilakukan pada satu jenis masalah "komputer mati total" dan *user* memilih gejala yang ditampilkan bedasarkan masalah tersebut dengan penjelasan sebagai berikut:



Gambar 1 Pohon Masalah

Dilihat dari Gambar Pohon Masalah diambil contoh kasus, *user* memilih "M1" sebagai masalah utama yang menjadi masalah pertama pada pohon masalah.

- Ketika "M1" dipilih akan keluar "G1A" sebagai kepala pohon gejala, di sini *user* dapat memilih "ya" atau "tidak".
- User memilih "ya" maka akan muncul "G2A", di sini *user* dapat memilih lagi antara "ya atau "tidak".
- User memilih "tidak" dan akan muncul "G1B" sebagai kepala pohon gejala yang kedua.
- Kemudian *user* memilih "ya" dan akan muncul "K2" yaitu kesimpulan dari "G1B".
- Karena *user* mendapat "K2" maka akan muncul juga "S1B", "S2B", dan "S3C" sebagai solusi dari "K2".

Penjelasan sebagai berikut:

- M1, M2, M3, M4, dan M... adalah kepala masalah yang akan menampilkan kepala pohon gejala.
- G1A, G1B, G1C, dan G1... adalah kepala pohon gejala yang akan menampilkan akar gejala dari kepala pohon gejala.
- G2A, G3A, G2B, G3B, dan G... adalah akar dari kepala pohon gejala masing-masing.
- K1, K2, K3, dan K... adalah kesimpulan yang diambil berdasarkan kepala gejala, dan akar gejala yang dipilih.
- S1A, S2A, S3A, S1B, S2B, dan S... adalah solusi dari setiap kesimpulan yang didapat. Kemudian dari solusi yang didapat akan mendapat rating menggunakan metode VIKOR.

Berikut ini adalah contoh perhitungan manual menggunakan metode Vikor dalam *ranking*. Proses *ranking* dilakukan tiga alternatif solusi seperti pada Gambar 2.1 "S1A", "S2A", dan "S3A" dengan tiga kriteria dengan penjelasan sebagai berikut:

A. Penjelasan kriteria
   1) Termudah dilakukan. Penilaian diambil dari tingkat kemudahan solusi untuk dilakukan oleh *user*.
   2) Biaya termurah. Penilaian dilakukan berdasarkan biaya yang dikeluarkan jika melakukan solusi tersebut.
   3) Disarankan oleh ahli. Penilaian dilakukan berdasarkan saran dari ahli komputer.

B. Alternatif solusi
   S1A = Cek kabel *power* pada *power supply*, apakah kendor atau tidak. S2A = Cek knop tombol *on/off* pada bagian belakang *power supply*, dan pastikan dalam keadaan on. S3A = Cek *power supply* masih hidup atau tidak, dengan cara menghubungkan pin dengan tembaga sesuai gambar, jika menyala coba pasangkan kembali dengan benar ke komputer.

C. Kriteria
   Kr1 = Termudah dilakukan, Kr2 = Biaya termurah, Kr3 = Disarankan oleh ahli

Tabel 1 Tabel Data Nilai Kriteria

	Kr1	Kr2	Kr3
A1	90	70	100
A2	80	60	80
A3	70	60	40

Keterangan :

- Nilai yang berwarna merah dan digaris bawahi

merupakan nilai terbesar dalam satu kriteria.
- Nilai yang dipertebal dan dimiringkan merupakan nilai terkecil dalam satu kriteria.

Seperti yang ditunjukkan pada Tabel 2.1 pada, masing-masing kriteria penilaian memiliki bobot nilai yang berbeda. Pemberian bobot kriteriadilakukan berdasarkan derajat kepentingan dari masing-masing kriteria. Bobot kriteria diberikan dalam bentuk desimal yang merepresentasikan persentase dari masing-masing kriteria berdasarkan nilai kepentingannya dengan jumlah total dari semua bobot kriteria adalah satu.

Pemberian nilai bobot kriteria dilakukan dengan memperhatikan tingkat kepentingan dari masing-masing kriteria. Untuk kriteria dengan tingkat kepentingan yang lebih tinggi maka nilai bobot kriterianya akan lebih tinggi dibandingkan dengan kriteria yang lain. Dan untuk kriteria dengan tingkat kepentingan yang lebih rendah maka nilai bobot kriterianya akan lebih rendah. Kriteria dengan tingkat kepentingan yang sama maka akan memiliki nilai bobot kriteria yang sama. Contoh :
- K1 memiliki tingkat kepentingan sebesar 50% (10/100 = 0.5)
- K2 memiliki tingkat kepentingan sebesar 30 % (20/100 = 0.3)
- K3 memiliki tingkat kepentingan sebesar 20 % (40/100 = 0.2)

Tabel 2 Tabel Bobot Kriteria

	Kr1	Kr2	Kr3
Bobot	0.5	0.3	0.2

Dari data nilai solusi pada Tabel 2.1 dan data bobot kriteria pada Tabel 2.2 maka akan dilakukan perhitungan manual menggunakan metode Vikor untuk mendapatkan hasil perangkingan dari metode.

Perhitungan metode Vikor dalam melakukan proses ranking dengan tiga alternatif solusi dan tiga jenis kriteria.

A. Alternatif solusi
   S1A = Cek kabel *power* pada *power supply*, apakah kendor atau tidak. S2A = Cek knop tombol *on/off* pada bagian belakang *power supply*, dan pastikan dalam keadaan on. S3A = Cek *power supply* masih hidup atau tidak, dengan cara menghubungkan pin dengan tembaga sesuai gambar, jika menyala coba pasangkan kembali dengan benar ke komputer.
B. Kriteria
   Kr1 = Termudah dilakukan, Kr2 = Biaya termurah, Kr3 = Disarankan oleh pakar

Normalisasi Matriks

$$Rij = \frac{x*j - xij}{x*j - x'j}$$

.....Rumus 1. Normalisasi Matriks Vikor

Keterangan :
Xij = Nilai data sampel i kriteria j
(i = A1, A2, A3)
(j = K1, K2, K3)
X*j = nilai terbaik dalam satu kriteria
X'j = nilai terburuk dalam satu kriteria

Dari data nilai solusi akan dilakukan normalisasi data nilai berdasarkan Rumus 1.

Perhitungan normalisasi data alternatif solusi A1.

1) Kr1
$$\boldsymbol{R_{A1} = \frac{90-90}{90-70} = \frac{0}{20} = 0}$$
2) Kr2
$$\boldsymbol{R_{A1} = \frac{70-70}{70-60} = \frac{0}{10} = 0}$$
3) Kr3
$$\boldsymbol{R_{A1} = \frac{100-100}{100-40} = \frac{0}{60} = 0}$$

Dengan langkah yang sama didapatkan data normalisasi semua alternatif. Hasil normalisasi matrik untuk semua alternatif ditunjukan pada tabel 2.3.

Tabel 3 Normalisasi Matriks

	Kr1	Kr2	Kr3
A1	0	0	0
A2	0.5	1	0.3333333
A3	1	1	1

Seperti yang ditunjukkan pada Tabel 2.3, hasil normalisasi matriks pada Vikor juga memiliki range antara 0.00 – 1.00. Namun pada Vikor nilai 0.00 merupakan nilai terbaik dalam satu kriteria dan nilai 1.00 adalah nilai terburuk dalam satu kriteria. Sedangkan nilai antara 0.00 sampai 1.00 tidak memiliki keunikan. Hal ini dikarenakan dalam Vikor, nilai dengan indeks terkecil adalah nilai yang terbaik.

Setelah mendapatkan hasil normalisasi untuk semua alternatif, maka langkah selanjutnya adalah mengalikan hasil normalisasi dengan bobot kriteria

Perhitungan nilai normalisasi alternatif A1 x bobot kriteria.
- (A1 x Kr1) = 0.00 x 0.5 = 0.00
- (A1 x Kr2) = 0.00 x 0.3 = 0.00
- (A1 x Kr3) = 0.00 x 0.2 = 0.00

Perhitungan hasil perkalian normalisasi dengan bobot kriteria untuk alternatif lain dilakukan dengan menggunakan cara yang sama. Selanjutnya hasil perkalian normalisasi dengan bobot kriteria untuk semua alternatif dapat dilihat pada tabel 4.

Tabel 4 Normalisasi x Bobot

	Kr1	Kr2	Kr3
A1	0	0	0
A2	0.25	0.3	0.66667
A3	0.5	0.3	0.2

Menghitung S

$S_i = \sum_{j=1}^{n} wj \; x \; (Rij)$

$wj = bobot\ kriteria$

.....Rumus 2. Utility Measure

Nilai S didapatkan dari penjumlahan hasil perkalian bobot kriteria dengan data normalisasi pada tiap alternatif. Nilai S didapatkan dengan cara menjumlahkan nilai masing-masing alternatif untuk semua kriteria.

Perhitungan nilai S untuk alternatif solusi A1 menggunakan Rumus 3.

$SA1 = 0.00 + 0.00 + 0.00 = 0.00$

Menghitung R

Ri = *Max* j[wj x Rij]

nilai terbesar dari [wj x Rij]

..... Rumus 3. Regret Measure

Sedangkan untuk mendapatkan nilai R dilakukan dengan cara mencari nilai terbesar dari hasil perkalian normalisasi dengan bobot kriteria berdasarkan masing-masing alternatif.

Nilai R untuk sampel A1 adalah nilai terbesar dari semua kriteria untuk alternatif A1.

$RA1 = 0.00$

Dengan langkah yang sama maka akan diperoleh data nilai S dan R dari semua alternatif yang ditunjukkan pada Tabel 5.

Tabel 5 Nilai S dan R

Sampel	Nilai S	Nilai R
A1	0	0
A2	0.61667	0.3
A3	1	0.5

Pada Tabel 5, untuk masing-masing kolom S dan R terdapat nilai data yang digaris bawahi dan ditebalkan, serta terdapat nilai data yang diberi warna merah dan dimiringkan. Nilai yang digaris bawahi dan ditebalkan merupakan nilai terbesar yaitu 1 untuk nilai S terbesar dan 0.5 untuk nilai R terbesar. Sedangkan nilai yangdiberi warna merah dan dimiringkan merupakan nilai terkecil untuk masing-masing nilai S yaitu 0 dan R yaitu 0.

Menghitung Indeks Vikor

$$Q_i = \left[\frac{Si - s'}{s* - s'}\right] xv + \left[\frac{Ri - r'}{R* - R'}\right] x(1 - v)$$

... Rumus 4. Index Vikor

Keterangan :

S' = nilai S terkecil

S* = nilai S terbesar

R' = nilai R terkecil

R* = nilai R terbesar

Setelah mendapatkan nilai S dan R, langkah terakhir adalah mendapatkan nilai Q dengan menggunakan Rumus 4. Alternatif dengan nilai Q terkecil merupakan alternatif terbaik.

Perhitungan nilai *index* vikor (Q) alternatif A1.

$$\boldsymbol{Q_{A1} = \left[\frac{0-0}{1-0}\right] x\ 0.5 + \left[\frac{0-0}{0.5-0}\right] X(1-0.5) = 0}$$

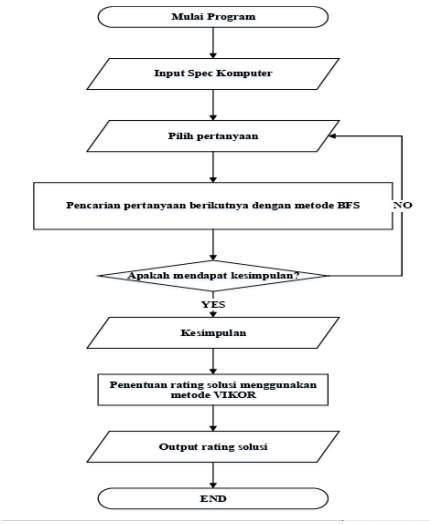
Dari perhitungan menggunakan Rumus 4 nilai index Vikor pada halaman 43 (nilai Q) untuk alternatif A1 adalah 0 dan perhitungan nilai *index* Vikor (Q) untuk semua alternatif dengan langkah yang sama ditunjukkan pada Tabel 2.6.

Tabel 2.6 Indeks Vikor dan Ranking

Sampel	Nilai S	Nilai R	Nilai Q	Ranking
A1	0	0	0	1
A2	0,616667	0.3	0.608333	2
A3	1	0.5	1	3

Nilai *index* Vikor (nilai Q) terkecil merupakan alternatif terbaik sehingga berdasarkan hasil perhitungan, peringkat pertama adalah alternatif A1 memiliki nilai *index* Vikor (nilai Q) terkecil yaitu 0, alternatif A2 berada pada peringkat kedua dengan nilai *index* Vikor (nilai Q) 0.608333 serta alternatif A3 berada pada peringkat terakhir dengan nilai *index* Vikor (nilai Q) 1.

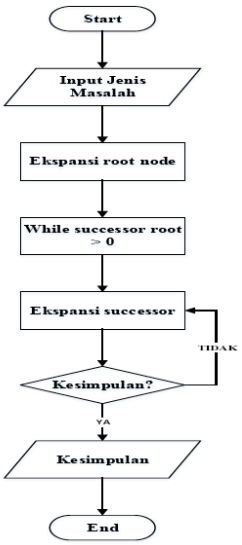
Proses ini dibangun bertujuan untuk mencari solusi dari permasalahan kerusakan *hardware* komputer dan memberikan solusi berdasarkan *ranking*. *Ranking* dilihat dari tingkat kemudahan, biaya yang dikeluarkan, dan terbanyak dilakukan. Alur kerja dari sistem pakar ini dapat dilihat pada Gambar2.



Gambar 2*Flowchart* System Aplikasi Diagnosa Kerusakan *Hardware*

Alur kerja sistem ini dimulai dengan melakukan

input spesifikasi komputer atau laptop yang nantinya akan digunakan untuk memajukan aplikasi yang dibuat, lalu memilih jenis kerusakan apakah itu *hardware* atau jaringan. Lalu proses selanjutanya adalah menampilkan hasil atau kesimpulan dengan metode *Breadth First Search*. Proses pencarian dapat dilakukan berulang-ulang sampai tidak menemukan titik temu pada program dan berakhir dengan peringatan error. Kemudian proses berikutnya akan melakukan pengurutan nilai pada solusi dengan metode Vikor.

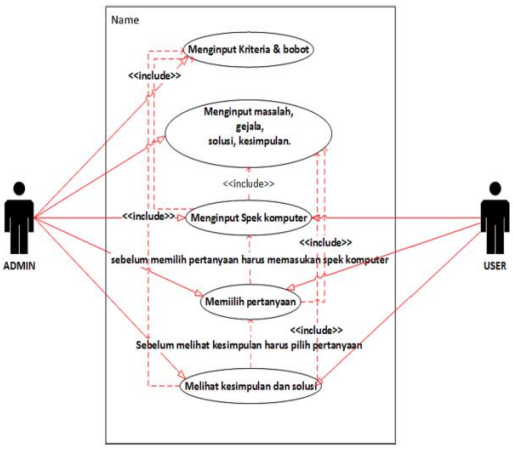


Gambar 3*Flowchart* Metode *Breadth First Search*

Seperti pada Gambar 3, alur kerja *Breadth First Search* dimulai dari pemilihan kepala pohon untuk memulai pencarian selanjutnya. Kepala pohon di pilih oleh *user* yang menggunakan sistem, lalu sistem akan mencari akar-akar pohon setelah kepala pohon dipilih, jika tidak mendapat penyelesaian dengan menemukan akar masalah yang tepat, metode ini akan mencari akar-akar dari kepala pohon lain dan memberikan solusi sesuai dengan kepala akar yang di temukan.

Proses Vikor dimulai dengan pengambilan nilai alternatif dan nilai bobot kriteria lalu dilakukan pencarian nilai terbesar dan terkecil dari setiap kriteria untuk digunakan dalam proses normalisasi dengan menggunakan rumus. Hasil normalisasi tersebut akan dikalikan dengan bobot kriteria berdasarkan masing-masing kriteria dan dari hasil perkalian tersebut akan dihitung nilai *Utility Measure* dengan rumus dan *Regret Measure* dengan rumus. Nilai *Utility Measure* merupakan nilai penjumlahan dari hasil perkalian normalisasi dengan bobot kriteria untuk setiap alternative, sedangkan nilai *Regret Measure* merupakan nilai terbesar dari hasil perkalian normalisasi dengan bobot kriteria untuk masing-masing alternatif. Selanjutnya adalah menghitung nilai *Index* Vikor dengan menggunakan rumus dan penentuan urutan dilakukan berdasarkan nilai *Index* Vikor terkecil seperti yang ditunjukkan Gambar 4.

Gambar 4 *Flowchart* Metode Vikor



Gambar 5 *Use Case* Diagram Aplikasi Diagnosa Kerusakan *Hardware*.

Admin dan *User* adalah aktor, sedangkan fitur yang ada pada sistem digambarkan dengan *use case* yang ada di dalam *system boundary*. Pada sistem ini admin dapat melakukan input kriteria dan bobot serta input masalah dan gejala kerusakan pada *hardware* komputer. Nilai dan masalah yang sudah masukan akan disimpan pada *database* dan akan dipanggil dalam proses pencarian solusi dilakukan dan pengurutan nilai pada saat proses pemilihan solusi. Setelah prosespencarian solusi dengan metode *Breadth First Search* maka akan muncul proses *rating* solusi terbaik oleh metode Vikor.

III. RESULT AND ANALYSIS

Berikut sepotong pseudocode dari Vikor

```
Algoritma Vikor
{mencari rating solusi dari kesimpulan dengan metode Vikor}
Deklarasi
Query : String
Query2 : String
DimensiMin_satu : array [] of String
DimensiMin_dua : array [] of String
DimensiMin_tiga : array [] of String
dimensiMax_empat : array [] of String
dimensiMin_empat : array [] of String
DimensiMax_empat_r : array [] of String
DimensiMin_empat_r : array [] of String
dimensiMax_empat_s : array [] of String
DimensiMin_empat_s : array [] of String
Query_array2 : String
Hasil_array_satu : real
Hasil_array_dua : real
Hasil_array_tiga : real
Hasil_nilai_s : real
Deskripsi
Id_kesimpulan (‘nilai’)
:
:
while(array2←fetch_array(query_array2)) do
hasil_array_satu←(((array2[1]-dimensiMin_empat_s[0]) / (dimensiMax_empat_s[0] - dimensiMin_empat_s[0])) * 0.5) + (((array2[2}- dimensiMin_empat_r[0] / (dimensiMax_empat_r[0] - dimensiMin_empat_r[0]) * (1-0.5)))
write(hasil_array_satu)
query”insert into tbl_nilai(id_nilai,nilainya_q, id_kesimpulan)
values(“,’hasil_array_satu’,’id_kesimpulan’)
```

Gambar 6 Pseudocode Vikor

Seperti Gambar 6*pseudocode* Vikor berfungi untuk mencari *rating* solusi berdasarkan nilai Q yang memiliki *range* dimulai 0 – 1. Langkah pertama adalah mengambil nilai yang ada pada setiap solusi yang dimasukan admin, kemudian mencari nilai max dan nilai min dari setiap nilai untuk masing-masing kriteria yang akan digunakan untuk proses normalisasi.

Setelah mendapatkan nilai normalisasi maka langkah selanjutnya adalah mencari nilai S dan R. Nilai tersebut didapatkan dari hasil perkalian nilai normalisasi dengan nilai bobot kriteria. Langkah terakhir adalah mendapatkan nilai *index* vikor (nilai Q) dengan menggunakan nilai S dan R yang sudah didapatkan sebelumnya dan mengacu pada Rumus *Index* Vikor. Nilai Q terkecil merupakan solusi dengan rating terbaik sedangkan nilai Q terbesar adalah solusi terburuk.

Berikut sepotong *pseudocode* dari Breadth First Search :

```
Algoritma BFS
{mencari jenis masalah, kepala gejala, akar gejala, kesimpulan, dan solusi}
Deklarasi
jenisMasalah : integer
Tidak : String
yes : String
gejala : String
level_gejala : Integer
jenis_masalah2 : Integer
id_kesimpulan : Integer
query_tidak_1 : String
data_tidak_1 : String
Deskripsi
jenisMasalah← POST(‘jenis_masalah’)
tidak←next(‘id_kesimpulan’,’jenisMasalah’)
yes←next(‘yes’,’jenisMasalah’)
gejala←next(‘nama_gejala’,’jenisMasalah)
write (gejala)
level_gejala←next(‘level_gejala’,’jenisMasalah’)
jenis_masalah2←next(‘id_masalah’,’jenisMasalah’)
:
:
write (gejala)
read(next)
goto loop
endif
endif
```

Gambar 7*Pseudocode Breadth First Search*

Seperti pada Gambar 7*Pseudocode Breadth First Search* berfungsi mencari solusi melalui proses pencarian gejala. Setiap gejala memiliki nilai yang akan dipakai pada proses pencarian. Proses pertama sistem akan mencari ID gejala yang berada pada pohon gejala, kemudian akan mencari ID gejala berikutnya yang berada pada pohon gejala sampai node terakhir yang menentukan kesimpulan.

Dari ID kesimpulan akan didapat ID solusi yang akan ditampilkan pada sistem beserta *rating* solusi. Jika node yang dicari tidak ditemukan maka sistem akan menampilkan peringatan bahwa masalah yang dicari oleh *user* tidak ditemukan dan akan mengulang kembali proses dari awal.

```
Function Next(input field:string, input jenisMasalah:integer)→string
{mencari kepala gejala berdasarkan id_gejala}
Deklarasi
Array : String
Query : String
Deskripsi
query←” select masalah.jenisMasalah,
masalah.id_masalah, gejala1.nama_gejala,
gejala1.level_gejala, gejala1.id_kesimpulan,
gejala1.yes,gejala1.id_gejala from masalah, gejala1 where
masalah.id_masalah = gejala1.id_masalah and
gejala1.id_masalah = 'jenisMasalah'"
array←fetch_array(query)
if field=’jenisMasalah’ then
return array[‘nama_gejala’]
else if field = ‘nama_gejala’ then
return array[‘nama_gejala’]
else if field = ‘id_kesimpulan’ then
return array[‘id_kesimpulan’]
else if field = ‘yes’ then
return array[‘yes’]
else if field = ‘id_masalah’ then
return array[‘id_masalah’]
else if field = ‘level_gejala’ then
return array[‘level_gejala’]
end if
```

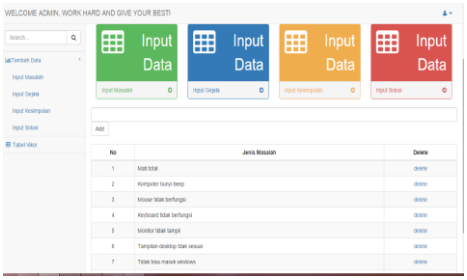
Gambar 8*Pseudocode Function Next*

Seperti Gambar 8 *Pseudocode Function Next* berfungsi mencari gejala melalui proses pencarian. Setiap gejala memiliki nilai yang akan dipakai pada proses pencarian. Proses pertama sistem mencari ID gejala yang berada pada pohon gejala. Proses kedua sistem mencari ID gejala berikutnya yang berada pada pohon gejala sampai node terakhir yang menentukan kesimpulan. Proses ketiga sistem mencari node lain jika belum menemukan kesimpulan.

Berdasarkan penelitian yang penulis lakukan, penulis dapat mengimplementasikan hasil analisis yang telah dilakukan kedalam sebuah aplikasi. Berikut salah satu halaman pada aplikasi yang penulis implementasikan :

A. Halaman *Input* Jenis Masalah

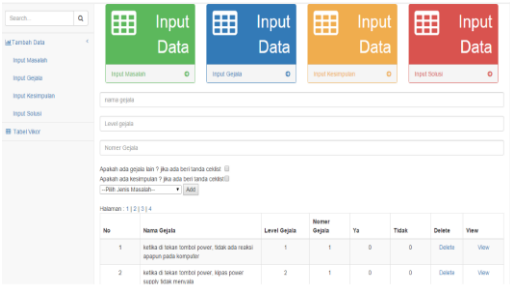
Halaman *input* jenis masalah merupakan *form* yang dapat menambahkan jenis masalah dan langsung dimasukan ke dalam tabel sebagai tampilan setelah ditambahkan.



Gambar 9 Halaman *Input* Jenis Masalah

B. Halaman *Input* Gejala

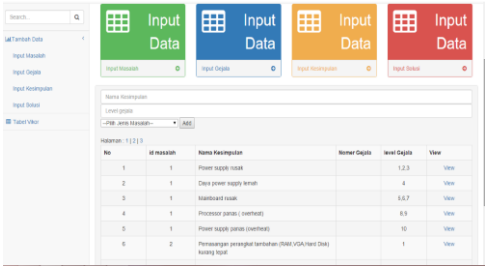
Halaman ini berfungsi untuk menambahkan gejala dan memasukannya ke dalam *database* kemudian akan ditampilkan dalam bentuk tabel di dalam*website* seperti pada Gambar 10.



Gambar 10 Gambar *Input* Gejala

C. Halaman *Input* Kesimpulan

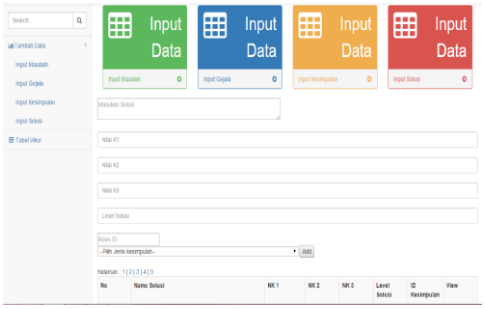
Halaman *input* kesimpulan memiliki fitur menambahkan kesimpulan, kemudian menambahkan *level* gejala yang berfungsi untuk menjalankan metode pencarian seperti pada Gambar 11.



Gambar 11 Gambar *Input* Gejala

D. Halaman *Input* Solusi

Pada halaman ini program dapat menambahkan solusi baru yang akan di *input*, nilai kriteria yang akan berfungsi untuk menghitung *ranking* di dalam program, *level* solusi berfungsi untuk membantu proses pencarian pada metode *Breadth First Search*, dan terakhir adalah rows_id yang berguna untuk menempatkan urutan solusi pada tampilan seperti Gambar 12.



Gambar 12 Gambar *Input* Solusi

E. Halaman Implementasi Tabel Vikor

Halaman tabel Vikor ini berfungsi untuk menjalankan proses perhitungan metode Vikor. Metode Vikor ini dijalankan dalam proses secara manual untuk mencegah admin yang lupa memasukan nilai pada setiap kriteria seperti pada Gambar 13.

Gambar 13 Implementasi Tabel Vikor

IV. CONCLUSION AND EXPRESSION

Berdasarkan hasil penelitian yang telah dibuat, maka dalam penelitian pembuatan aplikasi ini dapat diambil kesimpulan yaitu Algoritma *Breadth First Search* merupakan algoritma yang sesuai untuk aplikasi diagnosa karena algoritma ini akan berjalan terus untuk menemukan solusi atau dengan kata lain

algoritma ini pasti akan menemukan solusi yang dicari oleh *user* jika sesuai dengan *database* dan metode VIKOR merupakan metode yang sesuai untuk sistem rating karena metode ini mengurutkan dengan menggunakan rumus dan menghitung nilai dari solusi pada sistem pakardengan tingkat kesamaan rekomendasi solusi terbaik adalah 99% sesusai dengan rekomendasi ahli.

REFERENCE

[1] Budiharto, Widodo, Suhartono, Derwin. Artificial Intelligence Konsep dan Penerapannya, Yogjakarta. 2014

[2] C.E. Leiserson and T.B. Schardl. A work-e_cient parallel breadth first search algorithm (or how to copet with the nondeterminism of reducers). In Proc. 22nd ACM Symp. on Parallism in Algorithms Ana Architectures (SPAA '10), June 2010.

[3] Datta, Saurav., Siba Sankar Mahapatra, Sabhyasachi Banerjee, Asish Bandyopadhyay. Comparative Study on Application of Utility Concept and VIKOR Method for Vendor Selection, AIMS International Conference on Value-based Management, 2010.

[4] Khezrian, M,. Kadir,. Wan M. N. Wan,. Suhaimi Ibrahim, and Alaeddin Kalantari. Service Selection based on VIKOR method, International Journal of Research and Reviews in Computer Science (IJRRCS) Vol. 2, No. 5, October 2011, ISSN: 2079-2557.

[5] Serafim Opricovic and Gwo Hshiung Tzeng. Extended Vikor Method in Comparison with Outranking Methods, European Journal of Operational Research. 2006.