MENENTUKAN JALUR TERPENDEK MENGGUNAKAN JARINGAN SARAF TIRUAN HOPFIELD

TUGAS AKHIR

Diajukan sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana
Jurusan Teknik Informatika

Disusun Oleh :

Nama : Nur Hasanah

NIM : 03 523 250

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA
2007

MENENTUKAN JALUR TERPENDEK MENGGUNAKAN JARINGAN SARAF TIRUAN HOPFIELD

TUGAS AKHIR

Diajukan sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana
Jurusan Teknik Informatika

Disusun Oleh :

Nama : Nur Hasanah

NIM : 03 523 250

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA
2007

LEMBAR PERNYATAAN KEASLIAN
HASIL TUGAS AKHIR

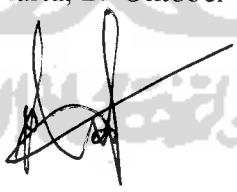
Saya yang bertandatangan di bawah ini,

Nama : Nur Hasanah
NIM : 03 523 250

Menyatakan bahwa seluruh komponen dan isi dalam Laporan Tugas Akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti bahwa ada beberapa bagian dari karya ini adalah bukan hasil karya saya sendiri, maka saya akan siap menanggung resiko dan konsekuensi apapun

Demikian pernyataan ini saya buat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 29 Oktober 2007



Nur Hasanah

LEMBAR PENGESAHAN PEMBIMBING

MENENTUKAN JALUR TERPENDEK MENGGUNAKAN JARINGAN SARAF TIRUAN HOPFIELD

TUGAS AKHIR

Disusun Oleh :

Nama : Nur Hasanah

NIM : 03 523 250

Yogyakarta, 29 Oktober 2007

Pembimbing,

Taufiq Hidayat, ST, MCS.

LEMBAR PENGESAHAN PENGUJI

MENENTUKAN JALUR TERPENDEK MENGGUNAKAN JARINGAN SARAF TIRUAN HOPFIELD

TUGAS AKHIR

Oleh :

Nama : **Nur Hasanah**

No. Mahasiswa : **03 523 250**

Telah Dipertahankan di Depan Sidang Penguji Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana Jurusan Teknik Informatika
Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 27 April 2007

Tim Penguji

Taufiq Hidayat, ST, MCS
Ketua

Sri Kusumadewi, S.Si, MT
Anggota I

Lizda Iswari, ST
Anggota II

Mengetahui,
Ketua Jurusan Teknik Informatika
Universitas Islam Indonesia

(Yudi Prayudi, S.si, M.Kom)

Halaman Persembahan

I dedicated to :

My heroes and my inspirations, Mom & Dad

My only sista and her husband, mba Jin and Mas Jefry

MOTTO

Wahai mereka yang beriman, mintalah pertolongan kepada Allah dengan sabar dan solat. Sesungguhnya Allah bersama-sama dengan orang yang sabar." (Al-Baqarah: 153)

"Dan, hanya kepada Allah-lah orang-orang mukmin bertawakal." (Ali 'Imran: 122)

Kemenangan kita yang paling besar bukanlah karena kita tidak pernah jatuh, melainkan karena kita bangkit setiap kali jatuh (CONFUSIUS)

Sahabat adalah dorongan ketika engkau hampir berhenti, petunjuk jalan ketika engkau tersesat, memapahmu saat engkau hampir tergelincir dan mengalungkan mutiara doa pada dadamu...Ikhwan and akhwat...moga hati kita dipertautkan karena-Nya.

DAFTAR ISI

LEMBAR PENGESAHAN ii

LEMBAR PENGESAHAN PEMBIMBING ..... *iii*

LEMBAR PENGESAHAN PENGUJI iv

HALAMAN PERSEMBAHAN v

HALAMAN MOTTO vi

KATA PENGANTAR vii

DAFTAR ISI ix

DAFTAR GAMBAR xii

DAFTAR TABEL xiv

ABSTRAKSI xv

I. BAB I PENDAHULUAN

1.1 Latar Belakang Masalah 1

1.2 Rumusan Masalah 2

1.3 Batasan Masalah 3

1.4 Tujuan Penelitian 4

1.5 Manfaat Penelitian 4

1.6 Metodologi Penelitian 4

1.6.1 Metode Pengumpulan Data 4

1.6.2 Metode Pengembangan Sistem 5

1.7 Sistematika Penulisan 5

II. BAB II LANDASAN TEORI

2.1 *Teori graf* 7

2.1.1 Definisi graf 7

2.1.2 Macam –macam graf 8
2.2 Optimisasi 9
2.2.1 Definisi optimisasi 9
2.2.2 Definisi Nilai Optimal 10
2.2.3 Macam-macam Persoalan Optimisasi 10
2.3 Permasalahan Jalur Terpendek (*Shortest Path Problem*) 11
2.4 Jaringan Saraf Tiruan 12
2.4.1 Cara Kerja Komponen Jaringan Syaraf Tiruan 17
2.4.2 Konsep Belajar Jaringan Saraf Tiruan 18
2.4.3 Arsitektur Jaringan Syaraf Tiruan 20
2.4.4 Fungsi Aktivasi atau Fungsi Transfer 23
2.4.5 Jaringan Saraf Tiruan Hopfield Diskrit 24
2.4.6 Arsitektur Jaringan Saraf Tiruan Hopfield 26
2.4.7 *Shortest Path Problem* dengan Jaringan Syaraf Tiruan *Hopfield* 28

III. BAB III METODOLOGI

3.1 Analisis Kebutuhan Perangkat Lunak 32
3.1.1 Metode Analisis 32
3.1.2 Hasil Analisis 32
3.1.2.1 Analisis kebutuhan proses 32
3.1.2.2 Analisis kebutuhan Masukan 33
3.1.2.3 Analisis kebutuhan Keluaran 34
3.1.3 Kebutuhan Antar Muka 35
3.1.4 Analisis Kebutuhan Perangkat Lunak 35
3.1.5 Analisis Kebutuhan Perangkat Keras 35
3.2 Perancangan Perangkat Lunak 36
3.2.1 Metode Perancangan 36

3.2.2 Hasil Perancangan 36
3.2.2.1 Perancangan antar muka 46
3.3 Implementasi Perangkat Lunak 47
3.3.1 Batasan Implementasi 47
3.3.2 Implementasi antarmuka 47
3.2.2.1 Halaman Utama 47
3.2.2.2 Halaman komputasi 48
3.3.3 Implementasi prosedural 51

IV. BAB IV HASIL DAN PEMBAHASAN

4.1 Pengujian program 57
4.1.1 Pengujian tidak normal 57
4.2 Analisis Kerja Sistem 59
4.2.1 Analisis berdasarkan jumlah titik 59
4.2.2 Analisis berdasarkan nilai parameter 63

V. BAB V KESIMPULAN DAN SARAN

5.1 Kesimpulan 67
5.2 Saran 67

SARI

Shortest Path Problem atau Kasus Jalur Terpendek merupakan salah satu contoh kasus dalam bidang optimasi. Tujuan dari kasus jalur terpendek ini adalah untuk mencari jalur minimum yang diperlukan untuk mencapai dari tempat asal ke tempat tujuan berdasarkan beberapa jalur alternatif yang tersedia. Setiap tempat tidak selalu mempunyai jalur yang terhubung dengan tempat tempay lainnya.

Ada banyak metode yang digunakan untuk memecahkan kasus ini. Salah satunya adalah dengan menggunakan cabang ilmu jaringan saraf tiruan (JST).

Jaringan syaraf tiruan merupakan salah satu representasi buatan dari otak manusia yang selalu mencoba untuk mensimulasikan proses pembelajaran pada otak manusia tersebut. Seperti halnya otak manusia, jaringan syaraf juga terdiri-dari beberapa neuron yang mentransformasikan informasi yang diterima melalui sambungan keluarnya menuju ke neuron-neuron yang lain dengan nama bobot.

Jaringan saraf tiruan yang digunakan untuk memecahkan kasus jalur terpendek adalah jaringan saraf tiruan Hopfield. Pada jaringan Hopfield, semua neuron saling berhubungan penuh. Neuron yang satu mengeluarkan output dan kemudian menjadi input bagi semua neuron yang lain. Proses pengiriman dan penerimaan sinyal antar neuron ini secara feedback tertutup terus menerus sampai dicapai kondisi stabil.

Jaringan saraf tiruan dapat memberikan solusi optimal untuk kasus jalur terpendek dengan menggunakan jenis arsitektur jaringan yang sesuai. JST Hopfield merupakan tools yang efektif untuk menyelesaikan kasus jalur terpendek.

Kata Kunci: *Kasus Jalur Terpendek, Jaringan Saraf Tiruan, Hopfield, Optimasi.*

BAB I

PENDAHULUAN

1.1 Latar Belakang

Kehadiran komputer memang sangat membantu dalam menyelesaikan segala permasalahan yang manusia miliki. Seiring dengan kebutuhan manusia yang semakin hari semakin kompleks, teknologi komputer pun berkembang pesat dalam berbagai bidang. Komputer akhirnya terpilih sebagai salah satu alternatif solusi dalam pengambilan keputusan yang tepat dan akurat. Hasil kerja sistem komputer ini diakui lebih cepat, teliti dan akurat dibandingkan dengan manusia, hal inilah yang mendorong lahirnya Kecerdasan Buatan atau *Artificial Intellegence.*

Jaringan saraf tiruan (*Artificial Neural Networks*) adalah salah satu cabang ilmu dari bidang ilmu Kecerdasan Buatan. Pada dasarnya JST mencoba meniru cara kerja otak makhluk hidup, yaitu bentuk *neuron*-nya (sel syaraf). Faktor kecerdasan dari syaraf tidak ditentukan di dalam sel tetapi terletak pada bentuk dan topologi jaringannya. JST dapat menyelesaikan persoalan yang rumit atau tidak mungkin jika diselesaikan dengan menggunakan komputasi secara konvensional. Sejumlah besar algoritma pelatihan telah dikembangkan untuk membuat berbagai macam aplikasi JST seperti optimasi. JST dapat digunakan untuk menemukan solusi optimal dari masalah – masalah yang melibatkan banyak parameter, misalnya menentukan Jalur Terpendek (*Shortest Path*). Salah satu

menentukan keputusan jalur terpendek di antara jalur-jalur yang tersedia dari tempat asal ke tempat tujuan.

1.3 Batasan Masalah

Dalam melaksanakan suatu penelitian diperlukan adanya batasan agar tidak menyimpang dari yang telah direncanakan sehingga tujuan yang sebenarnya dapat tercapai. Batasan masalah yang diperlukan yaitu:

1. Sistem yang akan dibangun merupakan aplikasi JST Hopfield untuk menentukan jalur terpendek.
2. Input sistem adalah koordinat kota, jalur antar kota, myu, jumlah epoh, kota asal dan kota tujuan.

   Myu merupakan parameter dari metode yang dipakai dalam aplikasi yaitu jaringan saraf tiruan Hopfield. Jumlah myu yang digunakan ada lima macam dan akan diisi bebas oleh pengguna aplikasi untuk menentukan jalur terpendek.

   Epoh adalah kumpulan dari beberapa iterasi perhitungan untuk mendapatkan satu jenis jalur. Pengguna bebas mengisi jumlah maksimal epoh.
3. Output sistem adalah graf jalur terpendek yang terdiri dari node-node dari kota asal ke kota tujuan dan total jarak jalur terpendeknya.
4. Perangkat lunak dibangun menggunakan bahasa pemrograman Java.

1. Literatur

   Menggunakan berbagai macam literatur yang berhubungan dengan JST Hopfield dan jarak terpendek.

2. Observasi

   Mengadakan observasi dan mengajukan pertanyaan-pertanyan kepada narasumber yang mengetahui tentang hal yang berhubungan dengan topik.

3. Referensi Internet

   Mencari referensi melalui internet.

1.6.2 Metode pengembangan sistem

Metode pengembangan sistem yang digunakan meliputi analisis kebutuhan perangkat lunak, perancangan perangkat lunak, implementasi perangkat lunak dan analisis kinerja perangkat lunak.

1.7 Sistematika Penulisan

Dalam penyusunan tugas akhir ini, sistematika penulisan dibagi menjadi beberapa bab sebagai berikut:

BAB I PENDAHULUAN

Bab ini berisi pembahasan masalah umum yang meliputi latar belakang masalah, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian dan sistematika penulisan

BAB II

LANDASAN TEORI

2.1 Teori Graf

2.1.1 Definisi Graf

Graf digunakan untuk merepresentasikan objek objek diskrit dan hubungan antara objek-objek tersebut [ZAK06].

Graf $G = (V, E)$, yang dalam hal ini:

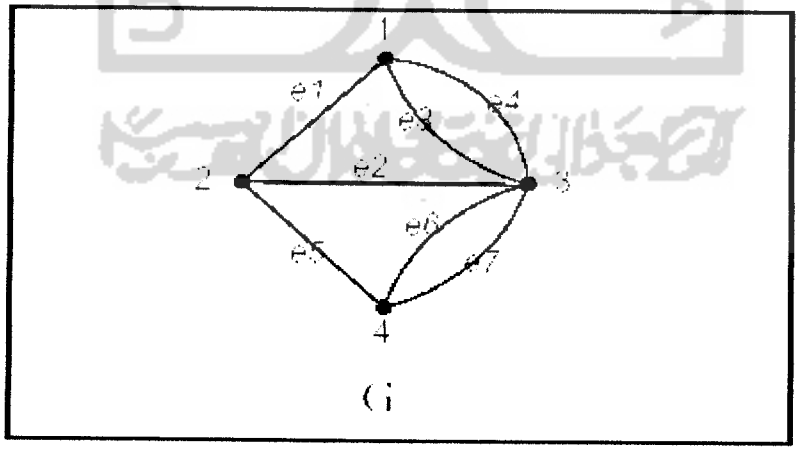
V = himpunan tidak-kosong dari simpul-simpul (*vertices*)

$= \{ v1 , v2 , ... , vn \}$

E = himpunan sisi (*edges*) yang menghubungkan sepasang simpul

$= \{e1 , e2 , ... , en \}$

Contoh graf ada pada gambar 2.1.



Gambar 2.1 Graf

Keterangan Gambar 2.1:

G adalah graf dengan

V = { 1, 2, 3, 4 }

E = { (1, 2), (2, 3), (1, 3), (1, 3), (2, 4), (3, 4), (3, 4) }

= { $e1$, $e2$, $e3$, $e4$, $e5$, $e6$, $e7$}

2.1.2 Macam-macam Graf

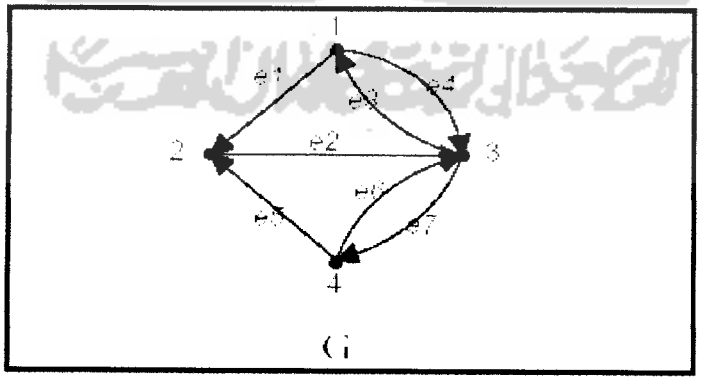
Berdasarkan orientasi arah pada sisi, maka secara umum graf dibedakan atas 2 jenis:

1. Graf tak-berarah (*undirected graph*)

   Graf yang sisinya tidak mempunyai orientasi arah disebut graf tak-berarah.

2. Graf berarah (*directed graph* atau *digraph*)

   Graf yang setiap sisinya diberikan orientasi arah disebut sebagai graf berarah. Contoh graf berarah dapat dilihat pada gambar 2.2.



Gambar 2.2 Graf Berarah

Dua buah simpul $v1$ dan simpul $v2$ disebut **terhubung** jika terdapat lintasan dari $v1$ ke $v2$.

2.2.2 Definisi Nilai Optimal

Nilai optimal adalah nilai yang didapat melalui suatu proses dan dianggap menjadi suatu solusi jawaban yang paling baik dari semua solusi yang ada.

Nilai optimal dapat dicari dengan dua cara, yaitu:

1. Cara konvensional, yaitu mencoba semua kemungkinan yang ada dengan mencatat nilai yang didapat cara ini kurang efektif, karena optimasi akan berjalan secara lambat.
2. Cara kedua adalah dengan menggunakan suatu rumus atau gambar sehingga nilai optimal dapat diperkirakan dengan cepat dan tepat.

2.2.3 Macam-macam Persoalan Optimisasi

Persoalan yang berkaitan dengan optimisasi sangat kompleks dalam kehidupan sehari-hari. Nilai optimal yang didapat dalam optimisasi dapat berupa besaran panjang, waktu, jarak, dan lain-lain.

Berikut ini adalah beberapa persoalan yang memerlukan optimisasi:

1. Menentukan lintasan terpendek dari suatu tempat ke tempat yang lain.
2. Menentukan jumlah pekerja seminimal mungkin untuk melakukan suatu proses produksi agar pengeluaran biaya pekerja dapat diminimalkan dan hasil produksi tetap maksimal.
3. Mengatur jalur kendaraan umum agar semua lokasi dapat dijangkau.
4. Mengatur *routing* jaringan kabel telepon agar biaya pemasangan kabel tidak terlalu besar dan penggunaannya tidak boros.

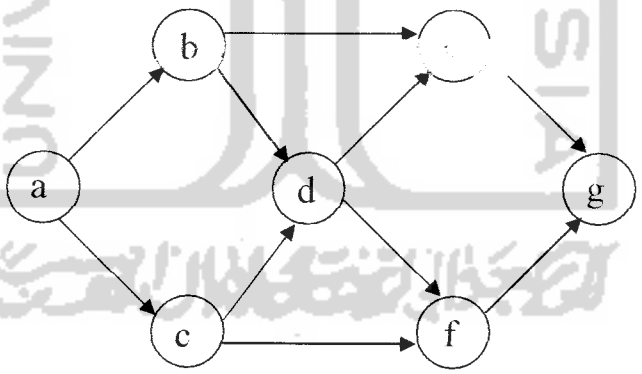
Selain beberapa contoh di atas, masih banyak persoalan lainnya yang terdapat di kehidupan sehari-hari.

2.3 Permasalahan Jalur Terpendek (*Shortest Path Problem*)

Jalur terpendek adalah lintasan minimum yang diperlukan untuk mencapai suatu kota dari kota tertentu berdasarkan beberapa jalur alternatif yang tersedia [MUT07]. Lintasan minimum yang dimaksud dapat dicari dengan menggunakan graf. Graf yang digunakan adalah graf yang berbobot, yaitu graf yang setiap sisinya diberikan suatu nilai atau bobot.

Dalam kasus ini, bobot yang dimaksud berupa jarak antara dua kota.

Gambar 2.4 menunjukkan suatu graf abcdefg.



Gambar 2.4 Graf abcdefg

Pada gambar diatas, misalkan kita dari kota a ingin menuju kota g. Untuk menuju kota g, dapat dipilih beberapa jalur yang tersedia :

a → b → c→d → e → g

a → b → c → d → f → g

a → b → c → d → g

a → b → c → e→ g

a → b → d → e → g

a → b → d → f → g

a → b → d → g

a → b → e → g

a → c → d → e → g

a → c →d → f → g

a → c→ d → g

a → c → f → g

Berdasarkan data pada gambar 2.4, dapat dihitung jalur terpendek dengan mencari jarak antara kota-kota pada tiap jalur. Jarak antar jalur belum yang belum diketahui dapat dihitung berdasarkan koordinat kota-kota tersebut. Setelah didapatkan hasil jarak antar kota, jalur terpendek dapat dihitung menggunakan metode yang ada.

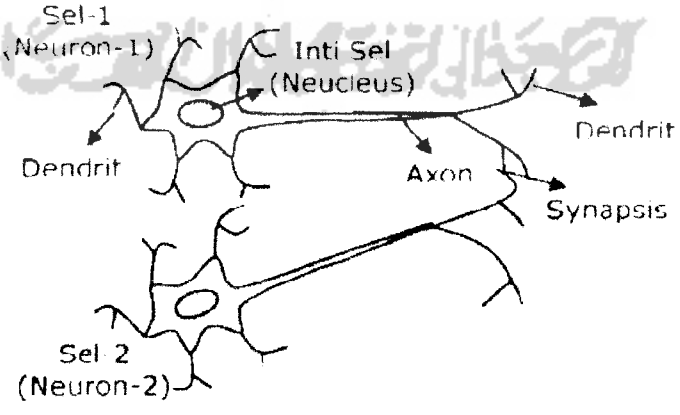
2.4 Jaringan Saraf Tiruan

Jaringan Saraf tiruan bisa dibayangkan seperti otak buatan yang dapat berpikir seperti manusia dan juga sepandai manusia dalam menyimpulkan sesuatu dari potongan-potongan informasi yang diterima. Khayalan manusia tersebut mendorong para peneliti untuk mewujudkannya. Komputer diusahankan agar bisa

berpikir sama seperti cara berpikir manusia. Caranya adalah dengan melakukan peniruan terhadap aktivitas-aktivitas yang terjadi di dalam sebuah jaringan saraf biologis [PUS06].

Salah satu contoh pengambilan ide dari jaringan saraf biologis adalah adanya elemen-elemen pemrosesan pada jaringan saraf tiruan yang saling terhubung dan beroperasi secara paralel. Ini meniru jaringan saraf biologis yang tersusun dari sel-sel saraf (neuron). Cara kerja dari elemen-elemen pemrosesan jaringan saraf tiruan juga sama seperti meng-*encode* informasi yang diterimanya. Jaringan saraf biologis merupakan kumpulan sel-sel saraf (neuron). Neuron mempunyai tugas mengolah informasi. Komponen-komponen utama dari sebuah neuron dapat dikelompokkan menjadi tiga bagian, yaitu :

1. *Dendrit*. Dendrit bertugas untuk menerima informasi.
2. *Badan sel (soma)*. Badan sel berfungsi sebagai tempat pengolahan informasi.
3. *Akson (neurit)*. Akson mengirimkan impuls-impuls ke sel saraf lainnya.



Gambar 2.5 Sel saraf biologis

Pada gambar 2.5 dapat diperhatikan sebuah neuron menerima impuls-impuls sinyal dari neuron yang lain melalui dendrit dan mengirimkan sinyal yang dibangkitkan oleh badan sel malalui akson akson dari sel saraf biologis ini bercabang-cabang dan berhubungan dengan dendrit dari sel saraf lainnya dengan cara mengirimkan impuls melaui *sinapsis*. Sinapsis adalah unit fungsional antara dua buah sel saraf, katakanlah A dan B, di mana yang satu adalah serabut akson dari neuron A dan satunya lagi dendrit dari neuron B.

Jaringan saraf tiruan disusun dengan asumsi yang sama seperti jaringan saraf biologis [PUS06].:

1. Pengolahan informasi terjadi pada elemen-elemen pemrosesan (neuron-neuron).
2. Sinyal antara dua buah neuron diteruskan melalui link-link koneksi.
3. Setiap link koneksi memiliki bobot terasosiasi.
4. Setiap neuron menerapkan sebuah fungsi aktivasi terhadap input jaringan jumlah sinyal input berbobot). Tujuannya adalah untuk menentukan sinyal output. Fungsi aktivasi yang digunakan biasanya fungsi yang nonlinier.

Tabel 2.1 Keanalogan jaringan saraf tiruan terhadap jaringan saraf biologis

Jaringan saraf tiruan	**Jaringan saraf biologis**
Node atau unit	Badan (sel soma)
Input	Dendrit
Output	Akson
Bobot	Sinapsis

Tabel 2.1 menunjukkan keanalogan antara jaringan saraf tiruan dengan jaringan saraf biologis, dimana node diasumsikan sebagai sel soma, input diasumsikan sebagai dendrit, output diasumsikan sebagai akson dan bobot diasumsikan sebagai sinapsis.

Jaringan saraf tiruan memiliki beberapa kelebihan yaitu [PUS06]:

1. Mampu memecahkan masalah yang sukar disimulasikan dengan menggunakan teknik analitikal logika seperti pada sistem pakar dan teknologi software standar.
2. Mampu memahami data yang dimasukkan meskipun data tersebut tidak lengkap (*incomplete data*) atau data yang terkena gangguan (*noisy data*).
3. Jaringan saraf tiruan yang sulit diciptakan dengan pendekatan simbolik/logikal dari teknik tradisional *artificial intelligence*, yaitu bahwa jaringan saraf tiruan mampu belajar dari pengalaman.
4. Hemat biaya dan lebih nyaman bila dibandingkan dengan harus menulis program seperti *software* standar.
5. Jaringan saraf tiruan terbuka untuk digabungkan dengan tekonologi lain, misalnya dengan *expert system*, logika *fuzzy*, algoritma genetika, atau diintregasikan dengan *database*.

Meskipun memiliki banyak kelebihan, jaringan saraf tiruan juga memiliki sejumlah keterbatasan antara lain kekurangmampuannya dalam melakukan operasi-operasi numerik dengan presisi tinggi, operasi algoritma aritmatik, operasi

logika, dan operasi simbolis serta lamanya proses pelatihan yang kadang-kadang membutuhkan waktu berhari-hari untuk jumlah data yang besar. Hal ini terjadi karena sulitnya mengukur *performance* sebenarnya dari jaringan saraf tiruan.

Dengan tidak mengesampingkan kelemahan-kelemahan di atas, jaringan saraf tiruan secara umum dapat dikatakan baik dalam menangani masalah-masalh pada bidang-bidang berikut:

1. Bidang yang melibatkan pengklasifikasian fitur geometrik atau fitur fisik.

   Contohnya :

   1. Pengenalan suara
   2. Pembelajaran robot
   3. Pengalan tulisan

2. Pengklasifikasian pola data.

   Contohnya :

1. Diagnosis medikal
2. Diagnostik mesin jet dan roket
3. Bidang yang melibatkan masalah optimasi

   Contohnya :

   1. *Traveling Salesman Problemn*
   2. *Chinese Postman Problem*

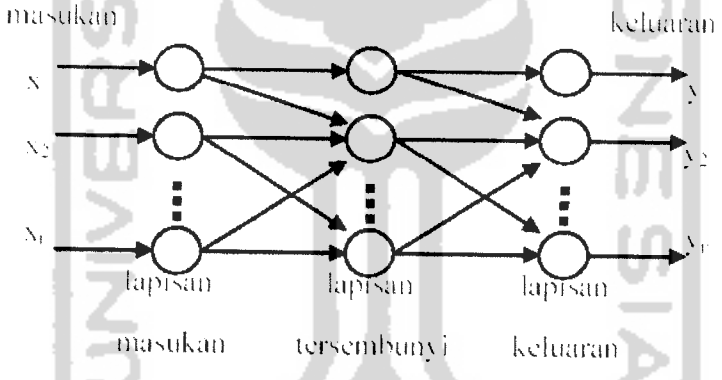
2.4.1 Cara Kerja Komponen Jaringan Syaraf Tiruan

Ada beberapa tipe jaringan syaraf tiruan, tetapi hampir semuanya memiliki komponen yang sama. Sama halnya seperti otak manusia, jaringan syaraf juga terdiri dari beberapa neuron, dan ada hubungan antara neuron tersebut. Neuron-neuron tersebut akan mentransformasikan informasi yang diterima melalui sambungan keluarnya menuju ke neuron-neuron yang lain. Pada JST hubungan ini dikenal dengan nama bobot. Informasi tersebut disimpan pada suatu nilai tertentu pada bobot tersebut [KUS03].

Gambar 2.6 Struktur neuron jaringan syaraf tiruan

Informasi (disebut dengan *input*) akan dikirim ke neuron dengan bobot kedatangan tertentu. *Input* ini akan diproses oleh suatu fungsi perambatan yang akan menjumlahkan nilai-nilai semua bobot yang datang. Hasil penjumlahan ini kemudian akan dibandingkan dengan suatu nilai ambang *(threshold)* tertentu melalui fungsi aktivasi setiap neuron. Apabila *input* tersebut melewati suatu nilai ambang tertentu , maka neuron tersebut akan diaktifkan, tetapi jika tidak maka neuron tersebut tidak akan diaktifkan. Apabila neuron tersebut diaktifkan, maka neuron tersebut akan mengirimkan *output* melalui bobot-bobot *output*-nya ke semua neuron yang berhubungan dengannya. Hal ini dilakukan secara terus menerus.

Pada jaringan syaraf, neuron-neuron akan dikumpulkan dalam lapisan-lapisan (*layer*) yang disebut dengan lapisan neuron (*neuron layers*). Biasanya neuron-neuron pada satu lapisan akan dihubungkan dengan lapisan-lapisan sebelum dan sesudahnya (kecuali lapisan *input* dan lapisan *output*). Informasi yang diberikan pada jaringan syaraf akan dirambatkan dari lapisan ke lapisan, mulai dari lapisan *input* sampai ke lapisan *output* melalui lapisan yang lainnya, yang sering dikenal dengan nama lapisan tersembunyi (*hidden layer*), tergantung pada algoritma pembelajarannya, bisa jadi informasi tersebut akan dirambatkan secara mundur pada jaringan.



Gambar 2.7 Jaringan syaraf dengan tiga lapisan

Beberapa jaringan syaraf ada juga yang tidak memiliki lapisan tersembunyi, dan ada juga yang neuron-neuronnya disusun dalam bentuk matriks.

2.4.2 Konsep Belajar Jaringan Syaraf Tiruan

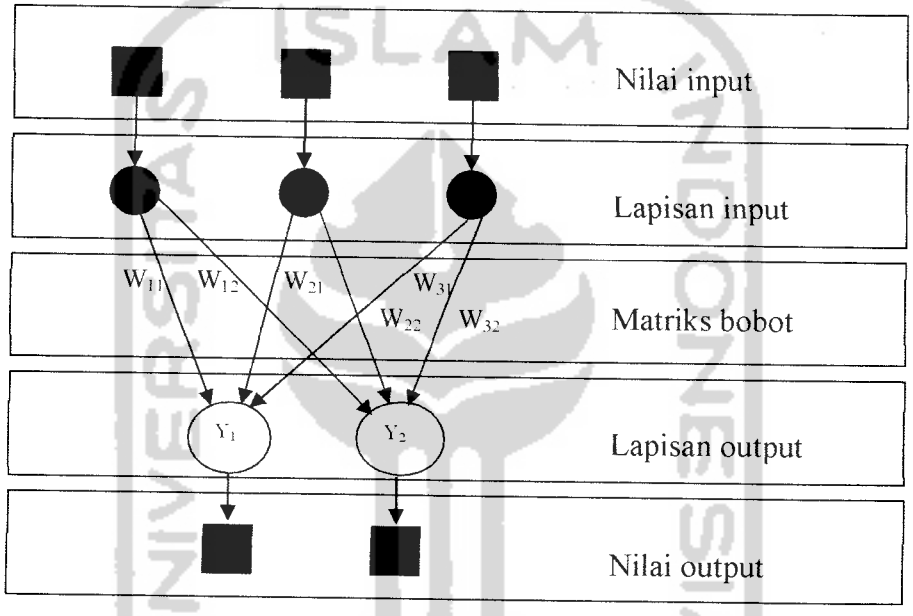
Ciri utama yang dimiliki oleh sistem jaringan syaraf tiruan adalah kemampuan untuk belajar. Agar berfungsi seperti yang diinginkan, jaringan tidak diprogram seperti yang dilakukan pada sistem komputer sekarang ini, melainkan harus diajari[KUS03].

2.4.3 Arsitektur Jaringan Syaraf Tiruan

Dalam jaringan syaraf, neuron-neuron dikelompokan dalam lapisan-lapisan yang umumnya setiap lapisan yang sama akan memiliki keadaan yang sama pula. Apabila neuron-neuron dalam suatu lapisan (misalkan lapisan tersembunyi) akan dihubungkan dengan neuron-neuron pada lapisan lain (misalkan lapisan *output*), maka setiap neuron pada lapisan tersebut (misalkan lapisan tersembunyi) juga harus dihubungkan pada setiap neuron pada lapisan yang lainnya (misalkan lapisan *output*). Ada beberapa arsitektur jaringan syaraf, antara lain [KUS03]:

1. Jaringan lapis tunggal (*single layer net*).

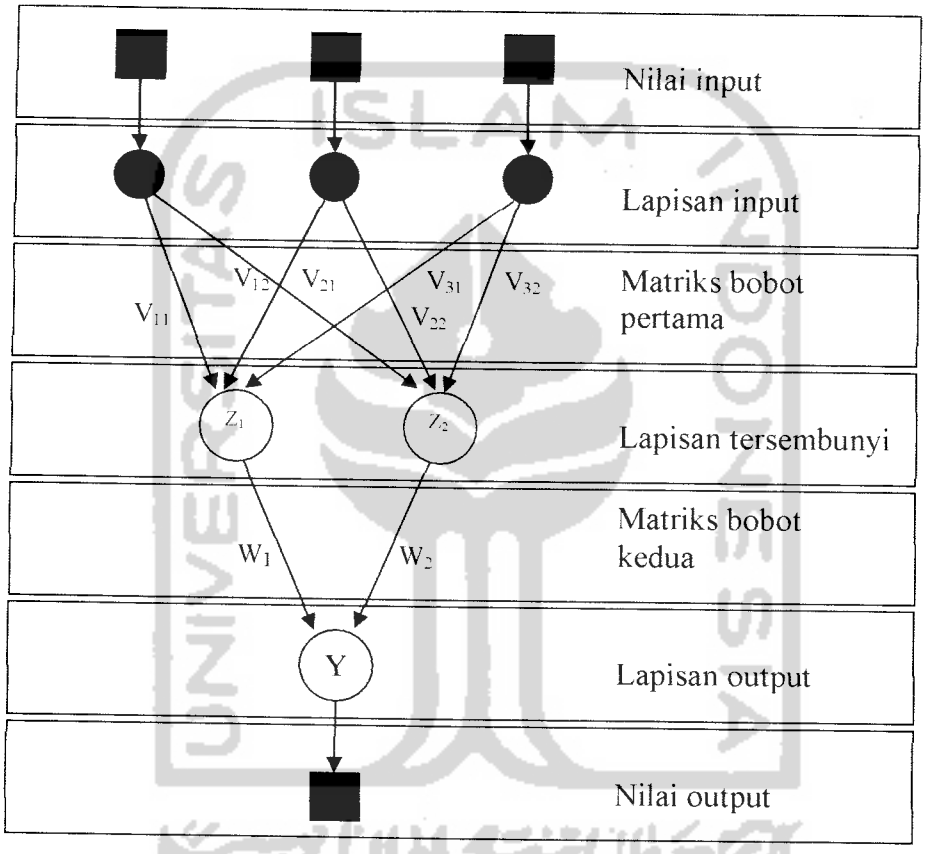
Jaringan yang memiliki arsitektur ini hanya memiliki satu buah lapisan bobot koneksi. Jaringan lapisan tunggal terdiri dari unit-unit input yang menerima sinyal dari dunia luar, dan unit-unit output dimana kita bisa membaca respons dari jaringan saraf tiruan tersebut.



Gambar 2.8 *Single Layer Net*

2. Jaringan multilapis *(multi layer net)*.

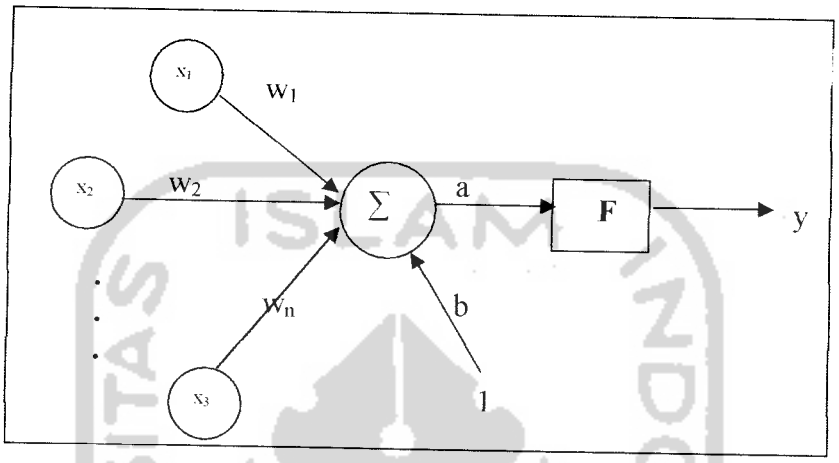
   Merupakan jaringan dengan satu atau lebih lapisan tersembunyi dan memiliki kemampuan lebih dalam memecahkan masalah bila dibandingkan dengan *single layer net*, namun pelatihannya lebih rumit.



Gambar 2.9 *Multi Layer Net*

f. Fungsi *sigmoid biner*

g. Fungsi *sigmoid bipolar*



Gambar 2.11 Fungsi Aktivasi JST

2.4.5 Jaringan Saraf Tiruan Hopfield Diskrit

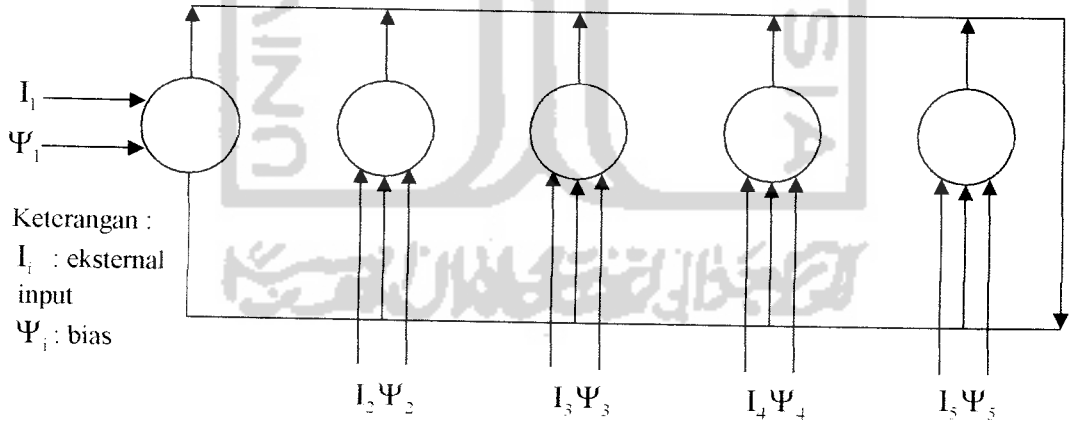
Pada jaringan Hopfield, semua neuron saling berhubungan penuh. Neuron yang satu mengeluarkan output dan kemudian menjadi input bagi semua neuron yang lain. Proses pengiriman dan penerimaan sinyal antar neuron ini secara *feedback* tertutup terus menerus sampai dicapai kondisi stabil [KRI04].

Dalam model diskritnya, jaringan Hopfield bobot sinaptiknya menggunakan vektor biner dimensi n atau a{0,1}n. Model semacam ini berisi n neuron dan jaringannya terdiri dari n(n-1) interkoneksi dua jalur.

Berikut bobot-bobot yang digambarkan sebagai vektor W:

$$W = \begin{bmatrix} 0 & w_{12} & w_{13} & w_{14} & w_{15} \\ w_{21} & 0 & w_{23} & w_{24} & w_{25} \\ w_{31} & w_{32} & 0 & w_{34} & w_{35} \\ w_{41} & w_{42} & w_{43} & 0 & w_{45} \\ w_{51} & w_{52} & w_{53} & w_{54} & 0 \end{bmatrix}$$

Bobot-bobot yang terletak pada diagonal utamanya adalah n yang menujukkan bahwa neuron-neuron pada jaringan Hopfield tidak memiiki hubungan dengan dirinya sendiri ($w_{ij} = 0;\ i = j$). Dengan kata lain, setiap neuron tidak memberi input kepada dirinya sendiri. Sementara itu kesimetrisan vektor bobot berarti berlakunya $w_{ij} = w_{ji}$ di mana $i \neq j$, sehingga $w_{12} = w_{21}$, $w_{13} = w_{31}$, $w_{14} = w_{41}$ dan seterusnya.



Gambar 2.12 Topologi JST Hopfield

Pada gambar 2.12 ditunjukkan bahwa semua neuron saling berhubungan penuh. Neuron yang satu mengeluarkan output dan kemudian menjadi input bagi neuron yang lain. Output setiap simpul diumpanbalikkan ke input dari simpul

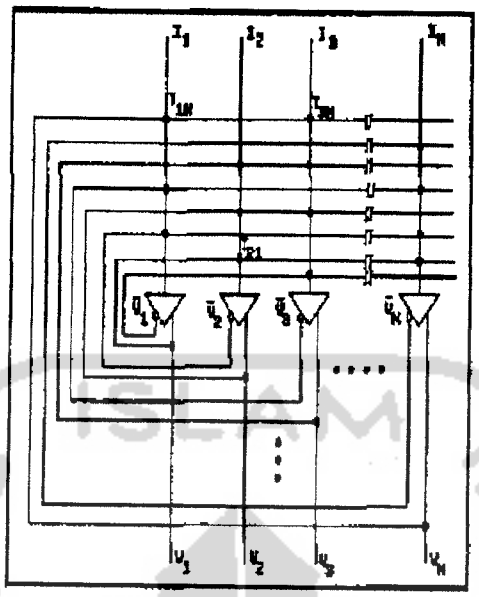
lainnya melalui bobot koneksi w_{ij} yang tetap. Nilai w_{ij} mula-mula diinisialisasi menggunakan rumus yang diberikan Hopfield untuk seluruh pola-pola contoh.

Pola-pola yang tidak dikenal dimasukkan ke jaringan satu kali saja yaitu pada waktu nol. Kemudian jaringan beriterasi hingga *konvergen.* Konvergen adalah kondisi dimana output yang sudah tidak berubah lagi untuk iterasi selanjutnya. emamp yang dinyatakan oleh simpul-simpul output setelah jaringan konvergen adalah output JST.

Kemampuan JST Hopfield untuk mengenali pola ditentukan dengan rumus 0.15 x n. Banyak dimensi yang digunakan disebut n. Misalnya dimensi yang digunakan 100, maka pola yang akan dikenali sebanyak 15 buah. Dimensi adalah suatu vektor yang mewakili banyaknya baris dan kolom yang digunakan. Dari contoh diatas berarti baris yang digunakan berjumlah 100 dan kolom yang akan digunakan berjumlah 100 juga.

2.4.6 Arsitektur Jaringan Saraf Tiruan Hopfield

Sirkuit jaringan *Hopfield* dapat dilihat pada gambar 2.13. Desain sirkuit ini diadaptasi dari komponen-komponen dasar jaringan saraf biologis. Setiap neuron dimodelkan sebagai sebuah *amplifier* dengan fungsi aktivasi *sigmoid* dengan input U_i dan output V_i dari neuron ke-i. Output dari V_i merupakan bilangan 0 atau 1 [ALI93].



Gambar 2.13 Jaringan Saraf Tiruan Hopfield

Fungsi sigmoid:

$$V_i = g_i(U_i) = \frac{1}{1 + e^{-\lambda_i U_i}} \tag{1}$$

Setiap neuron menerima *input impuls* dari neuron-neuron lainnya. Hubungan ini biasanya disebut dengan koneksi sinapsis dan dideskripsikan melalui matrik $T = [T_{ij}]$. Dalam jaringan, matrik T disebut juga dengan matrik hubungan. Setiap neuron juga menerima bias yang disebut I_i. Rumus perubahan U_i dari jaringan Hopfield adalah sebagai berikut:

$$\frac{dU_i}{dt} = \sum_{j=1}^{n} T_{ij} V_j - \frac{U_i}{\tau} + I_i \tag{2}$$

Dimana τ adalah konstanta dari waktu sirkuit.

Dari kedua persamaan di atas didapatkan pula fungsi energi:

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} T_{ij}V_iV_j - \sum_{i=1}^{n} I_iV_i \qquad (3)$$

Berdasarkan rumus energi (3), rumus perubahan neuron ke i[th] dirumuskan sebagai berikut:

$$\frac{dU_i}{dt} = -\frac{U_i}{\tau} - \frac{\partial E}{\partial V_i} \qquad (4)$$

2.4.7 *Shortest Path Problem* dengan Jaringan Syaraf Tiruan *Hopfield*

Pada kasus *Shortest Path Problem*, tujuannya adalah menemukan rute yang terpendek antara node *source s*, ke *node destination d*, di jaringan yang ada. Jaringan didefinisikan sebagai sebuah graf berarah $G = (N,A)$, dengan N adalah banyaknya node dan A adalah banyaknya sisi (*arc*) atau dalam kasus ini adalah jalan (path). Adapun C_{ij}, adalah nilai yang menunjukkan besarnya jarak, panjang, atau waktu transit dari node *i* ke node *j*. Rute terpendek dapat dirumuskan dengan graf P^{sd}, yang berisi urutan node-node yang menghubungkan *s* ke *d*:

$$P^{sd} = (s, i, j, k, ..., r, d) \qquad (5)$$

Maka, panjang dari jalur terpendeknya adalah $L^{sd} = C_{si} + C_{ij} + C_{jk} + ... + C_{rd}$. Inti permasalahannya adalah mencari rute L^{sd} dengan panjang minimum.

Pada kasus *shortest path*, node-node disediakan dalam bentuk matrik n x n. Dengan semua elemen diagonal dihilangkan, karena tidak dibutuhkan. Setiap elemen merepresentasikan neuron dengan indikator (i,j), dimana i menyatakan kolom dan j menyatakan nomor dari node. Jadi, sebuah jaringan memerlukan

n (n - 1) neuron, dan neuron dengan letak (i,j) menggambarkan output V_{ij}, dengan ketentuan sebagai berikut:

$$V_{ij} = \begin{cases} 1 & \text{Bila jalur dari node i ke node j ada dalam rute terpendek} \\ 0 & \text{Sebaliknya} \end{cases} \quad (6)$$

Dan juga ditentukan ρ_{ij} sebagai berikut:

$$\rho_{ij} = \begin{cases} 1 & \text{Bila tidak ada jalur dari node i ke node j} \\ 0 & \text{Sebaliknya} \end{cases} \quad (7)$$

Selain V_{ij} dan ρ_{ij}, diperlukan juga C_{ij}, yaitu besarnya jarak dari node *i* ke node *j* dan harus bernilai positif. Untuk jalur yang tidak ada, maka besarnya jarak akan dianggap sebagai nol, dan jalur ini akan tereliminasi dari solusi.

Untuk menetukan apakah sebuah rute merupakan rute terpendek pada jaringan Hopfield ini digunakan sebuah fungsi energi (3) yang telah dimodifikasi, dimana hasil akan diperoleh bila energi yang dihasilkan minimum. Berikut adalah rumus fungsi energinya:

$$E = \frac{\mu 1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1 \\ j\neq i}}^{n} C_{ij}\cdot V_{ij} + \frac{\mu 2}{2}\sum_{i=1}^{n}\sum_{\substack{j=1 \\ j\neq i}}^{n} \rho_{ij}\cdot V_{ij} + \frac{\mu 3}{2}\sum_{i=1}^{n}\left(\sum_{\substack{j=1 \\ j\neq i}}^{n} V_{ij} - \sum_{\substack{j=1 \\ j\neq i}}^{n} V_{ij} - \gamma_i\right)^2$$
$$+ \frac{\mu 4}{2}\sum_{i=1}^{n}\sum_{\substack{j=1 \\ j\neq i}}^{n}\left(V_{ij}\cdot\left(1 - V_{ij}\right)\right) + \frac{\mu 5}{2}\sum_{i=1}^{n}\sum_{\substack{j=1 \\ j\neq i}}^{n}(1 - V_{ds}) \quad (8)$$

Koefisien C_{ij} merupakan besar jarak dari kota *i* ke kota *j* dan V_{ij} menggambarkan koneksi antara kota *i* dan kota *j*, dimana akan bernilai 1 jika kedua kota tersebut tidak terkoneksi atau berhubungan dan akan bernilai 1 jika terkoneksi. Konstanta $\mu 1$ digunakan untuk meminimalkan total jarak; konstanta $\mu 2$ digunakan untuk mencegah jalur yang tidak ada agar tidak ikut dalam rute; konstanta $\mu 3$ bernilai 0 untuk setiap node (jumlah jalur yang masuk sama dengan jumlah jalur yang keluar); konstanta $\mu 4$ digunakan agar jaringan dapat meraih kondisi konvergen; konstanta $\mu 5$ bernilai nol disaat output. V_{ij} adalah sama dengan 1, untuk memastikan shortest path yang ada menpunyai sumber dan tujuan yang sesuai.

Berdasarkan persamaan (1), (2), dan (4) maka didapatkan rumus perubahan U_{ij} :

$$\frac{dU_{ij}}{dt} = -\frac{U_{ij}}{\tau} + \sum_{k=1}^{n}\sum_{\substack{l=1 \\ l\neq k}}^{n} T_{ij,kl}V_{kl} + I_{ij} \tag{9a}$$

$$\frac{dU_{ij}}{dt} = -\frac{U_{ij}}{\tau} - \frac{\partial E}{\partial V_{ij}} \tag{9b}$$

$$V_{ij} = g_{ij}(U_{ij}) = \frac{1}{1+e^{-\lambda_{ij}U_{ij}}} \tag{10}$$

$$\forall (i,j) \in \overline{N}x\overline{N} \,/\, i \neq j.$$

Dengan mensubstitusikan (8) di (9b), persamaan untuk perubahan jaringan saraf menjadi:

$$\frac{dU_{ij}}{dt} = -\frac{U_{ij}}{\tau} - \frac{\mu 1}{2} C_{ij}(1-\delta_{id}.\delta_{js}) - \frac{\mu 2}{2}\rho_{ij}(1-\delta_{id}.\delta_{js}) - \mu 3 \sum_{\substack{k=1 \\ k\neq i}}^{n} (V_{ik} - V_{ki})$$

$$+\mu 3 \sum_{\substack{k=1 \\ k\neq j}}^{n} (V_{jk} - V_{kj}) - \frac{\mu 4}{2}(1-2V_{ij}) + \frac{\mu 5}{2}\delta_{id}\delta_{js} \qquad (11)$$

$$\forall (i,j) \in \overline{N} x \overline{N} / i \neq j.$$

Dimana δ merupakan delta Kronecker dimana:

$$\delta_{ab} = \begin{cases} 1 & \text{Bila a = b} \\ 0 & \text{Sebaliknya} \end{cases} \qquad (12)$$

Dengan membandingkan persamaan (10a) dan (12), maka kekuatan koneksi dan bias ditetapkan sebagai berikut:

$$T_{ij,kl} = \mu 4 \delta_{ik}\delta_{jl} - \mu 3 \delta_{ik} - \mu 3 \delta_{jl} + \mu 3 \delta_{il} + \mu 3 \delta_{jk} \qquad (13)$$

$$I_{ij} = -\frac{\mu 1}{2} C_{ij}(1-\delta_{id}.\delta_{js}) - \frac{\mu 2}{2}\rho_{ij}(1-\delta_{id}.\delta_{js}) - \frac{\mu 4}{2} + \frac{\mu 5}{2}\delta_{id}.\delta_{js}$$

$$= \begin{cases} \frac{\mu 5}{2} - \frac{\mu 4}{2} & \text{jika } (i,j) = (d,s) \\ -\frac{\mu 1}{2} C_{ij} - \mu 2 \rho_{ij} - \frac{\mu 4}{2} & \text{sebaliknya} \end{cases} \qquad (14)$$

$$\forall (i \neq j), \forall (k \neq j).$$

BAB III

METODOLOGI

3.1 Analisis Kebutuhan Perangkat Lunak

3.1.1 Metode Analisis

Metode analisis berfungsi untuk menganalisis kebutuhan dari perangkat lunak Menentukan Jalur Terpendek dengan JST Hopfield (Shortest Path Hopfield).

Pada tahap analisis ini, digunakan suatu alat untuk melakukan pemodelan agar pengembangan perangkat lunak dapat memenuhi semua kebutuhan pengguna dengan lengkap dan tepat. Pemodelan dilakukan menggunakan notasi UML (*Unified Modelling Language*) yang merupakan standar dalam dunia industri perangkat lunak untuk melakukan visualisasi, perancangan dan pendokumentasian suatu perangkat lunak yang berorientasi objek.

3.1.2 Hasil Analisis

Dari data yang diperoleh melalui survey dan wawancara selama penelitian dan setelah dilakukan proses analisis yang terdiri dari kebutuhan proses, kebutuhan input dan kebutuhan keluaran, yaitu :

3.1.2.1 Analisis Kebutuhan Proses

Kebutuhan proses dalam sistem penentuan jalur terpendek antara lain:

- Proses pembuatan kota dan rute pada peta
- Proses penentuan kombinasi jalur selanjutnya
- Proses penentuan jalur terpendek
- Proses pelaporan hasil

3.1.2.2 Analisis Kebutuhan Masukan

Input atau masukan dari aplikasi penentuan jalur terpendek ini adalah:

1. **Input Data Kota**

   Input koordinat kota diinputkan langsung oleh pengguna. Proses ini menggunakan peta dua dimensi yang menggunakan titik X dan Y sebagai input yang kemudian ditentukan kota-kota dari peta tersebut. Pengguna dapat menge-klik langsung pada *layout* peta atau memasukkan nilai X dan Y satu per satu.

2. **Input Variabel JST Hopfied**

   Masukan data yang berupa variabel-variabel JST Hopfield.

   a. Myu 1 ($\mu 1$)

      Konstanta $\mu 1$ digunakan untuk meminimalkan total jarak.

   b. Myu 2 ($\mu 2$)

      Konstanta $\mu 2$ digunakan untuk mencegah jalan yang tidak ada agar tidak ikut dalam jalur.

   c. Myu 3 ($\mu 3$)

      Konstanta $\mu 3$ bernilai 0 untuk setiap node (jumlah jalur yang masuk sama dengan jumlah jalur yang keluar.

d. Myu 4 ($\mu 4$)

Konstanta $\mu 4$ digunakan agar jaringan dapat meraih kondisi konvergen.

e. Myu 5 ($\mu 5$)

Konstanta $\mu 5$ bernilai nol disaat output. I adalah sama dengan 1, untuk memastikan shortest path yang ada menpunyai sumber dan tujuan yang sesuai.

f. Max Epoh

Max Epoh digunakan untuk menentukan jumlah epoh atau siklus yang digunakan untuk mendapatkan jalur tependek.

g. Kota Asal dan Kota Tujuan

3.1.2.3 Analisis Kebutuhan Keluaran

Sedangkan kebutuhan keluaran yang dihasilkan oleh perangkat lunak pencarian jalur terpendek yaitu :

1. Jarak antar kota : jarak antar kota yang dihasilkan dari posisi koordinat kota.
2. Grafik koordinat kota : dihasilkan dari pemasukan posisi koordinat kota
3. Pelaporan : dihasilkan dari pemasukan koordinat dan pemasukan parameter algoritma Hopfield.

3.1.3 Kebutuhan Antar Muka

Perancangan antar muka dari aplikasi *Shorthest path using Hopfield Neural Network* menggunakan *Swing (Java™ Foundation Classes)* dengan *Notepad++* sebagai editornya.

Penggunaan Java Swing merupakan pilihan yang tepat untuk mengimplementasikan perangkat lunak, dengan tampilan yang indah dan memudahkan pengguna untuk menggunakan sistem, dan sifat orientasi obyek dari Java juga membuat programmer lebih mudah untuk mengembangkan perangkat lunak.

Penggunaan Notepad++ juga memudahkan pengguna dalam menulis *listing* program karena error-error dalam tiap baris mudah ditemukan. Selain itu, penggunaan Java akan membuat programmer lebih mudah untuk memisahkan tampilan dari kode, sehingga memudahkan pengembangan perangkat lunak.

3.1.4 Analisis Kebutuhan Perangkat Lunak

1. Sun Microsystem Java 2 Standard Edition SDK (J2SDK 1.6)
2. Notepad++
3. Perancangan UML menggunakan Rational Rose

3.1.5 Analisis Kebutuhan Perangkat Keras

Perangkat keras yang digunakan pada perangkat lunak pencarian jalur terpendek menggunakan JST Hopfield adalah :

a. Processor AMD Barton 2500 1,8 GHz

b. Memori 512 MB

c. Hardisk 40 GB

d. VGA Ati Radeon HIS 9250 128MB

e. Monitor

f. Mouse

g. Keyboard

3.2 Perancangan Perangkat Lunak

3.2.1 Metode Perancangan

Metode perancangan yang digunakan dalam pengembangan aplikasi ini adalah menggunakan bahasa UML (*Unified Modelling Language*) yang kemudian di perjelas perancangan terstruktur (*structure design method*) atau flow chart. Flow chart pada dasarnya merupakan konsep perancangan yang mudah dengan penekanan pada sistem modular (*Top Down Design*) dan pemrograman terstruktur (*structure programming*).

Tahapan perancangan yang dibahas akan menghasilkan kebutuhan sistem aplikasi dan pemilihan teknologi.

3.2.2 Hasil Perancangan

a. *Unified Modelling Language* (UML)

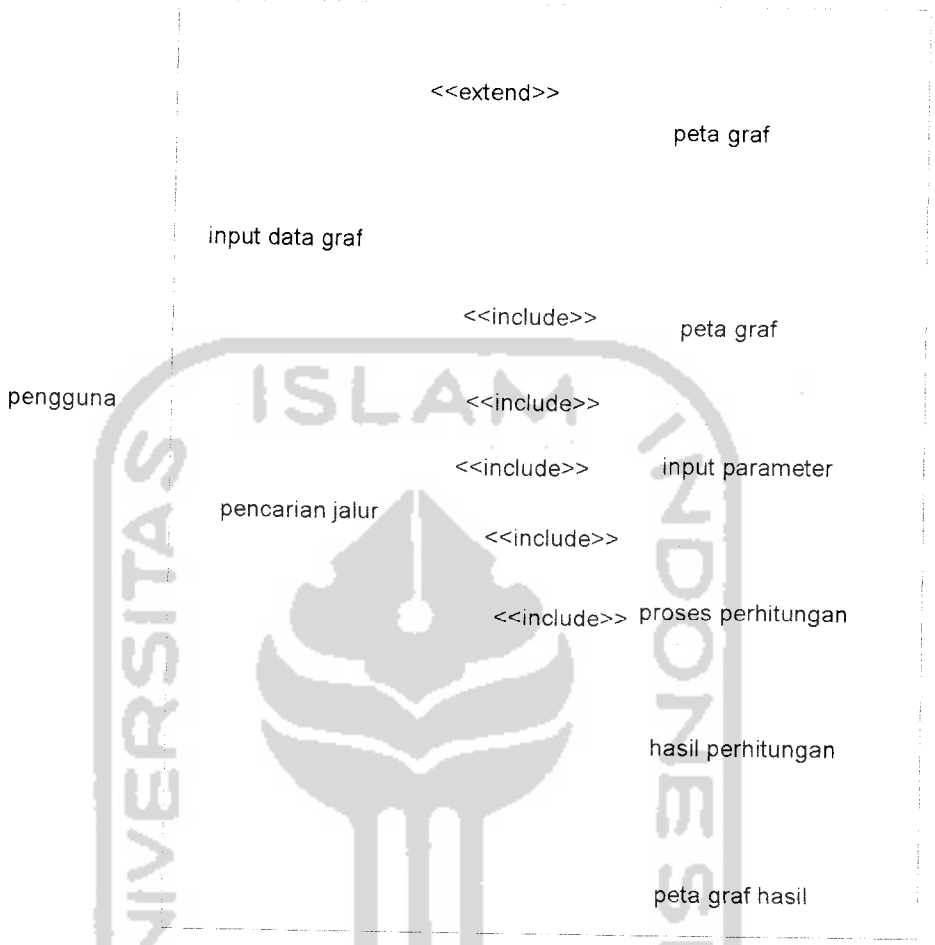
UML merupakan bahasa pemodelan yang dapat digunakan untuk berbagai tujuan, yang menggunakan standar notasi tertentu. UML juga menjadi standar industri yang dibuat di bawah pengawasan *Object Management Group* (OMG).

Untuk lebih menjelaskan perancangan aplikasi yang dibangun, digunakan 4 (empat) model diagram, yaitu : *use case diagram*, *class diagram*, *sequence diagram dan activity diagram*.

1. *Use Case Diagram*

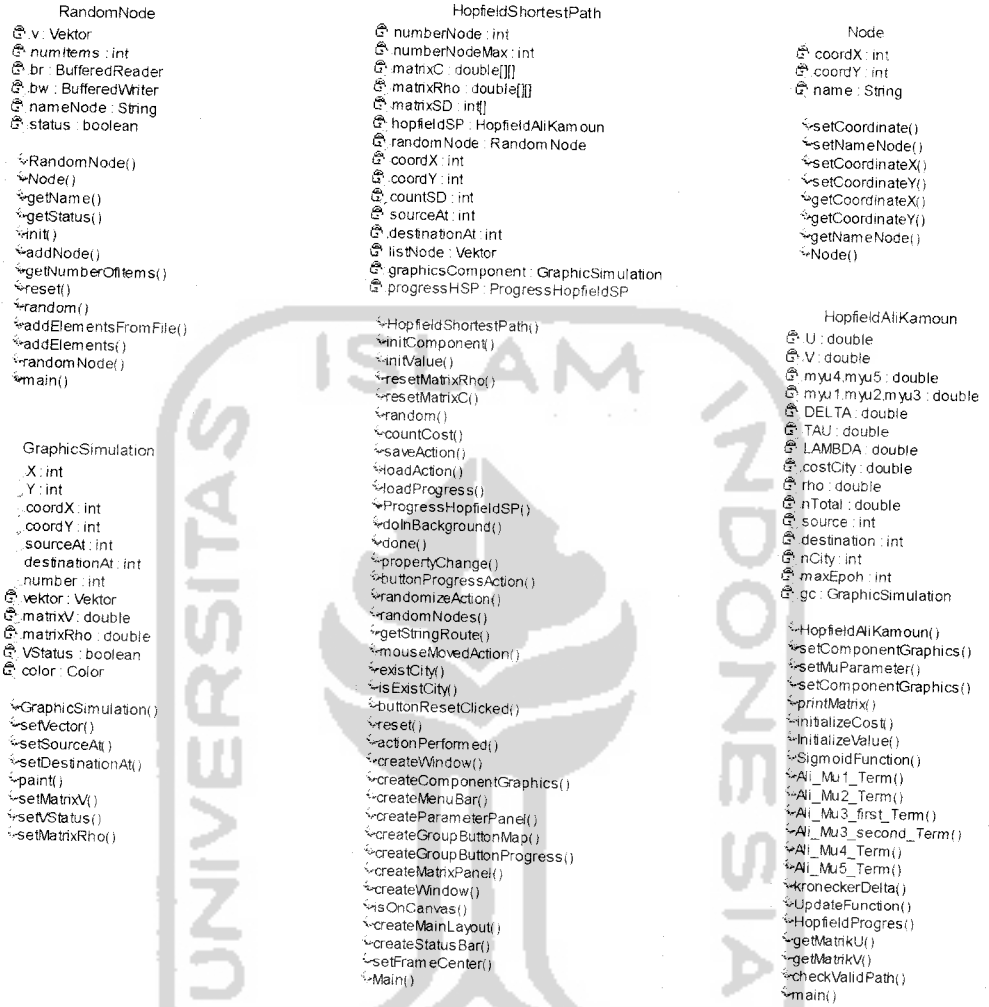
Use case diagram menggambarkan fungsi-fungsi yang berlangsung dilihat dari sisi pengguna. Diagram ini menunjukkan struktur sederhana dari suatu sistem. *Use case* merupakan skenario tertulis dari suatu proses bisnis. Pada rancangan yang dibuat, hanya terdapat 1 (satu) aktor saja. aktor tersebut dapat berupa manusia, perangkat keras, sistem lain, ataupun yang berinteraksi dengan sistem.

Pada aplikasi pencarian jalur terpendek antar kota menggunakan JST Hopfield, *use case* menjelaskan tentang hubungan antara sistem dengan aktor. Hubungan ini dapat berupa input aktor ke sistem ataupun output ke aktor. Gambar 3.1 menjelaskan aplikasi Menetukan Jalur Terpendek menggunakan JST Hopfield (Shortest Path Hopfield) dalam model *use-case diagram*.



Gambar 3.1 Use Case Diagram Shortest Path Hopfield

Pengguna dapat melakukan dua aktivitas yaitu input data graf dan pencarian jalur. Pengguna dapat melakukan input data graf dengan cara memasukkan peta graf, tetapi hal ini tidak harus dilakukan bila pengguna lebih memilih sistem untuk merandom peta graf secara otomatis. Pengguna dapat melakukan pencarian jalur dengan cara memasukkan peta graf, memasukkan parameter, lalu program akan melakukan proses perhitungan jalur terpendek kemudian akan diperoleh hasil dari



Gambar 3.2 Class Diagram Aplikasi Shortest Path Hopfield

Pada gambar 3.2 dapat dilihat bahwa pada aplikasi *Shortest Path Hopfield* terdapat lima kelas yaitu *HopfieldShortestPath*, *HopfieldAliKamoun*, *GraphicSimulation*, *RandomNode* dan *Node*. Kelas *HopfieldShortestPath* merupakan kelas utama dan mempunyai hubungan asosiasi dengan empat kelas lainnya, dimana kelas *HopfieldShortestPath* menggunakan empat kelas lainnya tanpa mengubah data dan method kelas tersebut. Kelas *Node*, *RandomNode* dan

GraphicSimulation digunakan untuk membuat segala sesuatu yang berhubungan dengan peta seperti membuat node kota, nama kota, jalur kota dan jalur terpendeknya. Kelas *HopfieldAliKamoun* menyimpan semua data dan method yang berhubungan dengan jaringan saraf tiruan Hopfield.

Kelas *GraphicSimulation* diinstance menjadi objek di kelas *HopfieldShortestPath* dengan nama *GraphicComponent*. Kelas *GraphicSimulation* merupakan kelas yang berupa extends dari *Jcomponent*, sehingga pada kelas *HopfieldShortestPathProblem*, *GraphicComponent* digunakan untuk membuat berbagai komponen pembentuk GUI seperti panel, label, *textfield* dan tombol.

Kelas *RandomNode* diinstance menjadi objek di kelas *HopfieldShortestPath* dan digunakan di method di kelas *HopfieldShortestPath* seperti *randomNode* dan *mouseClickAction*. Pada kedua method ini, digunakan juga beberapa fungsi dari kelas *RandomNode* seperti fungsi *getNode* untuk mengambil nama node, fungsi *addNode* untuk menambah node dan *getNumberOfItems* untuk mengambil nama dari daftar node jika user ingin merandom node.

Kelas *Node* diinstance menjadi objek di kelas *HopfieldShortestPath* dan digunakan pada beberapa method seperti *mouseClickAction* dan *mouseMovedMotions*. Fungsi dari kelas Node yang digunakan antara lain *getNameNode*, *getCoordinate*, *setNameNode*, dan *setCoordinate*.

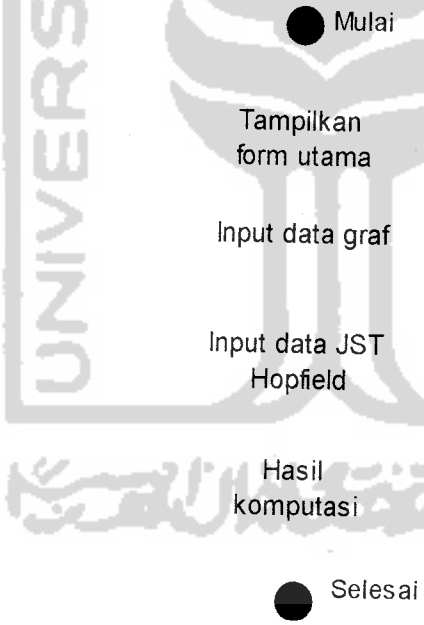
3. Sequence Diagram

Sequence diagram menunjukkan urutan waktu dan proses yang terjadi pada sebuah fungsi sistem sehingga dapat terlihat pertukaran data yang terjadi diantaranya. Pada aplikasi Shortest Path Hopfield, ada dua sequence diagram yaitu Input Data Graf dan Pencarian Jalur.

4. Sequence Diagram

Activity diagram menggambarkan berbagai aliran aktivitas dalam sistem yang sedang di rancang, bagaimana masing-masing aliran berawal, decision yang mungkin terjadi dan bagaimana mereka berakhir. Activity diagram juga dapat menggambarkan proses pararel yang mungkin terjadi pada beberapa eksekusi.

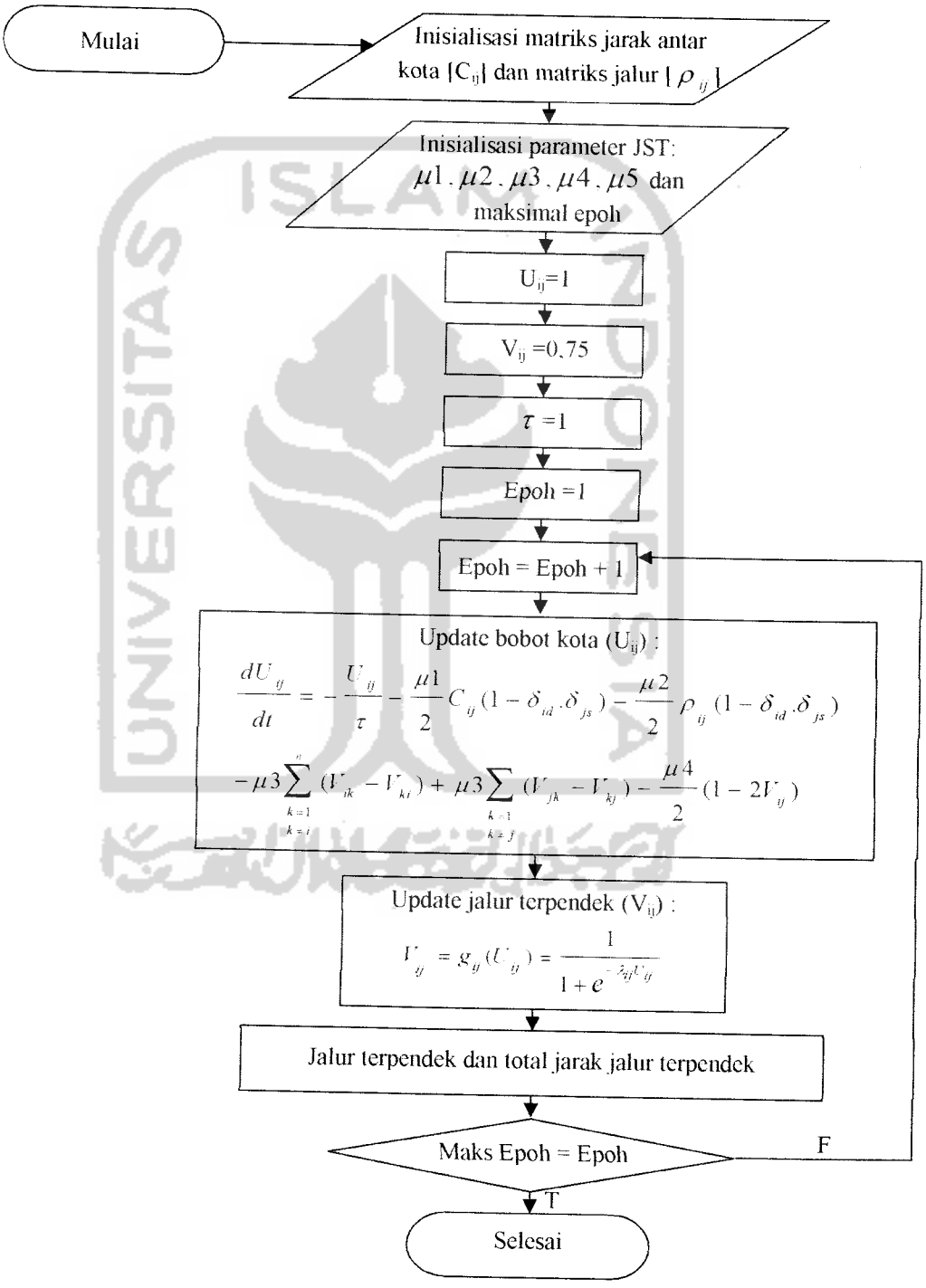
Activity Diagram dari aplikasi Shortest Path Hopfield dapat dilihat pada gambar 3.5.



Gambar 3.5 Activity Diagram Aplikasi Shortest Path Hopfield

b. Perancangan Flow Chart

Flow chart digunakan untuk memperjelas perancangan dan algoritma yang akan dibuat.

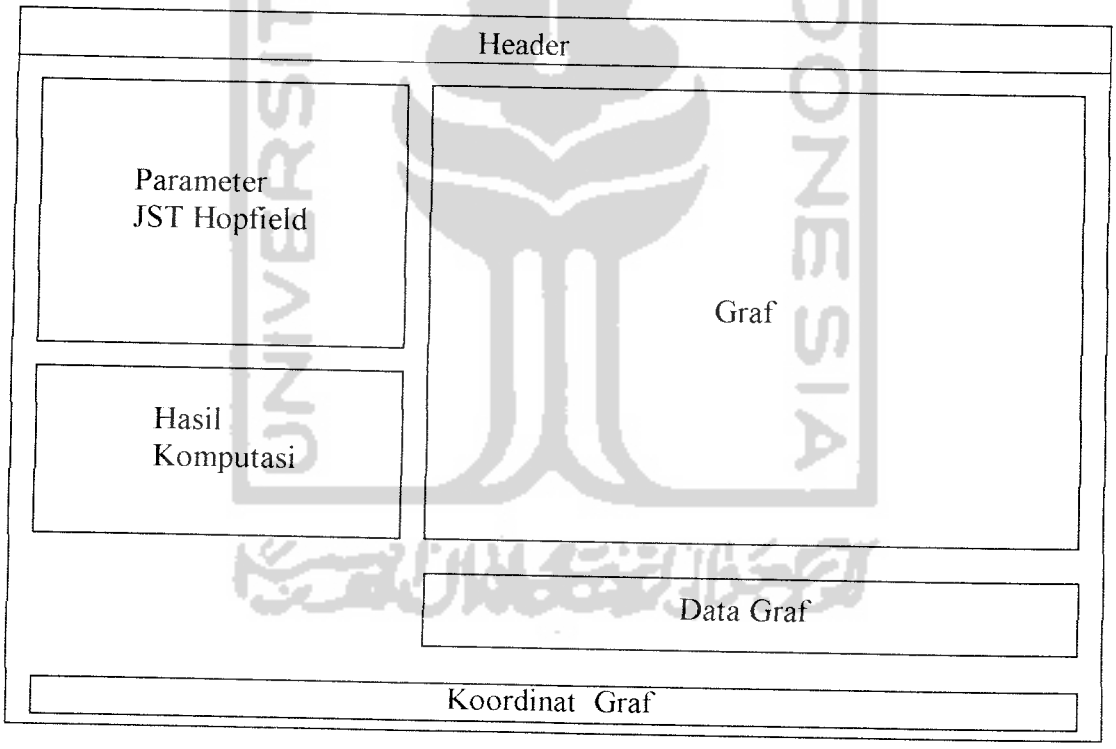


Gambar 3.6 Flow chart Aplikasi Shortest Path Hopfield

3.2.2.1 Perancangan Antar Muka

Rancangan antar muka dari aplikasi Shorthest path using ant colony menggunakan Swing (Java™ Foundation Classes) dengan Notepad++ sebagai editornya.

Gambar 3.7 adalah tampilan dari aplikasi Shortest Path Hopfield. Aplikasi terdiri dari satu menu utama yang terdiri dari header, masukan parameter JST Hopfield, masukan data graf, dan gambar graf yang selain sebagai masukan, juga digunakan untuk menampilkan hasil keluaran. Pada header terdapat opsi



Gambar 3.7 Rancangan Halaman Utama aplikasi Shortest Path Hopfield

Menu dan Bantuan. Pada opsi Menu terdapat pilihan untuk keluar. Pada opsi Bantuan terdapat pilihan untuk melihat menu bantuan untuk menjalankan aplikasi dan menu Tentang Kami.

3.3 Implementasi perangkat lunak

Implementasi merupakan tahap dimana sistem siap dioperasikan pada tahap yang sebenarnya, sehingga akan diketahui apakah sistem yang telah dibuat benar-benar sesuai dengan yang direncanakan. Pada implementasi perangkat lunak ini akan dijelaskan bagaimana program sistem ini bekerja, dengan memberikan tampilan form-form yang dibuat.

3.3.1 Batasan Implementasi

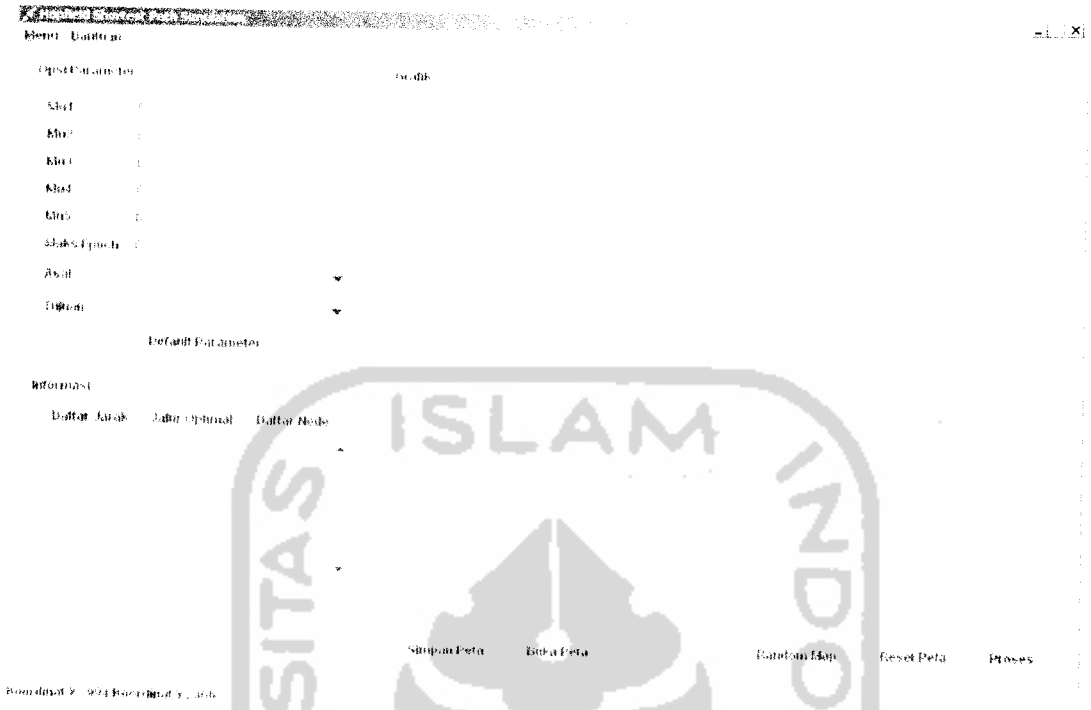
Aplikasi Shortest Path Hopfield untuk pencarian jalur terpendek ini, dalam implementasinya dibatasi hanya pada pencarian jalur terpendek dari graf yang telah di masukkan oleh pengguna.

3.3.2 Implementasi antarmuka

Implementasi dari aplikasi Shortest Path Hopfield ini terdiri dari beberapa form yang memiliki fungsi sendiri-sendiri. Form-form tersebut akan tampil secara berurutan sesuai dengan urutan yang telah terprogram, setelah pengguna melakukan proses tertentu.

3.3.2.1 Halaman utama

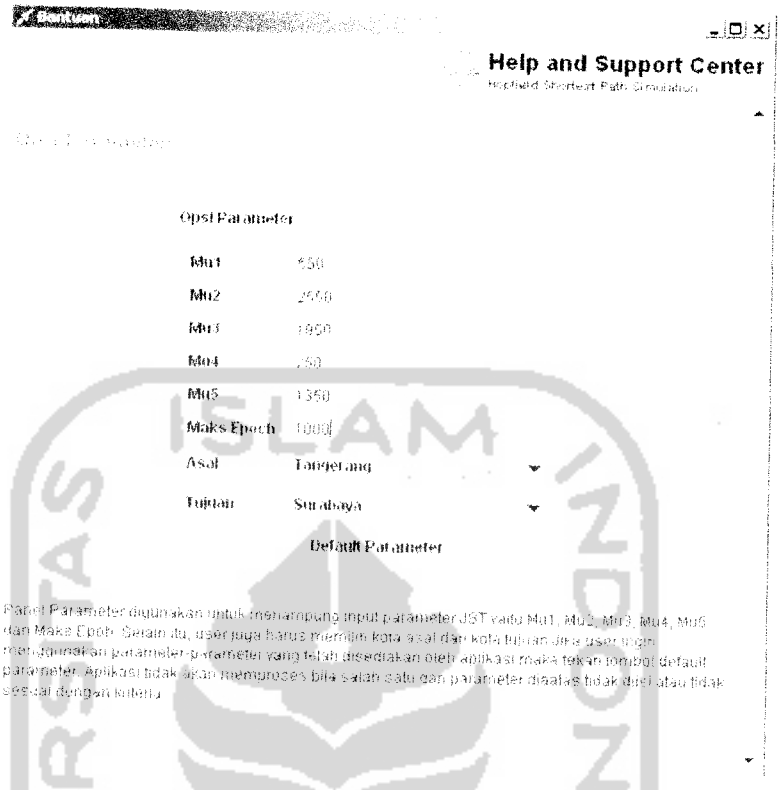
Halaman ini merupakan halaman utama dari aplikasi Shortest Path Hopfield. Pada halaman utama ini terdapat beberapa menu antara lain masukan parameter JST Hopfield, gambar graf dan data graf. Tampilan dari halaman utama dapat dilihat pada gambar 3.8



Gambar 3.8 Halaman Utama Shortest Path Hopfield

3.3.2.2 Halaman Komputasi

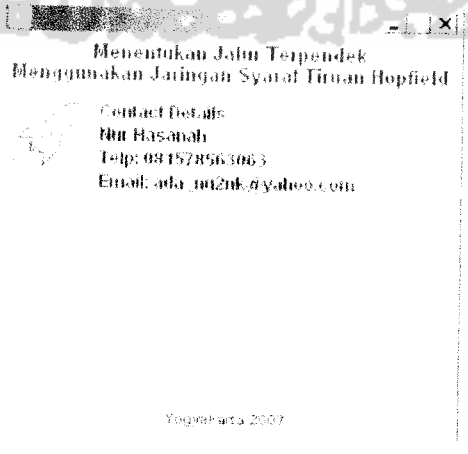
Halaman komputasi digunakan untuk menjalankan dan melihat hasil komputasi dari program aplikasi pencarian jalur terpendek. Implementasi halaman komputasi dapat dilihat pada gambar 3.9.



Gambar 3.10 Halaman Bantuan Shortest Path Hopfield

2. **Halaman Tentang Kami**

Halaman Tentang Kami menjelaskan tentang informasi pembuat aplikasi.. Implementasi halaman Tentang Kami dapat dilihat pada gambar 3.11.



Gambar 3.11 halaman Tentang Kamii Shortest Path Hopfield

3.3.3 Implementasi Prosedural

Implementasi prosedural merupakan implementasi pada pemrograman sistem. Yang akan dijelaskan di sini adalah pemrograman JST Hopfield, yang merupakan proses inti dalam perangkat lunak Shortest Path Hopfield.

1. **Method initializeCost**

Method ini berfungsi untuk inisialisasi matrik C sekaligus matriks rho. Berikut adalah implementasi dari method initializeCost :

a. Pertama, inisalisasi dilakukan dengan memberikan nilai 0 untuk semua nilai jarak 0.

```
for (int i=0; i<node; i++) {
        for (int j=0; j<node; j++) {
        costCity[i][j] = -1.0;
        rho[i][j] = 1.0;
        }
}
```

b. Mengisi elemen matrik rho berdasarkan nilai elemen matrik C, bila ada elemen pada matrik C menyimpan nilai jarak, maka elemen pada matrik rho diisi dengan nilai 0, bila elemen pada matrik C tidak menyimpan nilai jarak, maka elemen pada matrik rho diisi dengan nilai.

```
for (int i=0; i<node; i++) {
       for (int j=0; j<node; j++) {
              if (costCity[i][j] == -1) {
              costCity[i][j] = 1.0;
              rho[i][j] = 1.0;
              } else {
                     costCity[i][j] /= 10000.0;
                     rho[i][j] = 0.0;
              }
       }
}
```

c. Memastikan matrik C dan matrik rho simetris, Jarak A-B = jarak B-A

```
for (int i=0; i<node; i++) {
                    for (int j=0; j<node; j++) {
                          if (costCity[i][j] != 1.0) {
                          costCity[j][i]= costCity[i][j];
                          gamma[j][i] = gamma[i][j];
                          }
                    }
              }
```

2. **Method untuk menghitung jarak antar dua titik**

Method ini berfungsi menghitung jarak antara dua titik (kota). Method ini menerapkan rumus Euclid (*Euclidean distance)*, dimana

jarak = $\sqrt{(x1-x2)^2+(y2-y1)^2}$

```
public double getCostValue(int x1, int y1, int x2, int y2)
{
              double tmpVal;
              tmpVal = Math.pow((x1-x2), 2) + Math.pow((y1-
y2), 2);
              return Math.floor(Math.sqrt(tmpVal));
        }
```

3. **Method initializeValue**

Method ini berfungsi untuk inisialisasi matrik V. Berikut implementasi dari initializeValue:

```
public void initializeValue(int node) {
      U = new double[node][node];
      V = new double[node][node];
      for (int i=0; i<node; i++) {
            for (int j=0; j<node; j++) {
                  U[i][j] = 0.0;
                  V[i][j] = SigmoidFunction(U[i][j]);
            }
      }
}
```

4. **Method Perhitungan Hopfield**

Method ini merupakan inti dari pemrograman Hopfield.

Berikut adalah implementasi dari method Perhitungan Hopfield :

a. Menghitung Myu1 untuk mendapatkan nilai input U_{xi} untuk mengupdate V_{xi}.

```
public double Ali_Mu1_Term(int x, int i, int s, int d)
{
        return( costCity[x][i] * (1.0 - kroneckerDelta(x,d)
*                   kroneckerDelta(i,s)) );
}
```

b. Menghitung Myu2 untuk mendapatkan nilai input U_{xi} untuk mengupdate V_{xi}.

```
public double Ali_Mu2_Term(int x, int i, int s, int d)
{
        return (rho[x][i] * (1.0 -

        kroneckerDelta(x,d)*kroneckerDelta(i,s)));
}
```

c. Menghitung Myu3 pertama untuk mendapatkan nilai input U_{xi} untuk mengupdate V_{xi}.

```
double Ali_Mu3_first_Term(int x, int Node)
{
        int y;
        double Value=0.0;
        for(y=0 ; y<Node ; y++){
                if(y!=x){
                Value+=(V[x][y] - V[y][x]);
                }
        }
                return Value;
}
```

d. Menghitung Myu3 kedua untuk mendapatkan nilai input U_{xi} untuk mengupdate V_{xi}.

```
double Ali_Mu3_second_Term(int i, int Node)
{
        int y;
        double Value=0.0;
                for(y=0 ; y<Node ; y++){
                        if(y!=i){
                        Value+=(V[i][y] - V[y][i]);
                        }
                }
                return(Value);
}
```

e. Menghitung Myu4 untuk mendapatkan nilai input U_{xi} untuk mengupdate V_{xi}.

```
double Ali_Mu4_Term(int x, int i)
{
        return( 1.0 - 2.0*V[x][i] );
}
```

f. Menghitung Myu5 untuk mendapatkan nilai input U_{xi} untuk mengupdate V_{xi}.

```
double Ali_Mu5_Term(int x, int i, int s, int d)
{
        return( kroneckerDelta(x,d)*kroneckerDelta(i,s) );
}
```

g. Fungsi Delta Kronecker. Fungsi ini digunakan oleh fungsi-fungsi myu. Tujuan dari fungsi ini adalah untuk memastikan apakah suatu elemen matrik mempunyai nilai baris dan kolom yg sama atau tidak. Jika iya, maka Delta Kronecker bernilai 1. Jika tidak, maka bernilai 0.

```
public double kroneckerDelta(int s, int d) {
        if (s == d) {
                return 1.0;
        } else {
                return 0.0;
        }
}
```

h. Menghitung fungsi Sigmoid. Fungsi ini berguna untuk memperoleh nilai V_{xi} baru. V_{xi} menunjukkan suatu jalur antar kota.

```
public double SigmoidFunction(double in) {
        double sigmoid;
        sigmoid = 1/(1+Math.exp((-1 * LAMBDA) * in));
        return sigmoid;
}
```

i. Untuk memperoleh nilai U_{xi} yang baru. U_{xi} merupakan bobot yang digunakan untuk memperoleh V_{xi}.

```
double updateFunction(int x, int i, int s, int d, int
Node){
double Value=0.0;
      if(x==d && i==s){
      Value = U[x][i] + DELTA*( -U[x][i]/(double)TAU
                -0.5*myu1*Ali_Mu1_Term(x,i,s,d)
                -0.5*myu2*Ali_Mu2_Term(x,i,s,d)
                -myu3*Ali_Mu3_first_Term(x,Node)
                +myu3*Ali_Mu3_second_Term(i,Node)
                -0.5*myu4*Ali_Mu4_Term(x,i)
                +0.5*myu5*Ali_Mu5_Term(x,i,s,d)
                +0.5*myu5
                -0.5*myu4);
      }
    else{
      Value = U[x][i] + DELTA*( -U[x][i]/(double)TAU
                -0.5*myu1*Ali_Mu1_Term(x,i,s,d)
                -0.5*myu2*Ali_Mu2_Term(x,i,s,d)
                -myu3*Ali_Mu3_first_Term(x,Node)
                +myu3*Ali_Mu3_second_Term(i,Node)
                -0.5*myu4*Ali_Mu4_Term(x,i)
                +0.5*myu5*Ali_Mu5_Term(x,i,s,d)
                -0.5*myu1*costCity[x][i]
                -0.5*myu2*rho[x][i]
                -0.5*myu4);
      }
      return(Value);
}
```

5. **Method Untuk Menampilkan Hasil Perhitungan Hopfield**

a. Mencetak kota-kota yang dilewati jalur terpendek.

```
for (int j=0; j<N; j++) {
```

b. Menampilkan matrik V. Matrik V menunjukkan output jalur terpendek.

```
for (int i=0; i<NCITY; i++) {
        for (int j=0; j<NCITY; j++) {
                if (V[i][j] > 0.5) {
                System.out.print(i + "    ");
                }
                else {
                System.out.print(0 + "    ");
                }
        }
        System.out.println("");
}
```

BAB IV

HASIL DAN PEMBAHASAN

4.1 Pengujian Program

Pada tahap analisis kinerja perangkat lunak dijelaskan tentang pengujian aplikasi pencarian jalur terpendek menggunakan JST Hopfield. Pengujian dilakukan dengan kompleks dan diharapkan dapat diketahui kekurangan-kekurangan dari sistem untuk kemudian diperbaiki sehingga kesalahan dari sistem dapat diminimalisasi atau bahkan dihilangkan. Pengujian sistem ini dilakukan untuk mendapatkan hasil yang akurat.

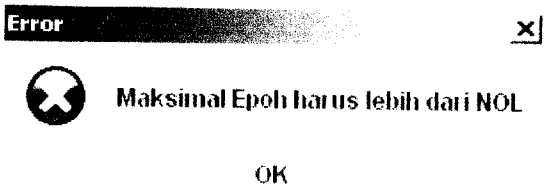
Pengujian sistem ini dapat dilakukan dengan mengisi *form* inputan yang telah ditampilkan pada BAB III, yaitu dengan mengisi data parameter JST Hopfield dan data graf.

4.1.1 Pengujian tidak normal

Dilakukan dengan memberikan masukan dengan spesifikasi yang tidak diijinkan sehingga sistem akan memberikan reaksi lain. Reaksi sistem berupa berupa peringatan *(alert)* atau penanganan kesalahan *(error handling)*.

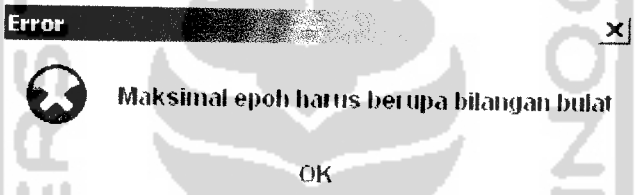
Penanganan kesalahan ini dilakukan untuk menangkap error yang terjadi ketika salah satu field pada form inputan kosong atau ketidaksesuaian tipe data. Contoh penanganan kesalahan input terdapat pada pemasukan *field* yang

membutuhkan data jumlah max epoh. Jika pada *field* maksimal epoh diisikan nilai nol atau kurang dari nol, maka akan muncul peringatan seperti pada gambar 4.1.



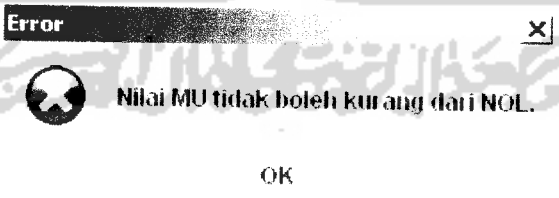
Gambar 4.1 Peringatan jika field maksimal epoh diisi dengan nol atau kurang dari nol

Jika pada *field* max epoh diisikan nilai yang bukan bilangan bulat, maka akan muncul peringatan seperti pada gambar 4.2.



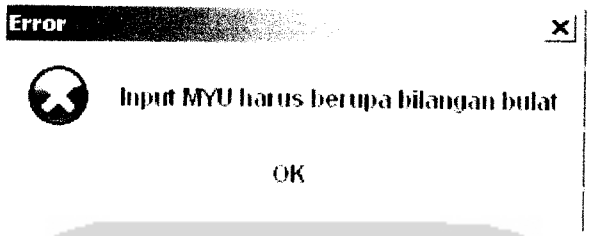
Gambar 4.2 Peringatan jika field maksimal epoh diisi nilai yang bukan bilangan bulat

Jika pada salah satu *field* myu diisikan nilai nol atau kurang dari nol, maka akan muncul peringatan seperti pada gambar 4.3.



Gambar 4.3 Peringatan jika field myu jika diisi dengan nol atau kurang dari nol

Jika pada salah satu *field* myu diisikan nilai yang bukan bilangan bulat, maka akan muncul peringatan seperti pada gambar 4.4.



Gambar 4.4 Peringatan jika field myu jika diisi nilai yang bukan bilangan bulat

4.2 Analisis Kinerja Sistem

4.2.1 Analisis berdasarkan jumlah titik

Dilakukan dengan memberikan masukan yang menurut spesifikasi awal dan pengetahuan yang diijinkan. Setelah diberikan masukan yang sesuai, dilakukan analisis pembandingan antara kebenaran masukan serta kesesuaian program dengan kebutuhan sistem. Pengujian dilakukan dengan masukan data kecil, data menengah, dan data besar. Data kota yang dimasukkan berupa data random. Parameter JST Hopfield yang digunakan adalah :

$\mu 1 = 550$

$\mu 2 = 2550$

$\mu 3 = 1950$

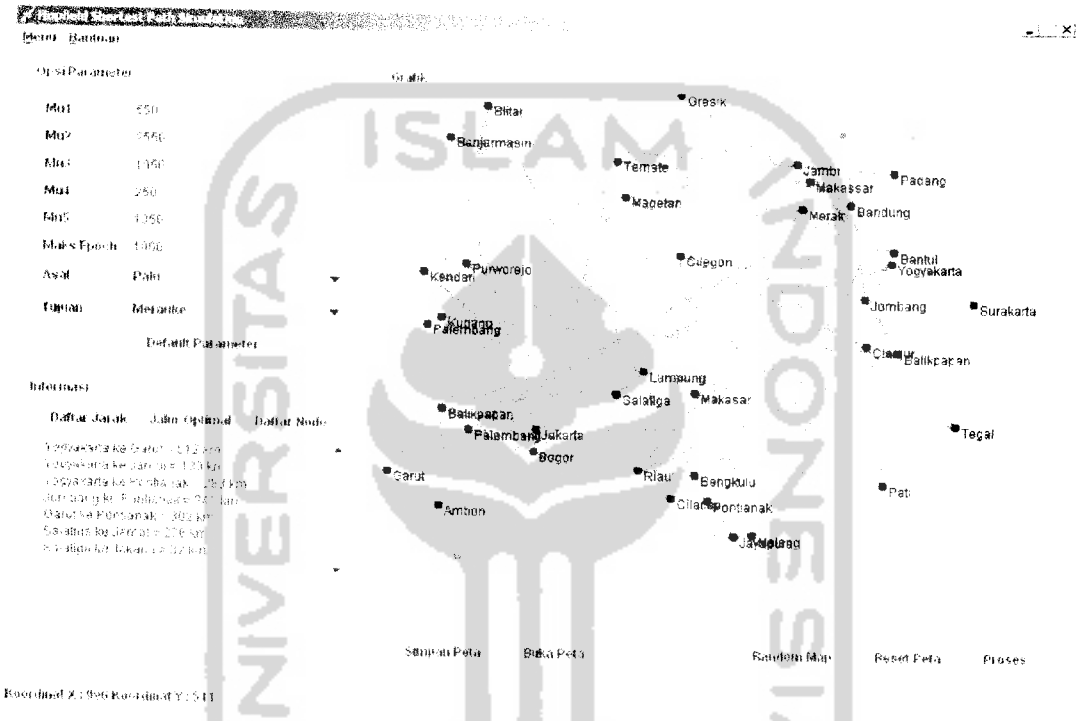
$\mu 4 = 250$

$\mu 5 = 1350$

Max Epoh = 1000, untuk data kecil dan data sedang.

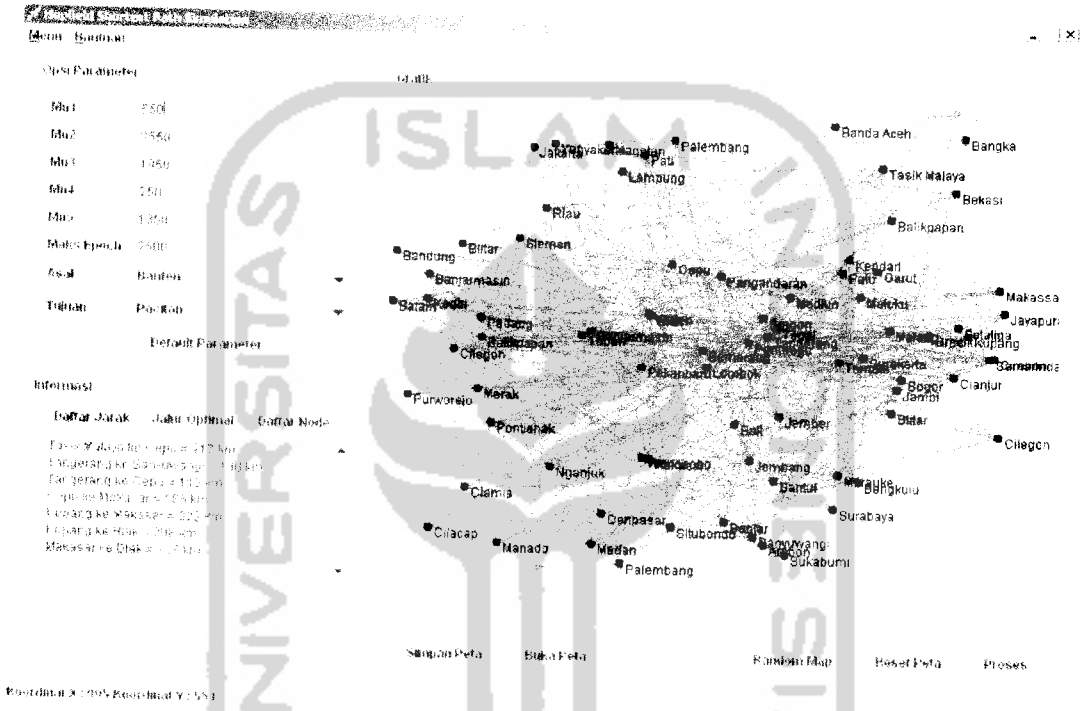
Max Epoh = 2500, untuk data besar.

Data graf sedang ditunjukkan oleh gambar 4.6. Pada graf ini, terdapat 40 buah kota. Parameter JST Hopfield yang dimasukkan adalah parameter default. Kota asal yang dipilih adalah Palu dan kota tujuan yang dipilih adalah Merauke. Jalur terpendeknya adalah Palu – Jombang – Merauke.



Gambar 4.6 Data Sedang

Data graf besar ditunjukkan oleh gambar 4.7. Pada graf ini, terdapat 80 buah kota. Parameter JST Hopfield yang dimasukkan adalah parameter default. Kota asal yang dipilih adalah Banten dan kota tujuan yang dipilih adalah Pacitan. Jalur terpendeknya adalah Banten – Kendari – Pacitan.



Gambar 4.7 Data Besar

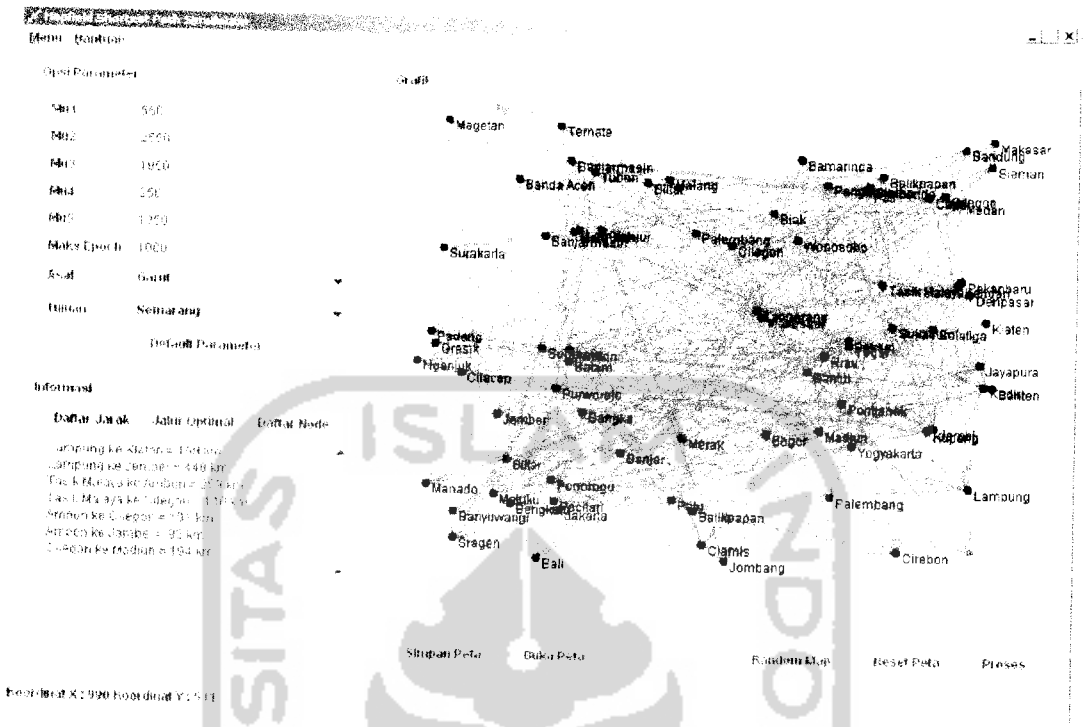
Total epoh yang dibutuhkan data kecil dan sedang adalah 1000 epoh, sedangkan total epoh yang dibutuhkan data besar adalah 2500 epoh. Dapat dilihat, Jumlah epoh mempengaruhi lamanya aplikasi dalam mencari jalur terpendek, hal ini dapat dilihat pada progres bar di kanan bawah aplikasi bahwa pada kasus kota kecil dan kota sedang jalur terpendek waktu yang dibutuhkan lebih sedikit dibandingkan pada kasus kota besar, karena total epoh yang dibutuhkan data besar lebih banyak dibandingkan dengan data kecil maupun data sedang. Jadi dapat

disimpulkan bahwa semakin besar jumlah kota, maka total epoh yang dibutuhkan lebih banyak, sehingga total waktu yang dibutuhkan juga akan lebih lama.

4.2.2 Analisis berdasarkan nilai parameter

Dalam menentukan jalur terpendek menggunakan JST Hopfield, ada parameter – parameter yang tentu saja berpengaruh terhadap hasil perhitungan. Mengacu pada penelitian sebelumnya (Ali dan Khamoun, 1993) ditentukan nilai–nilai parameter yang digunakan adalah $\mu 1$=550, $\mu 2$=2550, $\mu 3$=1950, $\mu 4$=250, dan $\mu 5$=1350. Kombinasi parameter ini ditetapkan sebagai parameter *default*.

Berikut akan diuji 10 kombinasi parameter $\mu 1, \mu 2, \mu 3, \mu 4, \mu 5$ untuk sebuah data besar, sebanyak 80 kota. Gambar 4.5 Menunjukkan graf awal saat belum diproses. Data kota didapatkan secara random. Pada kasus ini akan dicari jalur terpendek dari Cianjur ke Yogyakarta dengan parameter-parameter yang berbeda, termasuk parameter *default*. Kombinasi parameter didapatkan secara random. Sedangkan maksimum epoh yang digunakan adalah 2500 epoh.



Gambar 4.8 Graf yang digunakan untuk pengujian

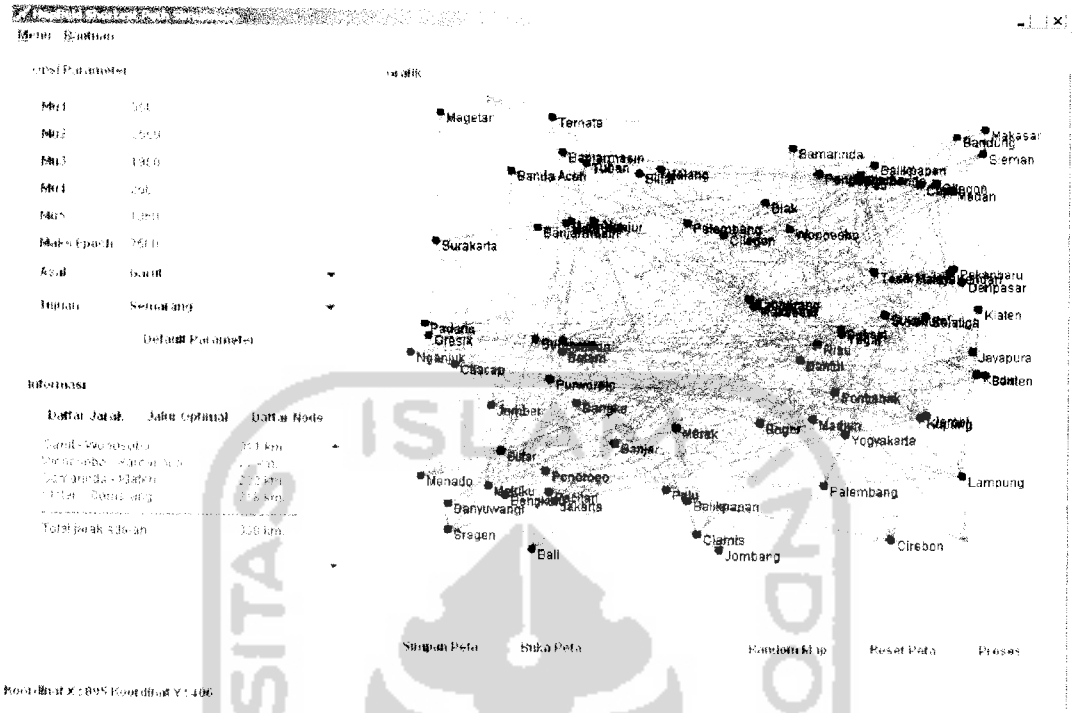
Setelah dicoba 10 macam kombinasi parameter $\mu 1, \mu 2, \mu 3, \mu 4, \mu 5$, maka didapatkan berbagai macam hasil jalur terpendek. Hasil ini dapat dilihat pada tabel 4.1.

Tabel 4.1 Macam-macam kombinasi parameter Myu beserta hasil jalur terpendek dan total jaraknya

No	Parameter Myu	Jalur Terpendek	Total Jarak
1	$\mu 1$ =300, $\mu 2$ =2550, $\mu 3$ =2900, $\mu 4$ =575, $\mu 5$ =2200	Garut – Wonosobo – Samarinda – Banjar – Jakarta – Pacitan – Lampung – Tasik Malaya – Ambon – Cilegon – Madiun – Blitar – Situbondo – Kupang – Cirebon – Kediri – Klaten – Semarang	3682 km
2	$\mu 1$ =250, $\mu 2$ =2550, $\mu 3$ =3050, $\mu 4$ =450, $\mu 5$ =1700	Garut – Wonosobo – Samarinda – Banjar – Jakarta – Pacitan – Lampung – Tasik Malaya – Ambon – Cilegon – Madiun – Blitar – Situbondo – Kupang – Cirebon – Kediri – Klaten – Semarang	3682 km
3	$\mu 1$ =100, $\mu 2$ =4000, $\mu 3$ =3500, $\mu 4$ =455, $\mu 5$ =2050	Garut – Wonosobo – Pontianak – Bogor – Tangerang – Pekanbaru – Tasik Malaya – Sragen – Bangka – Kupang – Cirebon –	2474 km

		Kediri – Klaten – Semarang	
4	$\mu 1$=350, $\mu 2$=2300, $\mu 3$=3400, $\mu 4$=500, $\mu 5$=2550	Garut – Wonosobo – Samarinda – Banjar – Jakarta – Pacitan – Lampung – Klaten – Semarang	1566 km
5	$\mu 1$=600, $\mu 2$=1200, $\mu 3$=3000, $\mu 4$=400, $\mu 5$=4000	Garut – Wonosobo – Samarinda – Banjar – Jakarta – Pacitan – Lampung – Klaten – Semarang	1566 km
6	$\mu 1$=500, $\mu 2$=1500, $\mu 3$=2000, $\mu 4$=200, $\mu 5$=2000	Garut – Wonosobo – Samarinda – Banjar – Jakarta – Pacitan – Lampung – Klaten – Semarang	1566 km
7	$\mu 1$=250, $\mu 2$=2000, $\mu 3$=2750, $\mu 4$=500, $\mu 5$=1950	Garut – Wonosobo – Cirebon – Kediri – Klaten – Semarang	1078 km
8	$\mu 1$=650, $\mu 2$=1550, $\mu 3$=2800, $\mu 4$=550, $\mu 5$=750	Garut – Wonosobo – Pontianak – Bogor – Tangerang – Klaten – Semarang	1100 km
9	$\mu 1$=175, $\mu 2$=2500, $\mu 3$=3000, $\mu 4$=350, $\mu 5$=2100	Garut – Wonosobo – Samarinda – Tangerang – Klaten – Semarang	969 km
10	$\mu 1$=550, $\mu 2$=2550 $\mu 3$=1950, $\mu 4$=250, $\mu 5$=1350	Garut – Wonosobo – Samarinda – Klaten – Semarang	838 km

Dari pengujian data besar dengan 10 kombinasi parameter myu, dapat disimpulkan bahwa kombinasi parameter nomor 10 merupakan yang terbaik dengan susunan $\mu 1$=550, $\mu 2$=2550, $\mu 3$=1950, $\mu 4$=250, $\mu 5$=1350, dengan jalur terpendek: Makassar – Ciamis – Semarang – Lampung – Bandung dengan total 838 km. Gambar 4.9 menunjukkan hasil jalur terpendek yang terbaik dari 10 kombinasi parameter yang telah dicoba.



Gambar 4.9 Jalur Terpendek Graf pengujian

BAB V

KESIMPULAN DAN SARAN

5.1 Simpulan

Berdasarkan hasil penelitian dan pembahasan yang telah dilakukan, dapat disimpulkan bahwa:

1. Jaringan saraf tiruan Hopfield dapat digunakan untuk memecahkan masalah jalur terpendek dengan memberikan solusi optimum yang tepat.
2. Nilai parameter untuk tingkat akurasi yang terbaik adalah $\mu 1 = 550$, $\mu 2 = 2550$, $\mu 3 = 1950$, $\mu 4 = 250$ dan $\mu 5 = 1350$.

5.2 Saran

Mengingat berbagai keterbatasan yang dialami penulis terutama masalah pemikiran dan waktu, maka penulis menyarankan untuk pengembangan penelitian dimasa yang akan datang sebagai berikut :

1. Sistem dapat dikembangkan lagi dengan memodifikasi layout peta yang sederhana menjadi layout yang mempunyai garis bujur dan garis lintang.
2. Penelitian dapat dilanjutkan dengan menambahkan jenis parameter jaringan saraf tiruan Hopfield, yaitu myu, untuk meningkatkan kecepatan dan keakuratan hasil.

3. Penelitian dapat dilanjutkan dengan menggunakan perpaduan cabang ilmu lain seperti FEA (Fuzzy Evolutionary Algorithm).