PENERAPAN METODE DEMPSTER SHAFER UNTUK DETEKSI PENYAKIT INFEKSI SALURAN PERNAFASAN DENGAN PEMBANGKITAN NILAI *BELIEF* MENGGUNAKAN ALGORITMA GENETIKA

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Noviani Hasianna Panjaitan
NIM: 115060807111086

PROGRAM STUDI INFORMATIKA
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2016

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

PENGESAHAN

PENERAPAN METODE DEMPSTER SHAFER UNTUK DETEKSI PENYAKIT INFEKSI SALURAN PERNAFASAN DENGAN PEMBANGKITAN NILAI *BELIEF* MENGGUNAKAN ALGORITMA GENETIKA

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh :
Noviani Hasianna Panjaitan
NIM: 115060807111086

Skripsi ini telah diuji dan dinyatakan lulus pada
19 Mei 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Rekyan Regasari MP, ST.,MT.	Drs. Marji,. MT
NIK: 770414 06 1 2 0253	NIP. 19670801 199203 1 001

Mengetahui
Ketua Program Studi Informatika/Ilmu Komputer

Issa Arwani, S.Kom, M.Sc
NIP: 19830922 201212 1 003

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 20 Mei 2016

Noviani Hasianna Panjaitan

NIM: 115060807111086

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Tuhan Yang Maha Esa karena hanya dengan rahmat dan karunia-Nya penulis dapat menyelesaikan skripsi dengan judul "Penerapan Metode Dempster Shafer Untuk Mendeteksi Penyakit Infeksi Saluran Pernafasan Dengan Pembangkitan Nilai *Belief* Menggunakan Algoritma Genetika".

Skripsi ini diajukan untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Komputer.

Melalui kesempatan ini, penulis ingin menyampaikan rasa hormat dan terima kasih yang sebesar-besarnya kepada semua pihak yang telah memberikan bantuan dan dukungan selama penulisan skripsi, diantaranya:

1. Rekyan Regasari MP., S.T., M.T. selaku dosen pembimbing I dan Drs. Marji, M.T., selaku dosen pembimbing II yang telah memberikan ilmu dan saran untuk menyelesaikan skripsi ini.
2. Ir. Sutrisno, MT. Sebagai Dekan Fakultas Ilmu Komputer Universitas Brawijaya.
3. Drs. Marji, M.T., selaku Ketua Program Studi Informatika / Ilmu Komputer Fakultas Ilmu Komputer (FILKOM) Universitas Brawijaya.
4. Aswin Suharsono, S.T, M.T selaku dosen pembimbing akademik yang telah banyak memberikan saran dan motivasi selama masa perkuliahan.
5. Bapak Hasan Panjaitan, Ibu Maria Immaculata M. T., Kakak Lirania Nauli, Adik Theresia Lovian Nauli, Samuel Geraldo, dan seluruh keluarga besar atas nasehat, kasih sayang, perhatian, kesabaran, semangat dan doa yang diberikan kepada penulis.
6. dr. Triwahju Astuti M.Kes, Sp.P(K) dan dr. Robby Soeyanto selaku pakar yang telah membantu proses pengumpulan data skripsi.
7. Seluruh dosen Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya atas kesediaan mendidik dan membagi ilmunya kepada penulis.
8. Seluruh staf dan karyawan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya yang telah membantu kelancaran pengerjaan skripsi.
9. Sahabat-sahabat penulis yaitu Firda Ika, Betty Anggoro, Sari Kusuma, dan Yekti Narika yang telah memberikan semangat kepada penulis selama menempuh studi di Teknik Informatika Fakultas Ilmu komputer Universitas Brawijaya.
10. Teman-teman kelas TIF J yang tidak bisa disebutkan satu persatu yang selalu mendukung saya.
11. Teman-teman kos TBW 2 yaitu Jojor Jennifer, Della Zagita, Mawarti, Ayu Kusumaningtyas, Febbe Cahya, Hikmah, Erida, dan Natasha atas dukungan dan semangat kepada penulis selama berada di Malang.
12. Teman-teman penulis, yaitu Dedi Harianja, Santo Sidauruk, Iunike Kartika, dan teman-teman PMK Daniel 2011, 2012, 2013 yang memberi dukungan dan semangat selama di Malang.

13. Semua pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat baik secara langsung maupun yang tidak langsung demi terselenggaranya skripsi ini.

Hanya doa yang bisa penulis berikan semoga Tuhan Yang Maha Esa memberikan pahala serta balasan kebaikan yang berlipat. Penulis menyadari bahwa skripsi ini masih banyak kekurangan. Oleh karena itu, kritik dan saran yang bersifat membangun sangat diharapkan untuk menyempurnakan skripsi ini. Semoga skripsi ini membawa manfaat bagi semua pihak yang menggunakannya.

Malang, 21 Maret 2016

Penulis

noviani.hasianna@gmail.com

ABSTRAK

Noviani Hasianna Panjaitan, 2016 : Penerapan Metode Dempster Shafer Untuk Deteksi Penyakit Infeksi Saluran Pernafasan Dengan Pembangkitan Nilai *Belief* Menggunakan Algoritma Genetika. Skripsi Program Studi Informatika / Ilmu Komputer, Fakultas Ilmu Komputer, Universitas Brawijaya, Malang.

Dosen Pembimbing : Rekyan Regasari MP., S.T., M.T.;
Drs. Marji,. MT.

Penyakit pneumonia yang merupakan penyakit infeksi saluran pernafasan bagian bawah termasuk dalam 10 jenis penyakit paling sering menyebabkan kematian di Indonesia. Tidak mudah untuk mendeteksi penyakit dengan cepat dan tepat, terutama bila gejala yang dialami pasien tidak jelas atau terlalu umum seperti pada penyakit pneumonia dan bronchitis akut dengan gejala yang terlalu umum serta memiliki kemiripan gejala yang dapat memungkinkan dokter salah mendiagnosis padahal penanganan terhadap kedua penyakit ini berbeda. Metode Dempster-Shafer telah banyak digunakan untuk menyelesaikan masalah diagnosa penyakit dengan nilai *belief* yang diberikan pakar, namun tidak dapat dipastikan kredibilitas nilai *belief* yang diberikan pakar tersebut. Diperlukan optimasi pada nilai *belief* agar didapatkan nilai *belief* yang konsisten dan optimal. Pada penelitian ini menggunakan algoritma genetika untuk optimasi nilai *belief* karena algoritma genetika merupakan komputasi sederhana namun kuat dalam pencarian untuk perbaikan. Metode yang digunakan dalam algoritma genetika yaitu Extended Internediate Crossover dan Random Mutation, dan Elitism Selection. Hasil yang didapatkan nantinya adalah kromosom dengan nilai *fitness* mendekati paling optimal, kemudian kromosom tersebut akan digunakan sebagai nilai *belief* untuk menghitung gejala-gejala dari penyakit dengan metode Dempster-Shafer.

Berdasarkan hasil pengujian dan analisis sistem parameter algoritma genetika yang baik untuk menentukan nilai *belief* yaitu probabilitas *crossover*(cr) 0,4, probabilitas mutasi(mr) 0,6, ukuran populasi 120, dan jumlah generasi 100. Hasil perhitungan akurasi sistem dengan menggunakan nilai *belief* yang telah dioptimasi pada 17 kasus penyakit infeksi saluran pernafasan menunjukkan akurasi sebesar 82,35%. Hasil akurasi ini mengalami kenaikan sebesar 29,41% dimana pada perhitungan akurasi menggunakan nilai *belief* dari pakar menunjukkan akurasi sebesar 52,94%.

Kata kunci : Algoritma Genetika, Optimasi Nilai Densitas, Dempster-Shafer, Penyakit Infeksi Saluran Pernafasan

ABSTRACT

Noviani Hasianna Panjaitan, 2016 : *Implementation of Dempster – Shafer Method to Detect Diseases Respiratory Tract Infection With Optimization Belief Value Using Genetic Algorithm. Undergraduate Thesis of Informatic / Computer Science Study Program, Computer Science Faculty, Brawijaya University*, Malang

Dosen Pembimbing : Rekyan Regasari MP., S.T., M.T.;
Drs. Marji,. MT.

Pneumonia is an infectious lower respiratory tract disease that included in 10 types of diseases most often leads to death in Indonesia. It's not easy to detect diseases quickly and accurately, especially when the symptoms experienced by patients is unclear or too general such as pneumonia and acute bronchitis with symptoms that are too general and have similar symptoms that make possible doctor diagosing incorrectly beside the handling of these two diseases totally different, Dempster-Shafer method has been widely used to solve the problem of diagnosis of diseases with belief values given expert, but can not be ascertained credibility of belief given the expert. Required optimization on the belief value in order to get the consistent and optimal of belief value. In this research uses genetic algorithms to optimize the belief value because of the genetic algorithm is simple but powerful computing in the search for improvement. The method used in the genetic algorithm is Extended Internediate Crossover, Random Mutation, and Elitism Selection. The results obtained later is chromosomes with the most optimal fitness value approach, then the chromosomes will be used as the belief value to calculate the symptoms of the disease with Dempster-Shafer method.

Based on the results of examination and systems analysis of parameter of Genetic Algorithm which is good to determine the value of the density, namely the probability of crossover (cr) 0.4, the probability of mutation (mr) 0.6, the size of a population of 120, and the number of generations of 100. The results of average accuracy the system to use the Belief value that has optimized on 17 cases of respiratory tract infections showed an accuracy of 82,35%. The result of this accuracy increased by 29,41% where the accuracy of the calculation using the value of belief of experts showed an accuracy of 52.94%.

Keywords: *Dempster Shafer, Genetic Algorithm, Pneumonia, Belief value*

DAFTAR ISI

PENGESAHAN ii
PERNYATAAN ORISINALITAS iii
KATA PENGANTAR iv
ABSTRAK vi
ABSTRACT vii
DAFTAR ISI viii
DAFTAR TABEL xi
DAFTAR GAMBAR xiii
DAFTAR *SOURCE CODE* xiv
DAFTAR LAMPIRAN xv
BAB 1 PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan masalah 3
1.3 Tujuan 3
1.4 Manfaat 3
1.5 Batasan masalah 4
1.6 Sistematika pembahasan 4
BAB 2 TINJAUaN PUSTAKA 5
2.1 Kajian Pustaka 5
2.2 Penyakit Infeksi Saluran Pernafasan 6
2.2.1 Pneumonia 8
2.2.2 Bronkitis Akut 8
2.2.3 URTI (Upper Respiratory Tract Infection) 9
2.3 Metode Dempster Shafer 9
2.4 Algoritma Genetika 10
2.4.1 Representasi Kromosom (*Chromosome*) 11
2.4.2 Inisialisasi 12
2.4.3 Reproduksi 12
2.4.4 Evaluasi 14
2.4.5 Seleksi 15

BAB 3 METODOLOGI PENELITIAN 19
3.1 Studi Literatur 19
3.2 Pengumpulan Data 20
3.3 Analisis Kebutuhan 21
3.4 Perancangan Aplikasi 22
3.5 Implementasi Aplikasi 23
3.6 Pengujian Aplikasi 23
3.7 Pengambilan Kesimpulan 26
BAB 4 PERANCANGAN 27
4.1 Perancangan Sistem 27
4.1.1 Proses Pembangkitan Nilai *Belief* 28
4.1.2 Inisialisasi Populasi Awal 29
4.1.3 Crossover 29
4.1.4 Mutasi 30
4.1.5 Menghitung Nilai *Fitness* 31
4.1.6 Perhitungan Dempster Shafer 33
4.1.7 Seleksi 34
4.1.8 Kondisi Berhenti 35
4.2 Perhitungan Manual 36
4.2.1 Pembangkitan Awal Nilai *Belief* Dengan Algoritma Genetika 36
4.2.2 Perhitungan Dempster Shafer 46
4.3 Perancangan Antarmuka 49
BAB 5 IMPLEMENTASI APLIKASI 51
5.1 Spesifikasi Aplikasi 51
5.1.1 Spesifikasi Perangkat Keras 51
5.1.2 Spesifikasi Perangkat Lunak 51
5.2 Implementasi Program 52
5.2.1 Inisialisasi Populasi Awal 52
5.2.2 Implementasi *Crossover* 53
5.2.3 Implementasi Mutasi 54
5.2.4 Perhitungan Nilai *Fitness* dan Metode *Dempster Shafer* 56
5.2.5 Implementasi Seleksi 62

5.3 Implementasi Antarmuka 63
BAB 6 PENGUJIAN DAN ANALISIS 66
6.1 Pengujian Parameter Algoritma Genetika 66
6.1.1 Pengujian Jumlah Populasi (*Popsize*) Terhadap Nilai *Fitness* 66
6.1.2 Pengujian Jumlah Generasi Terhadap Nilai *Fitness* 67
6.1.3 Pengujian Probabilitas Crossover dan Probabilitas Mutasi Terhadap Nilai *Fitness* 68
6.2 Pengujian Nilai *Belief* Menggunakan Parameter Algoritma Genetika. 70
6.3 Nilai *Belief* Optimal 71
6.4 Pengujian Akurasi 71
BAB 7 PENUTUP 74
7.1 Kesimpulan 74
7.2 Saran 74
DAFTAR PUSTAKA 75
LAMPIRAN A DAFTAR NILAI *BELIEF* DARI PAKAR 77
LAMPIRAN B Daftar gejala 78
LAMPIRAN C Data latih 79
LAMPIRAN D Data UJI 83

DAFTAR TABEL

Tabel 2.1 Tabel Gejala Penyakit Infeksi Saluran Pernafasan 7

Tabel 2.2 Contoh populasi untuk metode *Replacement Selection* 15

Tabel 2.3 Contoh *offspring* yang dihasilkan setelah proses *crossover* dan mutasi 16

Tabel 2.4 Contoh hasil seleksi dengan menggunakan metode *Replacement Selection* 16

Tabel 2.5 Contoh populasi untuk metode *Tournament Selection* 17

Tabel 2.6 Contoh hasil seleksi menggunakan metode *Tournament Selection* 17

Tabel 2.7 Contoh populasi untuk metode *Elitism Selection* 17

Tabel 2.8 Hasil perankingan semua individu 18

Tabel 2.9 Hasil seleksi menggunakan metode *Elitism Selection* 18

Tabel 3.1 Akusisi pengetahuan 20

Tabel 3.2 Penjelasan gejala 20

Tabel 3.3 Skenario uji coba populasi 24

Tabel 3.4 Skenario uji coba generasi 24

Tabel 3.5 Skenario uji coba kombinasi *Crossover rate* dan *Mutation rate* 25

Tabel 3.6 Tabel pengujian akurasi 25

Tabel 4.1 Data uji perhitungan manual 36

Tabel 4.2 Inisialisasi populasi awal 37

Tabel 4.3 Aturan kombinasi untuk m_3 39

Tabel 4.4 Aturan kombinasi untuk m_5 39

Tabel 4.5 Aturan kombinasi m_7 40

Tabel 4.6 Hasil Akurasi Individu 1 41

Tabel 4.7 Hasil perhitungan nilai *fitness* 42

Tabel 4.8 Perhitungan *Crossover* 42

Tabel 4.9 Tabel nilai alpha 43

Tabel 4.10 Perhitungan Mutasi 44

Tabel 4.11 Himpunan populasi yang menjadi parent 45

Tabel 4.12 Offspring beserta parent dan *fitness*nya 45

Tabel 4.13 Individu baru 46

Tabel 4.14 Aturan Kombinasi untuk m_3 46
Tabel 4.15 Aturan Kombinasi untuk m_5 47
Tabel 4.16 Aturan kombinasi untuk m_7 47
Tabel 4.17 Aturan kombinasi untuk m_9 48
Tabel 5.1 Spesifikasi perangkat keras 51
Tabel 5.2 Spesifikasi perangkat lunak 51
Tabel 6.1 Hasil pengujian jumlah populasi 66
Tabel 6.2 Pengujian Jumlah Generasi 67
Tabel 6.3 Hasil pengujian kombinasi cr dan mr 68
Tabel 6.4 Pengujian Akurasi 70
Tabel 6.5 Nilai *belief* optimal 71
Tabel 6.6 Hasil Perbandingan Diagnosa 72

DAFTAR GAMBAR

Gambar 3.1 Tahapan penelitian 19
Gambar 3.2 Diagram blok desain arsitektur sistem 22
Gambar 4.1 Diagram alir perancangan sistem 27
Gambar 4.2 Diagram alir pembangkitan nilai *belief* 28
Gambar 4.3 Diagram alir proses inisialisasi populasi awal 29
Gambar 4.4 Diagram alir proses crossover 30
Gambar 4.5 Diagram alir proses mutasi 31
Gambar 4.6 Diagram alir proses perhitungan nilai *fitness* 32
Gambar 4.7 Diagram alir algoritma perhitungan *Dempster-Shafer* 34
Gambar 4.8 Diagram alir proses seleksi 35
Gambar 4.9 Inisialisasi kromosom 36
Gambar 4.10 Antarmuka Halaman Pembangkitan Nilai *Belief* 49
Gambar 4.11 Antarmuka Halaman Kasus 49
Gambar 4.12 Antarmuka Halaman Pembangkitan Nilai *Belief* 50
Gambar 4.13 Antarmuka Halaman Diagnosa 50
Gambar 5.1 Menu utama 63
Gambar 5.2 Tampilan Menu Rule Gejala 63
Gambar 5.3 Tampilan Kasus 64
Gambar 5.4 Tampilan Proses Algoritma Genetika 65
Gambar 5.5 Tampilan Deteksi Penyakit 65
Gambar 6.1 Grafik Pengujian Jumlah Populasi Terhadap Nilai *Fitness* 67
Gambar 6.2 Grafik Pengujian Jumlah Generasi Terhadap Nilai *Fitness* 68
Gambar 6.3 Grafik Pengujian Nilai Probabilitas *Crossover* dan Probabilitas Mutasi Terhadap Nilai *Fitness* 69
Gambar 6.4 Grafik Pengujian Akurasi 70

DAFTAR *SOURCE CODE*

Source Code 5.1 Inisialisasi Populasi Awal 53
Source Code 5.2 Proses *Extended Intermediate Crossover* 54
Source Code 5.3 Rumus *Extended Intermediate Crossover* 54
Source Code 5.4 Proses *Random Mutation* 56
Source Code 5.5 Perhitungan *Fitness* 56
Source Code 5.6 Metode Dempster Shafer 61
Source Code 5.7 Proses *Elitism Selection* 63

DAFTAR LAMPIRAN

LAMPIRAN A DAFTAR NILAI *BELIEF* DARI PAKAR 77
LAMPIRAN B Daftar gejala 78
LAMPIRAN C Data latih 79
LAMPIRAN D Data UJI..... 83

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Infeksi saluran napas bawah masih merupakan masalah utama dalam bidang kesehatan, baik negara yang sedang berkembang maupun yang sudah maju. Menurut CNN Indonesia, penyakit pneumonia yang merupakan penyakit infeksi saluran pernafasan bawah termasuk dalam 10 jenis penyakit paling sering menyebabkan kematian di Indonesia (Widowati, 2015). Dan pneumonia sendiri merupakan salah satu pencetus utama penyakit HIV/AIDS (Raviglione, 2008).

Pneumonia merupakan infeksi saluran pernafasan bawah yang mempengaruhi paru-paru. Paru-paru terdiri dari ribuan tabung (bronkus) yang membagi menjadi saluran udara yang lebih kecil (bronkiolus), yang berakhir pada kantung kecil (alveoli). Alveoli berisi kapiler di mana oksigen ditambahkan ke darah dan karbon dioksida akan dikeluarkan. Ketika seseorang memiliki pneumonia, nanah dan cairan mengisi alveoli pada satu atau kedua paru-paru dan akan mengganggu penyerapan oksigen dan membuat sulit bernapas (UNICEF/WHO, 2006).

Untuk mendiagnosis pneumonia perlu dilakukan uji laboratorium yang melewati banyak tahapan dan memerlukan waktu beberapa hari untuk mendapatkan hasilnya, sedangkan pneumonia dapat menyebabkan kematian bila tidak segera diobati. Dalam mendiagnosis pneumonia ragiografi dada merupakan cara terbaik untuk membedakan pneumonia dari infeksi saluran pernafasan lainnya (Melbye, 1992). Diagnosis yang tepat dapat mengurangi resiko pasien. Namun, karena tidak adanya fasilitas *chest x-ray* dan terbatasnya biaya, pasien dengan kemungkinan pneumonia tidak mendapat pemeriksaan *chest x-ray.* Akibatnya, pasien dengan infeksi pernafasan tersebut mendapat diagnosis dan penanganan yang salah.

Memang tidak mudah untuk menemukan diagnosa yang tepat, terutama bila gejala yang dialami pasien tidak jelas atau terlalu umum. Seperti pada penyakit pneumonia dan bronchitis akut dengan gejala yang terlalu umum seperti sakit kepala, demam, dan kelelahan, serta memiliki kemiripan gejala yang dapat memungkinkan dokter salah mendiagnosis. Pneumonia dan bronkitis akut sama-sama mempengaruhi sistem pernafasan bagian bawah dan kedua penyakit ini dapat meningkatkan resiko kanker paru-paru. Namun, penanganan terhadap kedua penyakit ini berbeda. Pada pneumonia sangat dianjurkan menggunakan antibiotic untuk mengurangi resiko komplikasi, sedangkan untuk bronkitis akut tidak dianjurkan menggunakan antibiotic (Evertsen, Baumgardner, Regnery, & Banerjee, 2010).

Masalah ketidakpastian dalam mendiagnosis pneumonia dan bronchitis akut dapat diselesaikan dengan bantuan teknologi yang sedang berkembang saat ini. Dalam kecerdasan buatan (*Artificial Intelligent*) terdapat berbagai metode yang

dapat menyelesaikan masalah ketidakpastian, salah satunya adalah metode Dempster-Shafer.

Metode *Dempster-Shafer* telah banyak digunakan untuk menyelesaikan masalah diagnosis penyakit dengan masukkan gejala dari pengguna. Seperti pada penelitian Dewi Pratama Kurniawati yang mengimplementasi metode Dempster Shafer untuk mendiagnosa jenis-jenis penyakit diabetes mellitus dengan masukkan gejala dari pengguna yang memberikan hasil akurasi 96,67% dari 30 data yang diuji (Kurniawati, 2014). Penelitian Aprilia Sulistyohati tentang sistem pakar penyakit ginjal menggunakan metode Dempster-Shafer menyimpulkan bahwa sistem pakar yang dibangun dapat memberikan hasil beserta tingkat kebenarannya berdasarkan nilai kepercayaan (nilai *belief)* yang dimiliki oleh gejala tiap masing-masing kasus (Sulistyohati & Hidayat, 2008).

Metode *Dempster-Shafer* sendiri adalah metode non monotonis yang dapat digunakan dalam mendiagnosa penyakit jantung koroner yang tidak konsistenan akibat adanya penambahan maupun pengurangan fakta baru yang akan mengubah aturan yang ada (Wahyuni & Prijodiprojo, 2013). Peneliti pun bermaksud untuk menerapkan metode Dempster Shafer dalam mendeteksi penyakit infeksi saluran pernafasan dari gejala umum yang terjadi.

Metode Dempster-Shafer membutuhkan nilai *belief*. Untuk mendapatkan nilai *belief* pakar diminta memprediksi pengaruh suatu gejala terhadap suatu penyakit dalam nilai random antara 0-1. Ini sama halnya dengan mengkonversi pengetahuan pakar ke dalam sebuah angka sedangkan nilai yang diberikan seorang pakar bisa berbeda dengan pakar lainnya. Hal ini pernah dibahas dalam sebuah penelitian yang mempertanyakan kredibilitas pengetahuan pakar yang dijadikan parameter probabilitas. Patrick Hester pun mengusulkan pengukuran kredibilitas suatu nilai *belief* yang diberikan pakar, namun belum ada penelitian lebih lanjut yang menunjukkan validitas penelitian ini (Hester, 2012). Peneliti pun bermaksud mengajukan metode lain untuk menyelesaikan masalah nilai *belief* ini dengan menggunakan metode yang sederhana dan sudah ada, yaitu algoritma genetika.

Penelitian mengenai prediksi pneumonia telah dibahas sebelumnya dalam jurnal yang berjudul *Use of genetic algorithms for neural networks to predict community-acquired pneumonia*. Paul Heckerling, dkk (2004) menggunakan algoritma genetika untuk mengoptimasi nilai parameter dalam metode Neural Network dalam mendeteksi pneumonia. Dari penelitian tersebut diketahui bahwa algoritma genetika dapat beradaptasi dengan metode neural network dan memberikan hasil yang baik dalam memprediksi pneumonia. Namun, pada penelitian tersebut peneliti memprediksi pasien yang telah melakukan *chest X-ray* sebelumnya dan menggunakan hasil *chest X-ray* sebagai parameternya. Sedangkan penelitian ini bertujuan untuk mendeteksi pneumonia dari gejala umum dan membedakannya dari penyakit infeksi saluran pernafasan lainnya sehingga data yang digunakan adalah gejala umum yang orang awam pun mengerti dan dapat digunakan untuk memprediksi pneumonia.

Algoritma Genetika merupakan komputasi sederhana namun kuat dalam pencarian untuk perbaikan. Dalam masalah optimasi, algoritma genetika sering digunakan untuk penyelesaiannya, seperti pada penelitian yang berjudul Implementasi Algoritma Genetika pada Struktur *Backpropagation Neural Network* untuk Klasifikasi Kanker Payudara yang menunjukkan hasil bahwa metode Neural Network dengan optimasi parameter menggunakan algoritma genetika menghasilkan rata-rata akurasi yang lebih baik daripada metode Naïve Bayesian dan metode *Neural Network* dengan *Association Rules* (Zamani, Amaliah, & Munif, 2012). Untuk itu penulis bermaksud menggunakan algoritma genetika untuk optimasi nilai *belief* dalam metode Dempster Shafer.

Berdasarkan paparan yang telah dijelaskan penulis mengusulkan penelitian dengan judul Penerapan Metode *Dempster-Shafer* Pada Deteksi Penyakit Infeksi Saluran Pernafasan Dengan Pembangkitan Nilai *Belief* Menggunakan Algoritma Genetika. Diharapkan aplikasi ini dapat mengidentifikasi penyakit infeksi saluran pernafasan berdasarkan gejala, serta memberikan hasil yang lebih baik ketika menggunakan algoritma genetika dalam membangkitkan nilai *belief*.

1.2 Rumusan masalah

Berdasarkan latar belakang yang telah diuraikan di atas maka didapat rumusan masalah yang meliputi:

1. Bagaimana mengimplementasikan algoritma genetika terhadap optimasi nilai *belief* pada metode Dempster Shafer untuk mendeteksi penyakit infeksi saluran pernafasan?
2. Berapa nilai-nilai parameter pada algoritma genetika yang paling optimal untuk mendapatkan nilai *belief* yang terbaik?
3. Bagaimana keakuratan deteksi penyakit infeksi saluran pernafasan dengan metode Dempster-Shafer?

1.3 Tujuan

1. Mengimplementasikan algoritma genetika pada optimasi nilai *belief* metode Dempster Shafer untuk mendeteksi penyakit pernafasan
2. Mengetahui nilai-nilai parameter pada algoritma genetika yang paling optimal terhadap nilai *belief* pada metode Dempster-Shafer.
3. Menguji tingkat akurasi metode Demspter Shafer pada aplikasi deteksi penyakit infeksi saluran pernafasan dengan optimasi nilai *belief* menggunakan algoritma genetika

1.4 Manfaat

Manfaat dari penelitian adalah:

1. Bagi penulis
   Dapat lebih memahami metode *Dempster-Shafer* dan pengembangannya dengan Algoritma Genetika
2. Bagi masyarakat umum (pengguna)

Dapat membantu masyarakat dalam mendeteksi penyakit pernafasan sehingga dapat dilakukan penanganan lebih dini.

1.5 Batasan masalah

1. Permasalahan penyakit yang dimasukkan ke dalam aplikasi ini hanya penyakit pneumonia, bronkitis akut, dan URTI (*Upper Respiratory Tract Infection*) berdasarkan jurnal kesehatan Jennifer Evertsen, dkk.
2. Parameter-parameter Algoritma Genetika yang digunakan adalah jumlah populasi, jumlah generasi, probabilitas *crossover*, dan probabilitas *mutation*.
3. Representasi kromosom yang digunakan adalah representasi bilangan rill (*real).*
4. Metode *crossover* yang digunakan adalah *extended intermediate crossover.*
5. Metode *mutation* yang digunakan adalah *random mutation*.
6. Aplikasi dibangun menggunakan bahasa pemrograman C# dan *Database Management System* yang digunakan adalah SQLite.

1.6 Sistematika pembahasan

Sistematika penulisan yang diterapkan dalam penelitian ini adalah sebagai berikut:

BAB I Pendahuluan

Pada bab ini dijabarkan latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, dan sistematika penulisan.

BAB II Tinjauan Pustaka

Bab ini menjelaskan tentang teori-teori yang digunakan serta kajian pustaka yang mengenai *Dempster-Shafer*, Algoritma Genetika, serta bahan kajian lain yang mendukung

BAB III Metodologi Penelitian dan Perancangan

Membahas tentang metode yang digunakan dalam penulisan yang terdiri dari studi literature, pengumpulan data, analisa kebutuhan, perancangan aplikasi, implementasi aplikasi, pengujian aplikasi, dan pengambilan kesimpulan dan saran.

BAB IV Implementasi dan Pembahasan

Membahas tentang hasil rancangan dari analisis kebutuhan serta implementasi dari hasil rancangan tersebut.

BAB V Pengujian dan Analisis

Membahas tentang pengujian dan analisis dari hasil rancangan.

BAB VI Penutup

Memuat kesimpulan dari penelitian yang telah dilakukan serta saran untuk pengembangan lebih lanjut.

BAB 2 TINJAUAN PUSTAKA

Bab ini terdiri dari kajian pustaka dan dasar teori. Kajian pustaka membandingkan penelitian ini dengan penelitian yang telah ada sebelumnya. Sedangkan dasar teori membahas teori yang digunakan untuk menunjang penelitian mengenai implementasi metode *Demspter-Shafer* untuk deteksi gangguan pernapasan dengan pembangkitan nilai *belief* menggunakan algoritma genetika.

2.1 Kajian Pustaka

Terdapat beberapa penelitian sebelumnya yang terkait dengan penelitian ini. Penelitian pertama berjudul 'Prototype Sistem Pakar untuk Mendeteksi Tingkat Resiko Penyakit Jantung Koroner dengan Metode *Dempster-Shafer* (Studi Kasus: RS PKU Muhammadiyah Yogyakarta)'. Dari penelitian tersebut diketahui bahwa sistem pakar dengan mesin inferensi *Dempster-Shafer* dapat dipergunakan untuk mendiagnosa tingkat resiko penyakit jantung koroner dengan masukan berupa gejala-gejala penyakit jantung coroner dan keluaran berupa nilai kepercayaan dari faktor resiko penyakit jantung yang dihasilkan (Wahyuni & Prijodiprojo, 2013).

Referensi selanjutnya yaitu penelitian yang dilakukan oleh Aprilia Sulistyohati dan Taufiq Hidayat dengan judul "*Aplikasi Sistem Pakar Diagnosa Penyakit Ginjal Dengan Metode Dempster-Shafer*". Penelitian ini menggunakan metode *Dempster-Shafer* untuk menyelesaikan permasalahan tidak konsistennya suatu aplikasi sistem pakar dalam menyelesaikan suatu masalah. *Input* penelitian ini gejala-gejala penyakit ginjal. *Output* dari penelitian ini yaitu hasil diagnosa, informasi mengenai penyakit ginjal, solusi pengobatan, dan nilai kepercayaan penyakit (Sulistyohati & Hidayat, 2008).

Penelitian dengan judul *'Epistematic Uncertainty Analysis: An approach Using Expert Judgment and Evidential Credibility'* mempertanyakan kredibilitas pengetahuan pakar yang dijadikan parameter probabilitas. Dalam teori Dempster-Shafer terdapat 3 fungsi penting, yaitu *Basic Probability Assignment* (BPA atau m), *belief function* (Bel), dan *Plausability function* (Pl). Untuk mendapatkan ketiga nilai tersebut dibutuhkan penilaian dari pakar. Pakar diminta untuk memprediksi nilai *belief* dengan mengkonversi pengetahuannya ke dalam sebuah angka, sedangkan setiap pakar memiliki perbedaan penilaian sehingga diperlukan penilaian dari beberapa pakar. Penelitian ini menunjukkan dalam metode pengambilan keputusan terdapat ketidakpastian dalam nilai yang diberikan pakar karena persepsi pakar dalam memberikan nilai dan analisis dapat berbeda dengan pakar yang lainnya (Hester, 2012).

Pada penelitian lainnya yang berjudul 'Implementasi Algoritma Genetika pada Struktur *Backpropagation Neural Network* untuk Klasifikasi Kanker Payudara' mengkombinasikan metode klasifikasi *Neural Network* dan algoritma genetika. *Neural network* digunakan sebagai *Artificial Intelligence* untuk

memprediksi kanker payudara, sedangkan algoritma genetika digunakan untuk optimasi parameter pada metode *Neural Network* seperti jumlah *hidden layer* dan *learning rate*. Penelitian ini menunjukkan akurasi yang lebih baik saat optimasi parameter menggunakan algoritma genetika dibandingkan menggunakan metode *Neural Network* dengan *Association Rules* (Zamani, 2012).

Perbedaan yang dibuat penulis pada penelitian ini adalah objek penelitian dan penggunaan algoritma genetika dalam mengoptimasi nilai *belief* pada metode Dempster Shafer. Metode Dempster Shafer yang biasanya menggunakan nilai *belief* yang diberikan pakar secara langsung digantikan dengan algoritma genetika. Algoritma genetika digunakan untuk mengoptimasi nilai *belief* sehingga diharapkan dapat menyelesaikan masalah deteksi penyakit infeksi saluran pernafasan melalui gejala umum.

Berdasarkan pemaparan di atas maka peneliti menggunakan *Dempster-Shafer* untuk mendeteksi penyakit infeksi saluran pernafasan dengan optimasi nilai *belief* menggunakan algoritma genetika. Objek penelitian ini menggunakan penyakit infeksi saluran pernafasan sesuai dengan jurnal kesehatan milik Jennifer Evertsen, dkk (2010) yang membahas mengenai prediksi pneumonia dan membedakannya dari penyakit infeksi saluran pernafasan lainnya yang memiliki gejala hampir sama. Diharapkan penelitian ini dapat mengidentifikasi penyakit pneumonia, bronkitis akut, dan penyakit infeksi saluran pernafasan bagian atas lainnya yang memiliki gejala umum serupa sesuai dengan identifikasi seorang pakar.

2.2 Penyakit Infeksi Saluran Pernafasan

Penyakit infeksi saluran pernafasan dapat dibagi menjadi dua, yaitu infeksi saluran pernafasan bawah dan infeksi saluran pernafasan atas. Pneumonia dan bronkitis akut termasuk dalam penyakit infeksi saluran pernafasan bawah yang mempengaruhi paru-paru dan sistem pernafasan. Pneumonia merupakan inflamasi akut pada parenkim paru (bronkiolus, duktus, dan kantung alveolus, dan alveoli) yang mengganggu pertukaran udara (Muscari, 2005). Sedangkan, bronchitis akut adalah suatu peradangan pada bronchus (Muttaqin, 2008).

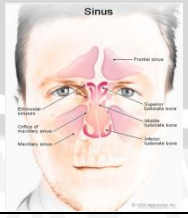
Pneumonia dan bronkitis akut memiliki gejala yang sama, namun memiliki penanganan yang berbeda. Bronkitis akut tidak memerlukan antibiotic, sedangkan pneumonia dibutuhkan antibiotic untuk mengurangi resiko komplikasi dari pneumonia (Evertsen, Baumgardner, Regnery, & Banerjee, 2010).

Pneumonia sendiri sulit untuk didiagnosis karena memiliki gejala umum seperti demam atau flu. Dokter mendiagnosis pneumonia berdasarkan riwayat kesehatan, pemeriksaaan fisik, dan serangkaian tes. Dokter akan menanyakan gejala yang dirasakan pasien. Dari jawaban yang dikemukakan pasien, dokter sudah memiliki beberapa diagnosis, yaitu diagnosis kerja (*working diagnosis)* yang merupakan diagnosis yang paling mungkin merupakan masalah pasien dan diagnosis banding (*differential diagnosis*) sebagai pengganti diagnosis kerja. Kemudian dilakukan pemeriksaan lebih lanjut untuk memastikan diagnosis kerja

tersebut, seperti pemeriksaan fisik dan laboratorium. Cara kerja dokter dalam menentukan diagnosis kerja dan diagnosis banding inilah yang digunakan penulis untuk membedakan pneumonia, bronkitis akut, dan URTI (*Upper Respiratory Tract Infection).*

Dalam jurnal Jennifer Evertsen, dkk. (2010) dijabarkan mengenai kriteria yang digunakan dalam mendiagnosa awal penyakit pneumonia dengan membandingkannya dengan bronkitis akut dan penyakit infeksi saluran pernafasan atas. Tabel 2.1 adalah gejala dari penyakit pneumonia, bronkitis, dan URTI yang dikutip dari jurnal Jennifer Evertsen, dkk.

Tabel 2.1 Tabel Gejala Penyakit Infeksi Saluran Pernafasan

	Penjelasan
Fever	Demam
Cough	Batuk
Chest pain	Ketidaknyamanan atau rasa sakit yang dirasakan di sepanjang bagian depan tubuh antara leher dan perut bagian atas. Jika sakit pada dada lebih dominan bagian kiri, menusuk, dan menjalar ke punggung maka bisa diindikasikan pada sakit jantung dan tidak berhubungan dengan pneumonia.
Chills	Menggigil yang mengacu pada rasa dingin setelah berada di lingkungan yang dingin. Dapat merujuk kepada sebuah episode menggigil(shivering) bersama dengan pucat(paleness) dan perasaan dingin (feeling cold).
Sore throat	Gejala sakit tenggorokan termasuk pada rasa sakit, terbakar atau scratching sensation (sensasi gatal) di bagian belakang tenggorokan, nyeri saat menelan, dan nyeri di leher. Gejala sakit tenggorokan bisa disertai dengan: batuk, bersin, demam, dan pembengkakan kelenjar getah bening.
Dyspnea (sesak napas)	Sulit atau nampak susah bernafas; sesak napas; Saat bernafas membutuhkan tenaga lebih.
Sinus pain	Sakit atau tekanan yang terjadi pada sekitar daerah sinus seperti pada gambar di bawah ini. 
Head congestion	Rasa sakit dan atau tekanan di kepala, wajah, dan mata hasil dari sinus tersumbat
Chest congestion	Gejala respirasi yang terjadi oleh karena banyaknya produksi mucus/dahak yang

	diakibatkan radang/infeksi. Saat bernafas, nafas terasa tidak bersih
Sputum: kental, kuning kehijauan	Dahak yang keluar bersamaan dengan batuk dengan tekstur kental dan warna kuning kehijauan
Sputum: Jernih, putih	Dahak yang keluar bersamaan dengan batuk dengan warna putih atau jernih
Rhinorrhea (runny nose)	Pilek
Temperature > 100˚F	Suhu badan saat diukur thermometer lebih dari 100˚F atau 37,8˚C
Sinusitis removed	Kondisi saat sinus telah dihilangkan
>10days	Gejala yang dialami berlangsung lebih dari 10 hari
7-10 days	Gejala yang dialami berlangsung antara 7 sampai 10 hari
2-7 days	Gejala yang dialami berlangsung antara 2 sampai 7 hari

2.2.1 Pneumonia

Secara kinis pneumonia didefinisikan sebagai suatu peradangan paru yang disebabkan oleh mikroorganisme (bakteri, virus, jamur, parasit) (Indonesia, Perhimpunan Dokter Paru, 2003). Pneumonia merupakan infeksi saluran pernafasan bawah yang mempengaruhi paru-paru. Paru-paru terdiri dari ribuan tabung (bronkus) yang membagi menjadi saluran udara yang lebih kecil (bronkiolus), yang berakhir pada kantung kecil (alveoli). Alveoli berisi kapiler di mana oksigen ditambahkan ke darah dan karbon dioksida akan dikeluarkan. Ketika seseorang memiliki pneumonia, nanah dan cairan mengisi alveoli pada satu atau kedua paru-paru dan akan mengganggu penyerapan oksigen dan membuat sulit bernapas (UNICEF/WHO, 2006).

Gambaran klinik biasanya ditandai dengan demam, menggigil, suhu tubuh meningkat dapat melebihi 40ºC, batuk dengan dahak mukoid atau purulen kadang-kadang disertai darah, sesak napas dan nyeri dada (Indonesia, Perhimpunan Dokter Paru, 2003).

Setelah diagnosis dapat ditegakkan oleh dokter dapat dilakukan pengobatan. Pengobatan terdiri atas antibiotik dan pengobatan suportif. Pemberian antibiotik pada penderita pneumonia sebaiknya berdasarkan data mikroorganisme dan hasil uji kepekaannya (Indonesia, Perhimpunan Dokter Paru, 2003).

2.2.2 Bronkitis Akut

Bronkitis akut merupakan infeksi bronki pada paru-paru dimana saluran udara membengkak dan terdapat lendir tebal di dalamnya. Seringkali disebabkan oleh virus yang sama yang menyebabkan demam/flu biasa. Bronkitis akut dapat terjadi bersamaan dengan demam atau influenza, dan dapat terjadi tersendiri (Asthma and Respiratory Foundation of New Zealand (Inc.), 2010).

Gejala utamanya adalah batuk secara produktif bersamaan dengan dahak. Batuk bertahan selama beberapa minggu. Beberapa orang mengalami nafas pendek dan wheeze (suara pernapasan frekuensi tinggi nyaring yang terdengar di akhir ekspirasi). Juga terdapat beberapa gejala yang berhubungan seperti pilek, sakit tenggorokan, demam, *muscle aches and pain*, *sore eye* atau sakit kepala. Chest pain (nyeri dada) biasanya tidak berhubungan dengan bronkitis. Tidak ada test spesifik untuk mendeteksi bronkitis akut (Asthma and Respiratory Foundation of New Zealand (Inc.), 2010).

2.2.3 URTI (Upper Respiratory Tract Infection)

URTI merupakan infeksi kuman yang menyebabkan radang dan infeksi pada hidung dan tenggorokan. URTI dikategorikan sebagai kumpulan penyakit yang terdiri dari common cold, pharyngitis, tonsillitis, epiglottitis, sinusitis, bronchitis, rhinitis, dan nasophaaryngitis, yang secara signifikan terjadi pada pernafasan bagian atas (Rohilla, Sharma, & Kumar, 2013).

Batuk merupakan gejala utama. Gejala lainnya adalah demam, sakit kepala, dan sakit. Flu terjadi jika infeksi menjalar ke hidung. Kebanyakan URTI tidak menyebabkan komplikasi. URTI juga memicu batuk, wheezing, dan nafas pendek pada penderita asma atau penyakit pernafasan lainnya.

Kebanyakan infeksi pernafasan bagian atas (URTI) disebabkan oleh viral infection (kuman) sehingga antibiotics tidak dianjurkan karena akan memperburuk keadaan yang menyebabkan efek samping berupa diare, merasa sakit dan ruam. Antibiotic dianjurkan jika keadaan semakin buruk dengan komplikasi seperti pada pneumonia, atau jika memiliki penyakit paru-paru kronik (Upper Respiratory Tract Infection, 2012).

2.3 Metode Dempster Shafer

Terdapat berbagai macam penalaran dengan model yang lengkap dan sangat konsisten, tetapi pada kenyataannya banyak domain permasalahan yang tidak dapat terselesaikan sesuai dengan harapan. Penambahan fakta baru adalah salah satu penyebab dari ketidakkonsistenan tersebut. Untuk menyelesaikan permasalahan ketidak konsistenan tersebut maka dapat menggunakan penalaran dengan teori *Dempster-Shafer*. Secara Umum teori *Dempster-Shafer* ditulis dalam suatu interval seperti pada persamaan berikut (Sulistyohati & Hidayat, 2008):

[*Belief,Plausibility*] (2.1)

Belief (Bel) adalah ukuran kekuatan *evidence* dalam mendukung suatu himpunan proposisi. Jika bernilai 0 maka mengindikasikan bahwa tidak ada *evidence*, dan jika bernilai 1 menunjukkan adanya kepastian. *Plausibility* (Pl) dinotasikan sebagai:

$Pl(s) = 1 - Bel(\neg s)$ (2.2)

Plausibility juga bernilai 0 sampai 1. Jika yakin akan ¬s, maka dapat dikatakan bahwa Bel(¬s) = 1, dan Pl(¬s) = 0.

Pada teori *Dempster-Shafer* dikenal adanya *Frame of Discernment* yang dinotasikan dengan θ. Frame ini merupakan semesta pembicaraan dari sekumpulan hipotesis.

Misalkan : θ = {P,B,U}
Dengan : P = Pneumonia;
B = Bronkitis Akut;
U = URTI (*Upper Respiratory Tract Infection*);

Tujuannya adalah mengaitkan ukuran kepercayaan elemen-elemen θ. Tidak semua *evidence* secara langsung mendukung tiap-tiap elemen.

Untuk itu perlu adanya probabilitas fungsi densitas (m). Nilai m tidak hanya mendefinisikan elemen-elemen θ saja, namun juga semua subsetnya. Sehingga jika θ berisi n elemen, maka subset θ adalah 2^n. Kita harus menunjukkan bahwa jumlah semua m dalam subset θ sama dengan 1. Apabila tidak ada informasi apapun untuk memilih ketiga hipotesis tersebut, maka nilai :

m{θ} = 1,0

Apabila diketahui X adalah subset dari θ, dengan m1 sebagai fungsi densitasnya, dan Y juga merupakan subset dari θ dengan m2 sebagai fungsi densitasnya, maka dapat dibentuk fungsi kombinasi m1 dan m2 sebagai m3 sehingga didapatkan Persamaan 2.3, yaitu:

$$m3(Z) = \frac{\sum_{X \cap Y = Z} m1(X).m2(Y)}{1 - \sum_{X \cap Y = \emptyset} m1(X).m2(Y)} \quad (2.3)$$

Keterangan :
m = Nilai Densitas (Kepercayaan)

XYZ = Himpunan *Evidence*

Ø = Himpunan Kosong

2.4 Algoritma Genetika

Algoritma ini ditemukan oleh John Holland dalam sebuah penelitian yang selanjutnya dipopulerkan oleh David Goldberg yang merupakan murid dari John Holland. Penelitian ini dilakukan pada tahun 1975 di Universitas Michigan, Amerika Serikat. Dengan meniru teori evolusi Charles Darwin yang bernama survival of fittnest, algoritma genetika ini mengadopsi proses sistem alam, yaitu dirancang dengan menirukan proses sistem alam yaitu proses crossover, mutasi, dan seleksi. Istilah yang digunakan dalam algoritma genetika ini pun mengadopsi dari ilmu genetika seperti kromosom, gen, crossover, mutasi, dan lain-lain (Sutojo dkk., 2011).

Kelebihan algoritma genetika sebagai metode optimasi (Mahmudy, 2014) :

1. Algoritma genetika dapat digunakan dalam masalah yang sangat kompleks. (Gen & Cheng dalam Mahmudy, 2014)
2. Individu yang ada pada populasi mampu dibagi menjadi sub-populasi yang dapat diproses pada komputer secara paralel sehingga mengurangi waktu komputasi pada masalah yang kompleks (Defersha & Chen, Qi, Burn & Harrison dalam Mahmudy, 2014). Sub-populasi dapat dilakukan pada satu komputer untuk menjaga keragaman dan meningkatkan kualitas hasil pencarian (Mahmudy dalam Mahmudy 2014).
3. Algoritma genetika menghasilkan himpunan solusi yang optimal yang sangat berguna pada penyelesaian masalah dengan banyak obyektif (Mahmudy & Rahman dalam Mahmudy, 2014).
4. Algoritma genetika dapat menggunakan banyak variabel. Variabel tersebut bisa bersifat kontinyu, diskrit atau campuran keduanya (Haupt & Haupt dalam Mahmudy, 2014).
5. Algoritma genetika menggunakan kromosom untuk mengkodekan solusi sehingga bisa melakukan pencarian tanpa memperhatikan informasi derivative yang spesifik dari masalah yang diselesaikan (Gen & Cheng, Haupt & Haupt dalam Mahmudy, 2014).
6. Algoritma genetika bisa diimplementasikan pada berbagai macam data seperti data yang dibangkitkan secara numeric atau menggunakan fungsi analitis (Haupt & Haupt dalam Mahmudy, 2014).
7. Algoritma genetika cukup fleksibel untuk dihibridisasikan dengan algoritma genetika lainnya (Gen & Cheng dalam Mahmudy, 2014).
8. Beberapa penelitian membuktikan bahwa algoritma genetika hybrid sangat efektif untuk menghasilkan solusi yang lebih baik (Mahmudy, Marian & Luong dalam Mahmudy, 2014).
9. Algoritma genetika bersifat ergodic, sembarang solusi bisa diperoleh dari solusi yang lain dengan hanya beberapa langkah. Hal ini memungkinkan eksplorasi pada daerah pencarian yang sangat luas dilakukan dengan lebih cepat dan mudah (Marian dalam Mahmudy, 2014).

Tahap-tahap proses dalam algoritma genetika (Mahmudy, 2014):

1. Representasi kromosom
2. Proses inisialisasi
3. Proses reproduksi. Proses reproduksi terdiri dari proses crossover dan mutasi.
4. Proses evaluasi dengan cara menghitung *fitness*.
5. Proses seleksi. Metode yang digunakan pada proses seleksi bermacam-macam, yaitu metode *elitism*, metode *roulette wheel* dan metode turnamen.
6. Proses memperoleh generasi baru.

2.4.1 Representasi Kromosom (*Chromosome*)

Sebuah algoritma genetika dimulai dengan sebuah populasi dari kromosom-kromosom yang dibangkitkan secara acak dan dilakukan berulang untuk menemukan kromosom yang lebih baik dengan menggunakan operator genetika

(Herrera, Lozano, & Verdegay, 1998). Representasi kromosom atau biasa disebut dengan encoding dalam algoritma genetika dapat menggunakan bilangan real, desimal atau biner. Namun tidak hanya itu, representasi kromosom secara permutasi juga dapat digunakan.

1. Representasi kromosom menggunakan bilangan real atau real-number encoding, nilai gen berada dalam interval [0,R] dimana R adalah bilangan real positif dan biasanya R=1.
2. Representasi kromosom menggunakan bilangan decimal atau *discrete decimal encoding,* nilai gen bisa berupa deretan bilangan bulat dalam interval [0,9].
3. Representasi kromosom menggunakan bilangan biner atau binary encoding, nilai gen bisa berupa derertan nilai 0 atau 1.
4. Representasi kromosom menggunakan permutasi atau *permutation encoding,* setiap individu berisi satu kromosom yang berisi gen-gen yang merepresentasi suatu solusi.

2.4.2 Inisialisasi

Inisialisasi merupakan tahap awal dalam proses algoritma genetika yang dilakukan untuk membangkitkan himpunan solusi baru secara acak. Himpunan solusi ini terdiri atas sejumlah *chromosome* yang ditempatkan pada penampungan yang disebut populasi. Ukuran dari populasi tersebut (*popSize*) harus ditentukan terlebih dahulu dengan menentukan banyaknya individu/*chromosome* yang ditampung dalam populasi (Mahmudy, 2013).

Dalam tahap ini akan dihitung nilai *fitness* dari masing-masing *chromosome*. Fungsi *fitness* dalam algoritma genetika merupakan sebuah tipe khusus dari fungsi objektif yang mengukur optimalitas dari sebuah kromosom (solusi) sehingga kromosom tersebut dapat diurutkan sesuai dengan nilai *fitness* dari keseluruhan kromosom (Lin, Huang, & Tsai, 2010).

2.4.3 Reproduksi

Reproduksi merupakan tahapan yang dilakukan setelah proses inisialisasi untuk menghasilkan keturunan dari individu-individu yang ada dalam populasi. Operator yang digunakan dalam proses ini adalah tukar silang (*crossover*) dan mutasi (*mutation*) (Mahmudy, 2013).

2.4.3.1 Crossover

C*rossover* adalah sebuah metode untuk mengkombinasikan ciri-ciri dari dua *chromosome parent* yang akan membentuk dua *offspring* dengan kemungkinan bahwa kromosom yang bagus dapat menghasilkan kromosom yang bagus pula (Herrera, Lozano, & Verdegay, 1998).

Dalam algoritma genetika dengan pengkodean real (Real-code Genetic Algorithm), metode crossover yang digunakan adalah *extended intermediate crossover* yang dilakukan dengan memilih dua induk (*parent*) secara acak dari populasi, kemudian akan dihasilkan offspring dari kombinasi nilai dua induk

tersebut. Dalam tahap reproduksi harus ditentukan tingkat *crossover* (*crossover rate*) yang menyakatan rasio *offspring* yang dihasilkan proses *crossover* terhadap ukuran populasi. Dengan adanya *crossover rate* (pc) akan dihasilkan *offspring* sebanyak pc × *popSize* (Mahmudy, 2013).

Dalam crossover terdapat istilah jumlah *offspring* (jumlah keturunan yang dihasilkan). Jumlah *offspring* didapat dari perkalian *crossover rate* dengan ukuran populasi. Jika terdapat 10 populasi dan *crossover rate*-nya 0,5, maka didapat 5 *offspring*. Ada beberapa metode *crossover* diantaranya (Mahmudy, 2013):

1. *Extended Intermediate Crossover*

Metode ini digunakan untuk algoritma genetika dengan pengkodean real (Real-code Genetic Algorithm) dengan membangkitkan *offspring* menggunakan persamaan yang ditunjukan pada persamaan 2.4 dan 2.5 (Mahmudy, 2013).

$$C_1 = P_1 + \alpha(P_2 - P_1) \quad (2.4)$$

$$C_2 = P_2 + \alpha(P_1 - P_2) \quad (2.5)$$

dimana α dipilih secara acak antara -0,25 sampai dengan 1,25.

Misalkan *parent* yang terpilih $P_1 = [0.35, 0.65]$ dan $P_2 = [0.55, 0.78]$ dengan $\alpha = [0.11, 1.23]$, maka akan menghasilkan dua *offspring* sebagai berikut

C_1: $x_1 = 0.35 + 0.11(0.55 - 0.35) = 0.37$

$x_2 = 0.65 + 1.23(0.78 - 0.65) = 0.81$

C_2: $x_1 = 0.55 + 0.11(0.35 - 0.55) = 0.53$

$x_2 = 0.78 + 1.23(0.65 - 0.78) = 0.62$

Anak yang dihasilkan melalui perhitungan *offspring* adalah $C_1 = [0.37, 0.81]$ dan $C_2 = [0.53, 0.62]$.

2. *One-Cut-Point Crossover*

Metode ini membagi susunan gen menjadi dua bagian, anak pertama akan mendapatkan susunan gen pertama dari induk pertama dan susunan gen kedua dari induk kedua, sebaliknya anak kedua mendapatkan susunan gen kedua dari induk pertama dan susunan gen kedua dari induk kedua (Mahmudy, 2013). Contohnya sebagai berikut:

P_1 [1 0 | 0 1]
P_2 [0 1 | 1 0]
C_1 [1 0 | 1 0]
C_2 [0 1 | 0 1]

2.4.3.2 Mutasi

Operator mutasi mengubah satu atau lebih komponen atau gen dari kromosom terpilih untuk meningkatkan keanekaragaman struktur populasi. Mutasi mengembalikan materi genetik yang hilang atau yang belum diselidiki ke dalam populasi untuk mencegah konvergensi prematur dan memperoleh solusi suboptimal (Herrera, Lozano, & Verdegay, 1998).

Dalam tahap reproduksi ditentukan pula nilai tingkat mutasi (*mutation rate*) yang menyatakan rasio *offspring* yang dihasilkan dari proses mutasi terhadap ukuran populasi. Dengan adanya *mutation rate* (pm) akan dihasilkan *offspring* sebanyak pm × *popSize.* Berikut ini beberapa metode yang ada pada mutasi (Mahmudy, 2013):

1. *Random Mutation*

Metode ini membangkitkan gen kromosom anak dengan menggunakan persamaan yang ditunjukkan oleh persamaan 2.6 berikut (Mahmudy, 2013):

$$x'_i = x'_i + r(max_i - min_j) \quad (2.6)$$

Keterangan
- x'_i merupakan gen induk yang dipilih untuk nantinya dapat dikurangi atau ditambahkan bilangan random
- r merupakan bilangan random
- $[min_j, max_j]$ domain variabel x_j

Misalkan *parent* yang terpilih $P = [0.45, 0.73]$ dengan $r = -0.06$, maka akan menghasilkan dua *offspring* sebagai berikut:

$$C: \quad x_1 = 0.45 - 0.06(0.53 - 0.37) = 0.44$$
$$x_2 = 0.73\ (tetap)$$

Anak yang dihasilkan berdasarkan perhitungan *Random Mutation* adalah $C = [0.44, 0.73]$.

2. *Reciprocal Exchange Mutation*

Metode ini menukarkan posisi gen *parent* (exchange point / XP) yang dipilih secara random (Mahmudy, 2013). Contohnya sebagai berikut:

P_1 [2 4 1 5 3]
C_1 [2 5 1 4 3]

3. *Insertion Mutation*

Metode *Insertion Mutation* bekerja dengan menyisipkan satu posisi yang dipilih secara random sehingga menggeser posisi yang lainnya (Mahmudy, 2013). Contohnya adalah sebagai berikut:

P_1 [2 4 1 5 3]
C1 [1 2 4 5 3]

2.4.4 Evaluasi

Tahapan setelah reproduksi yaitu proses evaluasi untuk menghitung nilai *fitness* setiap *chromosome*. Semakin besar nilai *fitness* dari suatu individu maka semakin baik *chromosome* tersebut untuk dijadikan sebagai calon solusi (Mahmudy, 2013). Semua individu mulai dari tahap inisialisasi dan reproduksi akan dikumpulkan bersama nilai fitnessnya.

Pada umumnya dibutuhkan *fitness* yang bernilai positif, oleh karena itu beberapa jenis dari penskalaan secara monoton dan/atau translasi mungkin

diperlukan jika fungsi objektif tidak sepenuhnya positif. Tergantung pada nilai *fitness*, kromosom tertentu diperbolehkan untuk berkembang biak dan mencampurkan kromosom yang lain melalui operator *crossover* untuk menghasilkan generasi baru yang diharapkan akan lebih baik (Lin, Huang, & Tsai, 2010).

2.4.5 Seleksi

Seleksi adalah tahapan akhir yang dilakukan untuk memilih individu dari himpunan populasi dan *offspring* yang dipertahankan hidup pada generasi berikutnya. Semakin besar nilai fitness suatu individu maka kemungkinan individu itu terpilih semakin besar. Sehingga individu ini dapat digunakan pada generasi selanjutnya dan diharapkan pada generasi berikutnya menghasilkan fitness yang lebih baik (Mahmudy, 2013).

Jumlah individu dari generasi selanjutnya sama dengan jumlah individu pada generasi sekarang. Jika pada generasi awal terdapat 100 individu dalam populasi maka populasi pada generasi selanjutnya akan memiliki 100 individu juga. Namun, kemungkinan akan diperoleh populasi dengan seluruh individu yang sama dengan seluruh individu dalam populasi pada generasi pertama akan sangat kecil karena pada generasi pertama telah dihasilkan *offspring* yang mungkin memiliki kromosom lebih baik dari kromosom *parent*.

Terdapat beberapa jenis metode seleksi, yaitu (Mahmudy, 2013):

1. *Replacement Selection*

Metode ini memiliki beberapa ketentuan diantaranya sebagai berikut (Mahmudy, 2013):

- Jika o*ffspring* yang dihasilkan dari proses *crossover* memiliki *fitness* yang lebih baik dari induknya, maka *offspring* akan menggantikan induk yang memiliki *fitness* terkecil.
- Jika *offspring* yang dihasilkan dari proses mutasi memiliki *fitness* yang lebih baik dari induknya, maka *offspring* akan menggantikan induknya.

Dimisalkan terdapat populasi induk seperti yang ditunjukkan pada Tabel 2.2:

Tabel 2.2 Contoh populasi untuk metode *Replacement Selection*

Individu	*Fitness*
P_1	13
P_2	11
P_3	15
P_4	16
P_5	19

Dari Tabel 2.2 dianggap menghasilkan *offspring* seperti yang ditunjukkan pada Tabel 2.3.

Tabel 2.3 Contoh *offspring* yang dihasilkan setelah proses *crossover* dan mutasi

Individu	*Parents*	*Fitness*
C_1	P_2 dan P_3	18
C_2	P_4 dan P_5	16
C_3	P_1	12

Berdasarkan data yang ditunjukkan pada Tabel 2.2 dan Tabel 2.3, C_1 menghasilkan *fitness* yang lebih baik dari induknya, karena C_1 merupakan *offspring* yang dihasilkan dari proses *crossover* dari induk P_2 dan P_3, dan P_2 merupakan *parent* yang memiliki *fitness* yang lebih rendah dibandingkan P_3 maka C_1 akan menggantikan P_2 Hasil seleksi dengan menggunakan metode *Replacement Selection* ditunjukkan pada Tabel 2.4.

Tabel 2.4 Contoh hasil seleksi dengan menggunakan metode *Replacement Selection*

Individu	Asal	*Fitness*
P_1	P_1	13
P_2	C_1	11
P_3	P_3	15
P_4	P_4	16
P_5	P_5	19

2. *Roulette Wheel*

Langkah-langkah yang dilakukan pada metode seleksi *Roulette Wheel* ini adalah sebagai berikut (Mahmudy, 2013) :

- Menghitung total *fitness* dengan menggunakan persamaan 2.16

$$\text{totalFitness} = \sum_{k=1}^{\text{popSize}} \text{fitness}(\text{Parent}_k) \qquad (2.16)$$

- Menghitung probabilitas seleksi (*prob*) dengan menggunakan persamaan 2.17

$$prob_k = \frac{fitness(Parent_k)}{totalFitness} \qquad (2.17)$$

- Menghitung nilai probabilitas kumulatif (*probCum*) dengan menggunakan persamaan 2.18

$$probCum_k = \sum_{j=1}^{k} prob_j \qquad (2.18)$$

3. *Tournament Selection*

Metode ini mengambil beberpa individu, yang nantinya individu dengan nilai terbesar akan menjadi individu baru pada generasi baru, proses ini dilakukan

sebanyak ukuran populasi yang ditentukan (Mahmudy, 2013). Misalnya terdapat populasi seperti pada Tabel 2.5.

Tabel 2.5 Contoh populasi untuk metode *Tournament Selection*

P(t+1)	*Fitness*
P_1	24,56
P_2	26,73
P_3	30,54
P_4	31,22
P_5	23,55
C_1	29,23
C_2	33,02
C_3	30.11

Berdasarkan populasi yang ditunjukkan pada Tabel 2.4, selanjutnya akan menyeleksi individu, hasil seleksi yang didapatkan ditunjukkan pada Tabel 2.6.

Tabel 2.6 Contoh hasil seleksi menggunakan metode *Tournament Selection*

P(t+1)	Individu Pertama	Individu Kedua	Individu Terpilih	*Fitness*
P_1	P_1	C_2	C_2	33,02
P_2	C_3	C_1	C_3	30.11
P_3	P_2	P_4	P_4	31,22
P_4	P_5	C_3	C_3	30.11
P_5	P_3	P_1	P_3	30,54

4. *Elitism Selection*

Pada metode ini akan dilakukan pengumpulan individu-individu yang ada pada populasi (*parents* dan *offspring*). Semua individu akan diurutkan dari *fitness* terbesar sampai terkecil. Individu sejumlah ukuran populasi teratas akan menjadi generasi baru (Mahmudy, 2013). Misalkan terdapat populasi seperti yang ditunjukkan pada Tabel 2.7.

Tabel 2.7 Contoh populasi untuk metode *Elitism Selection*

P(t+1)	*Fitness*
P_1	24,56
P_2	26,73
P_3	30,54
P_4	31,22
P_5	23,55
C_1	29,23
C_2	33,02
C_3	30.11

Selanjutnya populasi diurutkan berfasarkan nilai *fitnesnya*, dan hasilnya adalah seperti yang ditunjukkan pada Tabel 2.8.

Tabel 2.8 Hasil perankingan semua individu

Rank	P(t+1)	*Fitness*
1	C_2	33,02
2	P_4	31,22
3	P_3	30,54
4	C_3	30.11
5	C_1	29,23
6	P_2	26,73
7	P_1	24,56
8	P_5	23,55

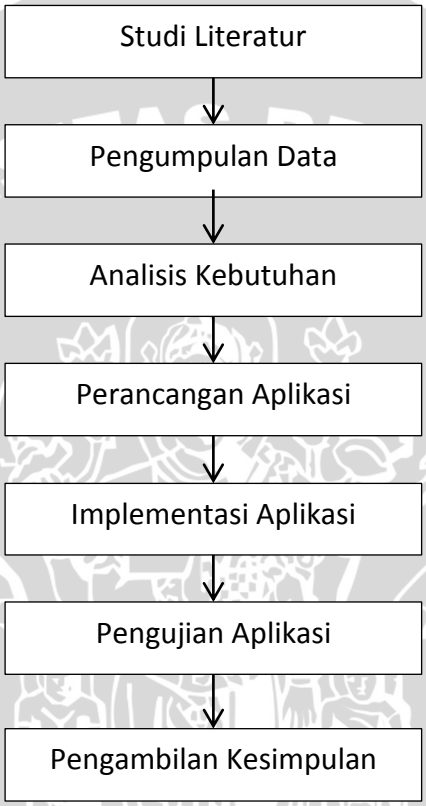
Ukuran populasi yang ditentukan adalah 5, sehingga hasil seleksi dengan menggunakan metode *Elitism Selection* ditunjukkan pada Tabel 2.9.

Tabel 2.9 Hasil seleksi menggunakan metode *Elitism Selection*

Individu	Asal	*Fitness*
P_1	C_2	33,02
P_2	P_4	31,22
P_3	P_3	30,54
P_4	C_3	30.11
P_5	C_1	29,23

BAB 3 METODOLOGI PENELITIAN

Bab ini menjelaskan tentang metode dan perancangan yang akan digunakan dalam penelitian ini, yaitu perancangan, implementasi, dan pengujian dari aplikasi perangkat lunak yang akan dibuat ini. Kesimpulan dan saran akan disertakan sebagai catatan atas aplikasi dan kemungkinan pengembangan aplikasi selanjutnya. Gambar 3.1 berikut ini adalah blok diagram dari metodologi yang dilakukan.



Gambar 3.1 Tahapan penelitian

3.1 Studi Literatur

Studi literatur digunakan untuk memahami konsep dan dasar teori yang akan diterapkan dalam penelitian ini. Berikut ini merupakan teori yang akan diperlajari dalam studi literatur:

1. Penyakit dan gejala penyakit pneumonia, bronkitis akut, dan URTI (*Upper Respiratory Tract Infection*)
2. Metode *Dempster-shafer*
3. Algortima genetika

Literatur tersebut diperoleh dari buku, jurnal, karya tulis ilmiah, website, dan penjelasan dari dr. Triwahju Astuti M.Kes, Sp.P(K) sebagai pakar dalam bidang kesehatan paru-paru.

3.2 Pengumpulan Data

Tahapan ini bertujuan untuk mengumpulkan data yang nantinya akan digunakan dalam aplikasi. Aplikasi ini membutuhkan data gejala dari penyakit pneumonia, bronkitis akut, dan URTI (*Upper Respiratory Tract Infection*). Data yang digunakan untuk penelitian ini adalah data *dummies,* yaitu data yang diperoleh dari hasil observasi dan wawancara dengan pakar secara langsung. Sehingga dalam penelitian ini digunakan data gejala dari jurnal kesehatan Jennifer Evertsen, dkk yang berjudul "*Diagnosis and management of pneumonia and bronchitis in outpatient primary care ractices*" dengan bantuan pakar, dr. Triwahju Astuti M.Kes, Sp.P(K) sebagai acuan dan pembanding dalam penelitian ini. Tabel 3.1 merupakan bentuk data yang diperoleh dari wawancara, jurnal, dan observasi bersama pakar. Tanda centang menunjukkan gejala yang mempengaruhi penyakit.

Tabel 3.1 Akusisi pengetahuan

		Pneumonia	Bronkitis Akut	URTI
G1	Fever	✓	✓	✓
G2	Cough	✓	✓	✓
G3	Chest pain	✓		
G4	Chills	✓	✓	✓
G5	Sore throat		✓	✓
G6	Dyspnea (sesak napas)	✓	✓	
G7	Sinus pain		✓	✓
G8	Head congestion		✓	✓
G9	Chest congestion	✓	✓	
G10	Sputum: kental, kuning kehijauan	✓		
G11	Sputum: Jernih, putih		✓	
G12	Rhinorrhea (runny nose)			✓
G13	Temp > 100F	✓	✓	
G14	Sinusitis removed		✓	
G15	>10days	✓		
G16	7-10 days	✓	✓	
G17	2-7 days		✓	✓

Pada tabel 3.2 merupakan penjelasan dari gejala-gejala yang telah disebutkan berdasarkan jurnal Jennifer Eversten, dkk dengan bantuan pakar sebagai narasumber.

Tabel 3.2 Penjelasan gejala

		Penjelasan
G1	Fever	Demam
G2	Cough	Batuk
G3	Chest pain	Ketidaknyamanan atau rasa sakit yang dirasakan di sepanjang bagian depan tubuh antara leher dan perut

		bagian atas. Jika sakit pada dada lebih dominan bagian kiri, menusuk, dan menjalar ke punggung maka bisa diindikasikan pada sakit jantung dan tidak berhubungan dengan pneumonia.
G4	Chills	Menggigil yang mengacu pada rasa dingin setelah berada di lingkungan yang dingin. Dapat merujuk kepada sebuah episode menggigil(shivering) bersama dengan pucat(paleness) dan perasaan dingin (feeling cold).
G5	Sore throat	Gejala sakit tenggorokan termasuk pada rasa sakit, terbakar atau scratching sensation (sensasi gatal) di bagian belakang tenggorokan, nyeri saat menelan, dan nyeri di leher. Gejala sakit tenggorokan bisa disertai dengan: batuk, bersin, demam, dan pembengkakan kelenjar getah bening.
G6	Dyspnea (sesak napas)	Sulit atau nampak susah bernafas; sesak napas; Saat bernafas membutuhkan tenaga lebih.
G7	Sinus pain	Sakit atau tekanan yang terjadi pada sekitar daerah sinus seperti pada gambar di bawah ini.
G8	Head congestion	Rasa sakit dan atau tekanan di kepala, wajah, dan mata hasil dari sinus tersumbat
G9	Chest congestion	Gejala respirasi yang terjadi oleh karena banyaknya produksi mucus/dahak yang diakibatkan radang/infeksi. Saat bernafas, nafas terasa tidak bersih
G10	Sputum: kental, kuning kehijauan	Dahak yang keluar bersamaan dengan batuk dengan tekstur kental dan warna kuning kehijauan
G11	Sputum: Jernih, putih	Dahak yang keluar bersamaan dengan batuk dengan warna putih atau jernih
G12	Rhinorrhea (runny nose)	Pilek
G13	Temperature > 100˚F	Suhu badan saat diukur thermometer lebih dari 100˚F atau 37,8˚C
G14	Sinusitis removed	Kondisi saat sinus telah dihilangkan
G15	>10days	Gejala yang dialami berlangsung lebih dari 10 hari
G16	7-10 days	Gejala yang dialami berlangsung antara 7 sampai 10 hari
G17	2-7 days	Gejala yang dialami berlangsung antara 2 sampai 7 hari

3.3 Analisis Kebutuhan

Analisis kebutuhan bertujuan untuk mendapatkan semua kebutuhan dalam membangun aplikasi agar tidak menyimpang dari permasalahan dan tujuan penelitian. Analisis kebutuhan dilakukan dengan mengidentifikasi kebutuhan sistem antara lain kebutuhan masukan, kebutuhan proses dan kebutuhan keluaran.

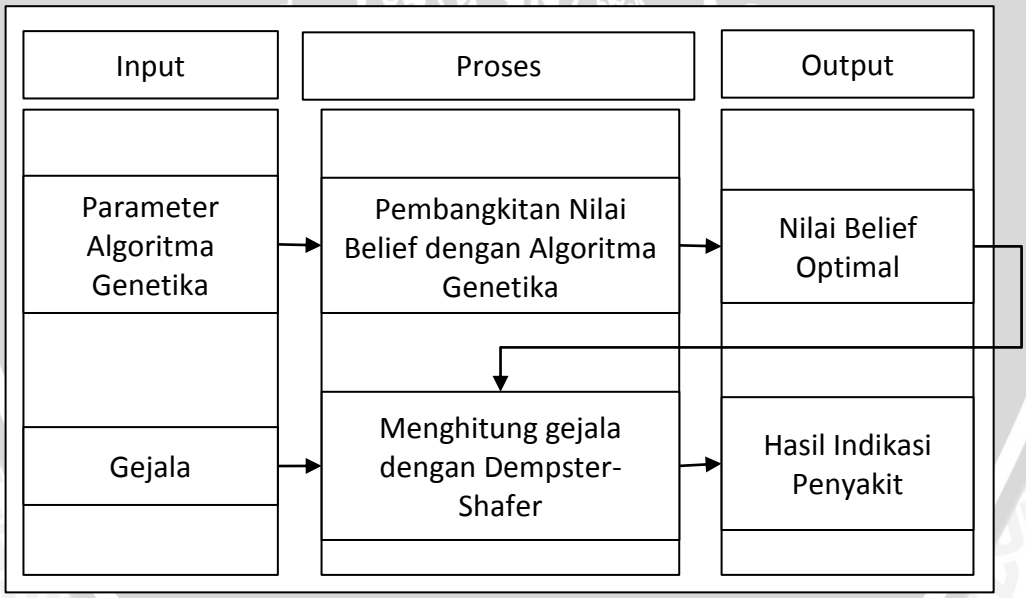
1. Kebutuhan perangkat keras, meliputi:
   - Laptop dengan *Processor* Intel® Core™ i5-3210M CPU @ 2.50GHz
2. Kebutuhan perangkat lunak, meliputi:

- Sistem Operasi Windows 7
- Microsoft Visual Studio 2013
- SQLite Database

3. Kebutuhan data, meliputi:
   - Data gejala dari penyakit pneumonia, bronkitis akut, dan URTI (*Upper Respiratory Tract Infection*).
   - Data hasil observasi kasus tentang pneumonia, bronkitis akut, dan URTI (*Upper Respiratory Tract Infection*) dengan pakar.

3.4 Perancangan Aplikasi

Tahap perancangan aplikasi bertujuan untuk mengubah informasi awal yang diperoleh pada tahapan analisis menjadi model yang akan digunakan dalam pengimplementasian aplikasi. Pada tahapan ini akan menjelaskan perancangan pada pembuatan aplikasi yang menerapkan metode Dempster Shafer dalam mendeteksi penyakit infeksi saluran pernafasan dengan optimasi nilai *belief* menggunakan algoritma genetika. Diagram blok perancangan sistem ini dapat dilihat pada Gambar 3.2.



Gambar 3.2 Diagram blok desain arsitektur sistem

Pada Gambar 3.2 terdapat tiga proses utama dalam aplikasi ini, yaitu:

1. Input

   Input pada aplikasi ini adalah parameter algoritma genetika yang akan digunakan untuk proses pembangkitan nilai belief dan gejala yang berkaitan dengan penyakit infeksi saluran pernafasan. Parameter algoritma genetika yang dibutuhkan adalah jumlah populasi (*popsize*),

probabilitas crossover (*Cr*), probabilitas mutasi (*Mr*), dan jumlah iterasi/generasi.

2. Proses

   Aplikasi ini terdiri dari dua proses, yaitu proses pembangkitan nilai belief dengan algoritma genetika dan proses deteksi penyakit dengan metode Dempster Shafer. Proses pembangkitan nilai belief dengan algoritma genetika ini dilakukan sebagai pengganti nilai belief yang diberikan pakar dengan tujuan agar nilai belief yang dihasilkan lebih optimal. Setelah didapatkan nilai belief yang optimal sistem dapat melakukan proses diagnosa penyakit menggunakan metode Dempster Shafer.

3. Output

   Hasil dari aplikasi ini adalah nilai belief yang optimal dan hasil diagnosa penyakit.

3.5 Implementasi Aplikasi

Implementasi aplikasi ini dilakukan dengan mengacu pada perancangan sistem yang telah dibuat. Langkah-langkah yang dilakukan dalam implementasi anatara lain:

1. Implementasi *interface* (antarmuka) berupa halaman yang menerima inputan dari pengguna.
2. Implementasi metode *Dempster-Shafer* dengan optimasi nilai *belief* menggunakan algortima genetika ke dalam bahasa pemrograman C# dan *Database Management System* yang digunakan adalah SQLite.
3. Output yang diperoleh berupa informasi hasil identifikasi dan saran penanganan atas kemungkinan penyakit yang diderita.

3.6 Pengujian Aplikasi

Pada tahap ini dilakukan pengujian terhadap keberhasilan aplikasi yang telah dibangun. Pengujian bertujuan untuk mengetahui apakah aplikasi telah mampu bekerja sesuai spesifikasi dari kebutuhan yang melandasinya. Serta melakukan analisis terhadap hasil perhitungan sistem yang menggunakan algoritma genetika untuk optimasi nilai *belief* dan menemukan nilai probabilitas crossover, probabilitas mutasi, jumlah populasi, dan jumlah generasi yang menghasilkan *fitness* terbesar. Pengujian yang dilakukan meliputi pengujian parameter algoritma genetika dan pengujian akurasi.

1. Pengujian parameter algoritma genetika meliputi pengujian jumlah populasi, pengujian jumlah generasi, serta pengujian probabilitas crossover dan mutasi. Pengujian yang dilakukan adalah membandingkan perubahan nilai parameter terhadap nilai *fitness* yang dihasilkan. Berikut adalah pengujian parameter algoritma genetika.
   a. Pengujian untuk mengetahui pengaruh perubahan ukuran populasi terhadap nilai *fitness* maksimum yang dihasilkan. Proses pengujian

dilakukan dengan mengubah ukuran populasi 10, 20, 40, 60, 80, 100, 120, 140, dan 160 populasi. Skenario uji coba dilakukan jumlah generasi sebanyak 20 generasi dengan nilai *Cr* dan *Mr* 0.5. Pada tiap pengujian dilakukan 10 kali percobaan lalu diambil nilai rata-rata *fitness* yang kemudian akan dibandingkan dengan setiap nilai *fitness* populasi yang lain untuk dianalisa. Tabel uji coba banyaknya populasi dapat dilihat pada Tabel 3.3

Tabel 3.3 Skenario uji coba populasi

Populasi	Nilai *Fitness* Pengujian ke-i										Rata-rata *fitness*
	1	2	3	4	5	6	7	8	9	10	
10											
20											
40											
60											
80											
100											
120											
140											
160											

b. Pengujian untuk mengetahui pengaruh perubahan jumlah generasi terhadap besar nilai *fitness* maksimum yang dihasilkan. Proses pengujian dilakukan dengan mengubah nilai generasi 10, 20, 50, 75, 100, 110 dan 120 generasi. Pada setiap pengujian satu jumlah generasi dilakukan 10 kali percobaan, lalu diambil nilai rata-rata *fitness* yang kemudian akan dibandingkan dengan nilai generasi lainnya. Tabel pengujian jumlah generasi ditunjukkan pada Tabel 3.4.

Tabel 3.4 Skenario uji coba generasi

Generasi	Nilai *Fitness* Pengujian ke-i										Rata-rata *fitness*
	1	2	3	4	5	6	7	8	9	10	
10											
20											
50											
75											
100											
110											
120											

c. Pengujian untuk mengetahui pengaruh perubahan probabilitas crossover dan probabilitas mutasi terhadap nilai *fitness* yang dihasilkan serta mengetahui berapa besar nilai probabilitas crossover dan probabilitas mutasi terbaik. Nilai kombinasi dari *Cr* dan *Mr* yang digunakan dalam

pengujian berskala 0,1 hingga 0,9. Pada pengujian ini populasi dan generasi yang digunakan adalah jumlah populasi dan generasi yang terbaik pada pengujian sebelumnya. Dalam setiap kali pengujian dilakukan 10 kali percobaan lalu diambil nilai rata-rata *fitness*nya. Tabel uji coba kombinasi *crossover rate* dan *mutation rate* dapat dilihat pada Tabel 3.5.

Tabel 3.5 Skenario uji coba kombinasi *Crossover rate* dan *Mutation rate*

Cr : *Mr*	Nilai *Fitness*										Rata-rata *fitness*
	Pengujian ke-i										
	1	2	3	4	5	6	7	8	9	10	
0.1:0.9											
0.2:0.8											
0.3:0.7											
0.4:0.6											
0.5:0.5											
0.6:0.4											
0.7:0.3											
0.8:0.2											
0.9:0.1											

2. Pengujian akurasi dilakukan dengan menggunakan nilai *belief* yang mempunyai *fitness* paling tinggi. Dalam pengujian akurasi disediakan beberapa kasus uji yang akan diidentifikasi oleh seorang pakar dan aplikasi. Hasil diagnosa aplikasi akan dicocokkan dengan hasil diagnosa pakar untuk mendapatkan nilai akurasi. Rumus perhitungan akurasi yang dilakukan adalah sebagai berikut:

$$Akurasi = \frac{Jumlah\ data\ akurat}{Jumlah\ seluruh\ data} \times 100\% \quad (3.1)$$

Pengujian akurasi dilakukan sebanyak 10 kali, kemudian dirata-rata. Tabel pengujian akurasi ditunjukkan pada Tabel 3.6.

Tabel 3.6 Tabel pengujian akurasi

Percobaan ke-*i*	*Fitness*
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
Rata-rata	

3.7 Pengambilan Kesimpulan

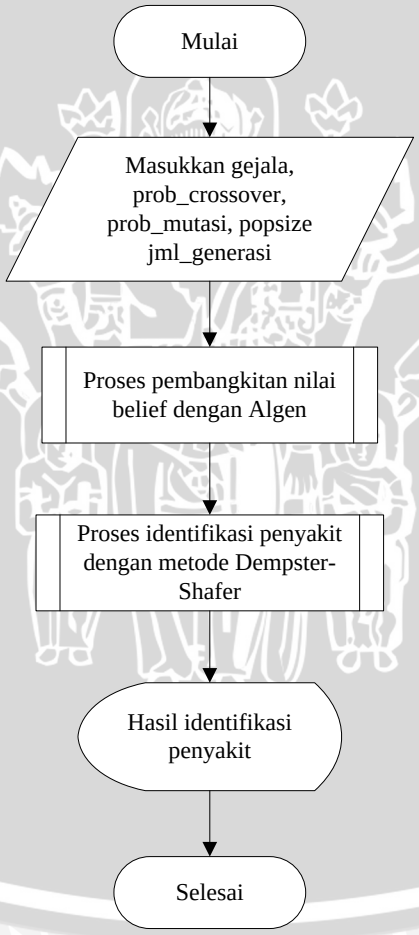
Kesimpulan dilakukan setelah semua tahap perancangan, implementasi, dan pengujian telah selesai dilakukan. Kesimpulan diambil dari hasil pengujian dan hasil analisis metode *Dempster-Shafer* dan Algoritma Genetika yang diterapkan. Tahap terakhir dari penulisan adalah saran yang dimaksudkan untuk memperbaiki kesalahan-kesalahan yang terjadi dan menyempurnakan penulisan serta memberikan pertimbangan aras pengembangan aplikasi selanjutnya.

BAB 4 PERANCANGAN

Pada bab ini akan membahas mengenai perancangan sistem yang meliputi perancangan sistem, perhitungan manual dan perancangan antarmuka.

4.1 Perancangan Sistem

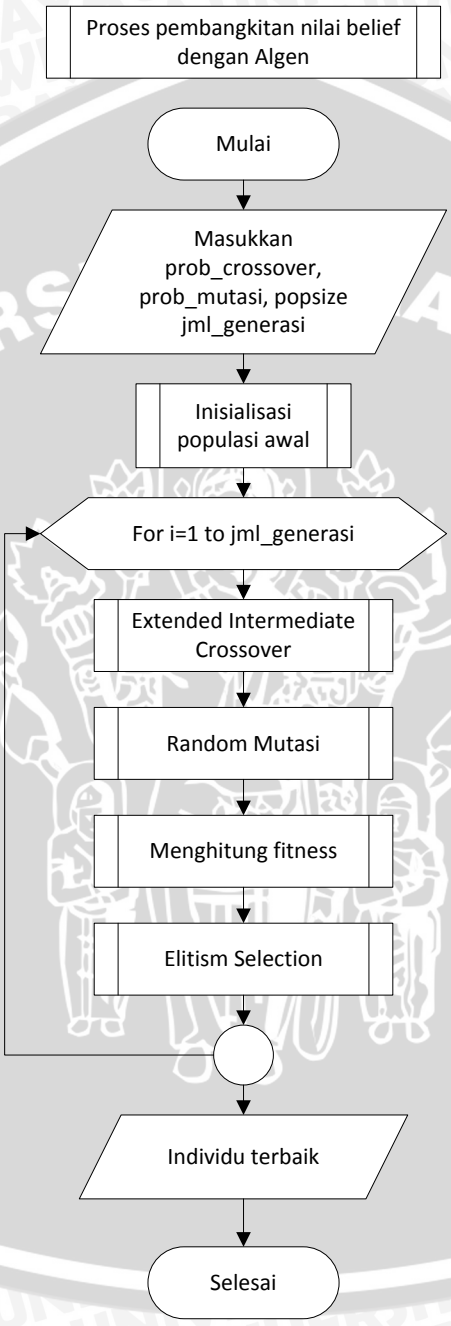
Perancangan sistem ini menggunakan notasi diagram alir untuk menggambarkan proses dalam penerapan metode Dempster Shafer dengan optimasi nilai *belief* menggunakan algoritma genetika. Sistem ini terdiri dari dua proses, yaitu proses pembangkitan nilai *belief* dengan algoritma genetika dan proses mendiagnosa menggunakan metode Dempster Shafer. Gambar 4.1 menunjukkan diagram alir penerapan metode Dempster-Shafer untuk deteksi penyakit infeksi saluran pernafasan dengan pembangkitan nilai *belief* menggunakan algoritma genetika.



Gambar 4.1 Diagram alir perancangan sistem

4.1.1 Proses Pembangkitan Nilai *Belief*

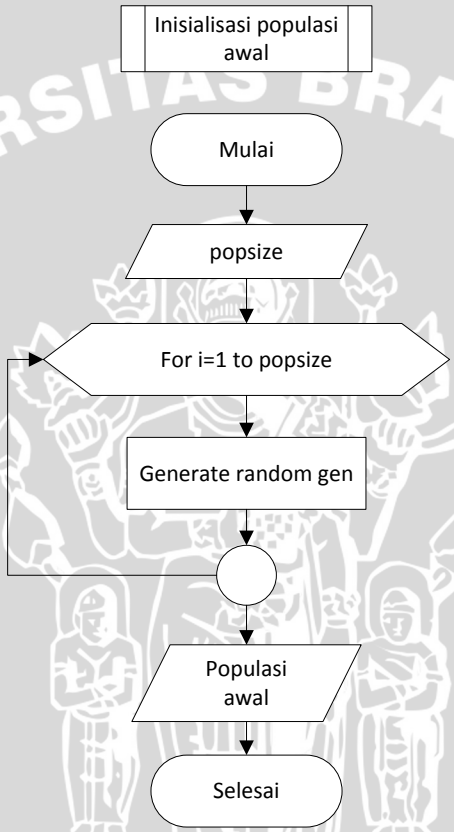
Sistem ini diawali dengan proses pembangkitan nilai *belief* menggunakan algoritma genetika. Algoritma genetika terdiri dari beberapa proses, diantaranya inisialisasi populasi awal, evaluasi *fitness*, proses reproduksi (*crossover* dan mutasi) dan seleksi. Proses-proses tersebut ditunjukkan pada gambar berikut.



Gambar 4.2 Diagram alir pembangkitan nilai *belief*

4.1.2 Inisialisasi Populasi Awal

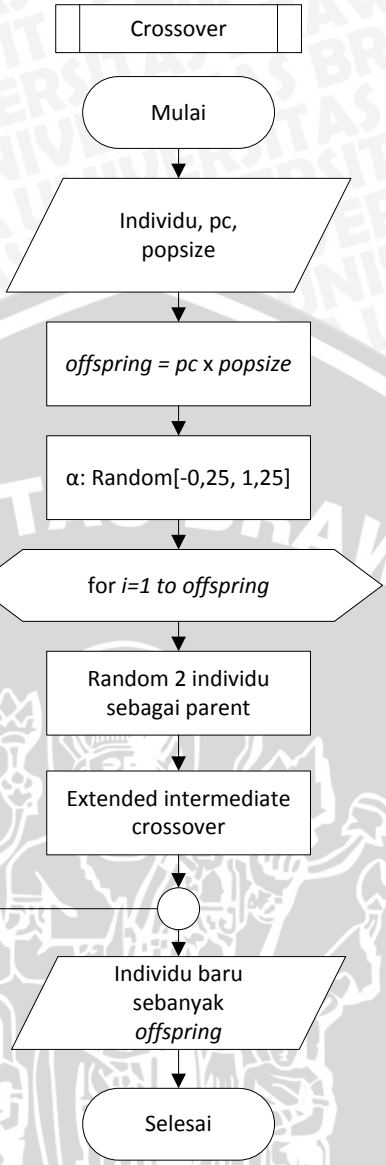
Inisialisasi populasi awal dilakukan dengan membangkitkan bilangan random sebanyak nilai populasi (popsize) yang diinputkan pengguna. Masing-masing individu akan mempunyai gen yang berisi angka acak yang merupakan bilangan real bertipe *floating point* yang dibangkitkan secara random oleh aplikasi. Random yang dibangkitkan dibatasi rentang nilainya, yaitu 0-1 karena dalam Dempster-Shafer nilai *belief* yang bernilai 0 mengindikasikan tidak ada *evidence,* dan jika bernilai 1 menunjukkan adalanya kepastian (Sulistyohati & Hidayat, 2008). Diagram alir untuk proses inisialisasi populasi awal akan ditunjukkan dalam Gambar 4.3.



Gambar 4.3 Diagram alir proses inisialisasi populasi awal

4.1.3 Crossover

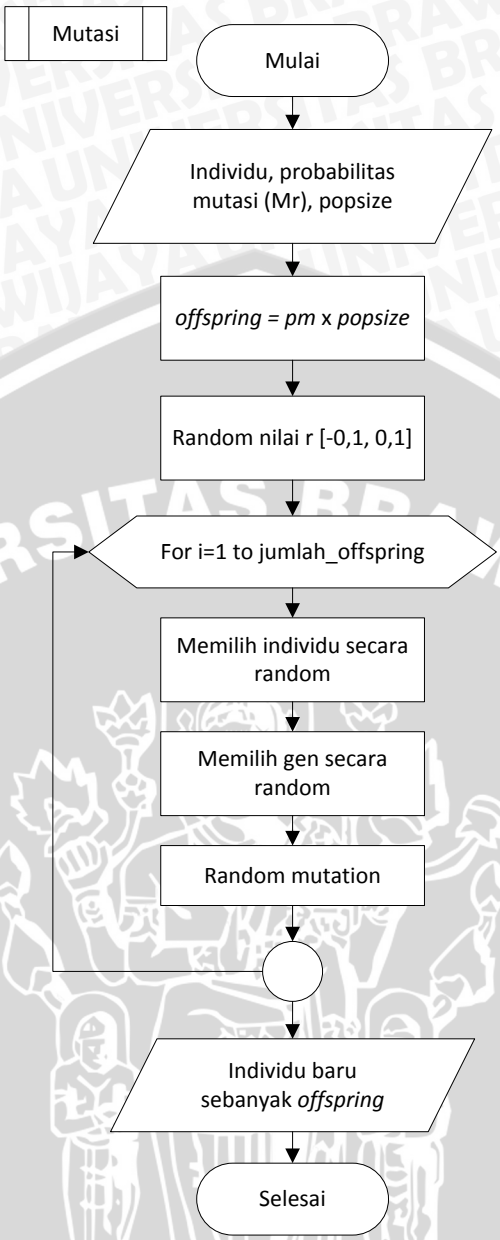
Metode crossover yang digunakan adalah *extended intermediate crossover.* Proses awal metode crossover ini adalah menentukan jumlah *offspring* yang akan dihasilkan dengan perhitungan probabilitas crossover (Cr) dikalikan dengan ukuran populasi (*popsize*). Selanjutnya membangkitkan alpha (α) secara random dan pemilihan dua individu (*parent*). Setiap *offspring* dibangkitkan menggunakan persamaan 2.4 dan persamaan 2.5. Diagram alir proses crossover ditunjukkan dalam Gambar 4.4.



Gambar 4.4 Diagram alir proses crossover

4.1.4 Mutasi

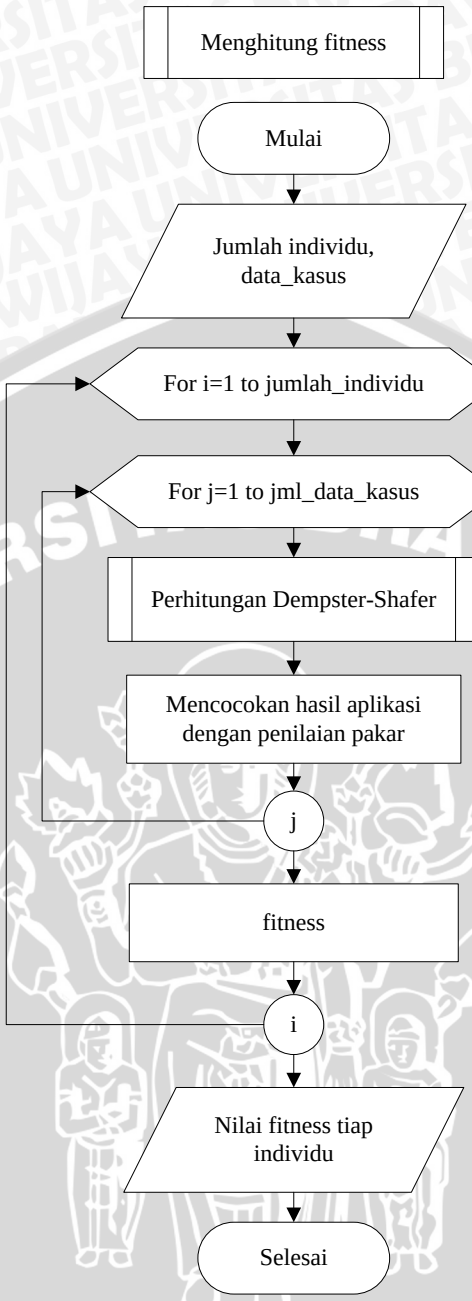
Proses selanjutnya adalah mutasi. Metode mutasi yang digunakan adalah *random mutation* yang dilakukan dengan mengubah satu atau lebih gen dari kromosom terpilih. Proses awal yang dilakukan adalah menentukan jumlah offspring yang dihasilkan dengan perhitungan probabilitas mutasi (*Mr*) dikalikan dengan populasi (*popsize*). Selanjutnya membangkitkan nilai r secara random dan memilih satu individu (*parent*). Setiap *offspring* dibangkitkan dengan persamaan, yaitu Persamaan 2.6. Diagram alir proses mutasi ditunjukkan dalam Gambar 4.5



Gambar 4.5 Diagram alir proses mutasi

4.1.5 Menghitung Nilai *Fitness*

Proses berikutnya adalah menghitung nilai *fitness.* Proses perhitungan nilai *fitness* digunakan untuk mengevaluasi setiap individu yang terdapat dalam populasi. Perhitungan nilai *fitness* ini menggunakan beberapa kasus yang sudah dinilai kebenarannya oleh pakar. Kasus dalam perhitungan nilai *fitness* ini menggunakan data latih yang ditunjukkan pada lampiran C. Diagram alir proses perhitungan nilai *fitness* ditunjukkan dalam Gambar 4.6.



Gambar 4.6 Diagram alir proses perhitungan nilai *fitness*

Proses perhitungan nilai *fitness* dilakukan terhadap nilai *belief* yang telah dibangkitkan pada proses sebelumnya. Nilai *belief* pada suatu individu akan diuji melalui sejumlah kasus menggunakan metode Dempster Shafer. Dari perhitungan Dempster Shafer tersebut akan menghasilkan diagnosa penyakit. Kemudian diagnosa perhitungan Dempster Shafer tersebut dicocokkan dengan hasil diagnosa pakar. Nilai *fitness* didapatkan dengan rumus berikut:

$$fitness = \frac{Klasifikasi\ benar}{jml_data_kasus}$$

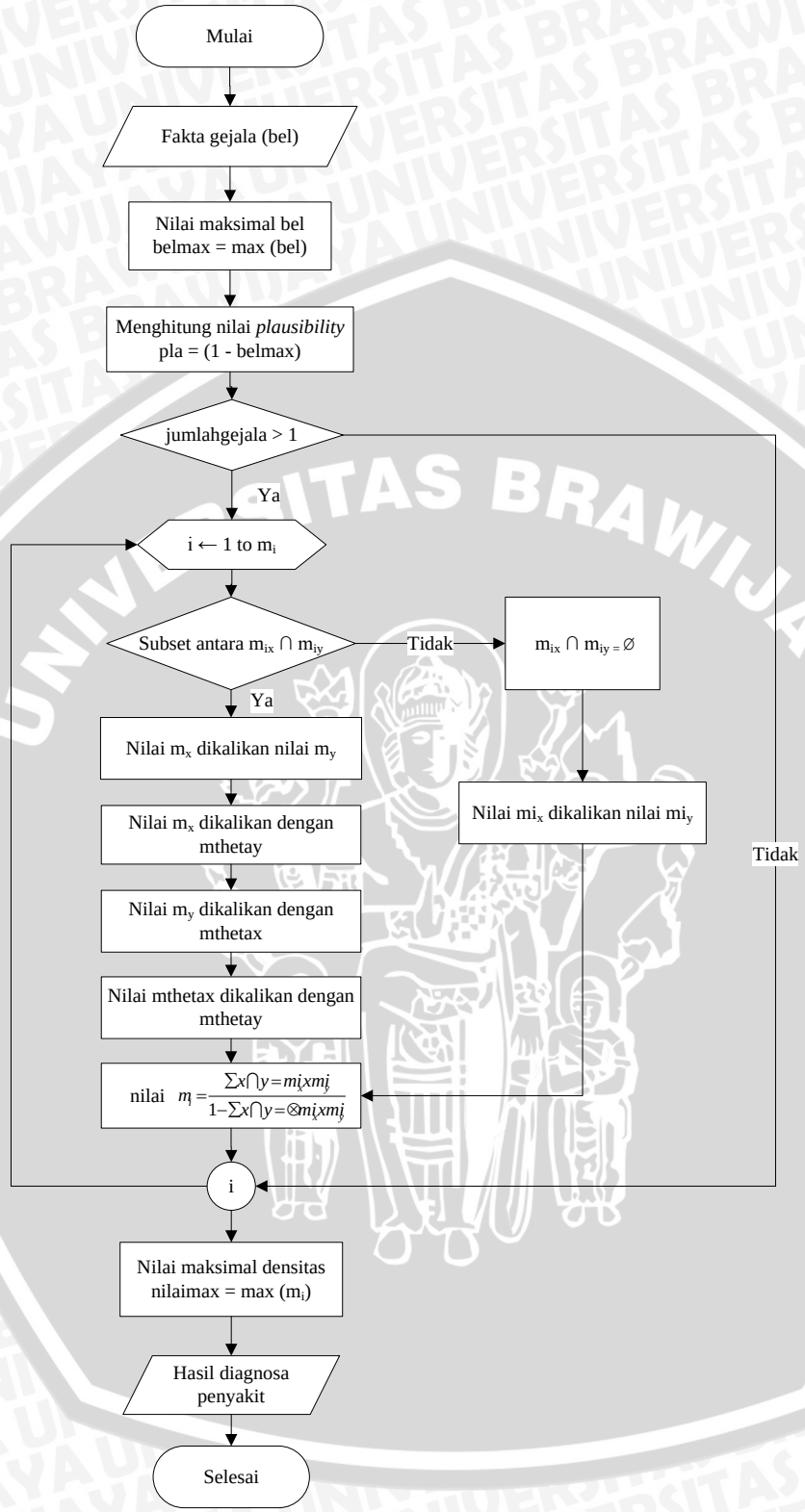
4.1.6 Perhitungan Dempster Shafer

Dalam aplikasi ini proses perhitungan Demspter-Shafer dilakukan dua kali, yaitu saat proses perhitungan nilai *fitness* dan proses diagnosa penyakit. Dalam proses perhitungan nilai *fitness*, metode Demspter Shafer digunakan untuk mengukur akurasi nilai *belief* dari individu yang dibentuk dengan membandingkan hasil perhitungan Demspter Shafer dengan hasil diagnosa pakar. Sedangkan, dalam proses diagnosa penyakit dilakukan saat nilai *belief* yang terbaik telah didapatkan. Proses perhitungan Dempster Shafer pun dapat dilakukan dengan gejala yang dimasukkan oleh pengguna.

Perhitungan dalam Dempster-Shafer dimulai dengan memasukkan nilai *belief* tiap gejala penyakit ke dalam basis data sebagai dasar perhitungan. Kemudian dari fakta gejala yang dimasukkan oleh pengguna diperoleh kemungkinan penyakit dan nilai densitasnya untuk kemudian dihitung nilai *belief* dan *plausibility.* Jika nilai sudah diperoleh dan hanya terdapat satu gejala saja, maka dari hasil kemungkinan nama penyakit yang sesuai ditentukan berdasarkan nilai *belief* tertinggi.

Jika gejala yang dimasukkan lebih dari satu gejala, maka hasil dari nama penyakit dan nilai *belief, plausibility* gejala pertama akan sementara disimpan. Untuk gejala kedua dilakukan tahapan yang sama dengan gejala pertama dan hasilnya juga disimpan sementara. Setelah diperoleh nilai dua gejala maka dapat dilakukan perhitungan untuk mencari nilai densitas gabungan atau nilai densitas ketiga yang berasal dari gejala 1 dan gejala 2 serta kemungkinan nama penyakit yang dimasukkan dalam persamaan 2-3 teori *Dempster-Shafer***.** Dari hasil nilai densitas ketiga akan diperoleh kemungkinan nama penyakit dengan nilai densitas baru yang kemudian disimpan.

Setelah diperoleh nilai densitas ketiga dan masih terdapat gejala lain yang dimasukkan pengguna maka dilakukan perhitungan nilai densitas gabungan baru antara nilai densitas ketiga dengan nilai dari gejala keempat seperti pada tahapan sebelumnya. Proses perhitungan tersebut akan terus berulang sebanyak gejala yang dimasukkan pengguna. Jika semua gejala sudah selesai dihitung, maka pengambilan kesimpulan diperoleh dari hasil nilai densitas gabungan yang paling terakhir dihitung. Diagram alir proses perhitungan dengan metode *Dempster-Shafer* dapat dilihat pada Gambar 4.7.

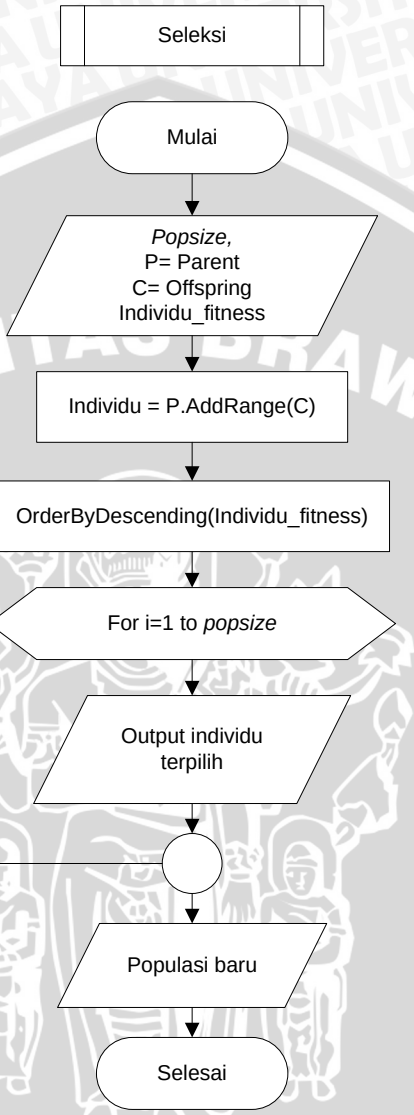


Gambar 4.7 Diagram alir algoritma perhitungan *Dempster-Shafer*

4.1.7 Seleksi

Metode seleksi yang digunakan adalah metode *elitism* yang bekerja dengan mengumpulkan semua individu *parent*(P) dan *offspring*(C) dalam satu

penampungan. Individu dalam penampungan ini akan diurutkan dari yang terbesar ke yang paling kecil. Kemudian akan dipilih individu terbaik sebanyak jumlah *popsize.* Individu ini yang akan menjadi populasi baru untuk diproses digenerasi selanjutnya. Gambar 4.8 menunjukkan diagram alir proses seleksi.



Gambar 4.8 Diagram alir proses seleksi

4.1.8 Kondisi Berhenti

Algoritma Genetika digunakan untuk mendapatkan individu terbaik yang memiliki nilai *belief* dengan akurasi tiap kromosomnya. Sehingga diharapkan generasi baru yang muncul bisa lebih baik dari generasi sebelumnya. Selama kondisi berhenti belum terpenuhi, maka generasi baru akan terus dimunculkan. Berikut dua macam kondisi berhenti yang digunakan dalam penelitian ini untuk membatasi generasi yang muncul sehingga waktu komputasinya tidak terlalu lama: (Zamani, Amaliah, & Munif, 2012)

1. Kondisi berhenti yang pertama adalah maksimum jumlah generasi, yaitu berhenti ketika mencapai generasi ke-*n.* Nilai n dapat ditentukan pengguna.

2. Kondisi berhenti kedua adalah konvergensi, yaitu berhenti ketika nilai rata-rata *fitness* pada 3 generasi terakhir tidak berubah atau konvergen.

4.2 Perhitungan Manual

Pada subbab ini menjelaskan alur dan perhitungan secara manual dari penerapan metode dempster shafer untuk deteksi penyakit pernafasan dengan pembangkitan nilai *belief* menggunakan algoritma genetika. Tabel 3.3 merupakan contoh suatu kasus dimana masukkan dari pengguna adalah gejala serta .

Tabel 4.1 Data uji perhitungan manual

Gejala	Cough Sputum: Jernih, putih More than 10 days Chest pain Dyspnea (sesak napas)
Popsize	5
Crossover rate	0.8
Mutation rate	0.2
Iterasi	100

4.2.1 Pembangkitan Awal Nilai *Belief* Dengan Algoritma Genetika

Tahap pertama dalam aplikasi ini adalah membangkitkan nilai *belief* dengan algoritma genetika. Di bawah ini merupakan tahap-tahap dalam pembangkitan nilai *belief* diantaranya adalah representasi kromosom, inisialisasi populasi awal, menghitung nilai *fitness*, crossover, mutasi, dan seleksi.

4.2.1.1 Representasi Kromosom

Pada penelitian ini kromosom direpresentasikan menggunakan pengkodean rill (*real code*). Kromosom dibangkitkan secara random dalam interval 0.0 sampai 1.0. Inisialisasi dari kromosom direpresentasikan kedalam bentuk *array* yang berisi gejala dari 3 penyakit. Inisialisasi kromosom ini dibentuk berdasarkan gejala dan penyakit yang telah dijabarkan pada sub bab akusisi pengetahuan dan ditunjukan pada Gambar 4.8.

Gejala 1			Gejala 2			Gejala …		
P	B	U	P	B	U	P	B	U
0	0.514	0.572	0.372	0.512	0	0.582	0	0
Gen-1	Gen-2	Gen-3	Gen-4	Gen-5	Gen-6	Gen-7	Gen-8	Gen-n

Gambar 4.9 Inisialisasi kromosom

Representasi kromosom pada Gambar 4.9 merupakan proses insialisasi dari komponen sebuah kromosom. Komponen dari kromosom tersebut berupa nilai

belief dari setiap gejala pada tiga penyakit yang dibentuk sesuai akusisi pengetahuan, dimana terdapat beberapa nilai yang kosong untuk gejala yang tidak mempengaruhi suatu penyakit. Terdapat 17 gejala dari 3 penyakit (P=pneumonia, B=bronkitis akut, dan U=URTI) sehingga panjang dari representasi sebuah kromosom pada perancangan yaitu 51 gen.

4.2.1.2 Inisialisasi Populasi Awal

Hal pertama yang dilakukan adalah membangkitkan secara random populasi awal. Pada perhitungan manual ini menggunakan popSize=5. Tabel berikut merupakan inisialisasi populasi awal.

Tabel 4.2 Inisialisasi populasi awal

P1	G1			G2			G3		
	0.534	0.875	0.53	0.238	0.374	0.573	0.468	0	0
	G4			G5			G6		
	0.375	0.583	0.384	0	0.347	0.474	0.472	0.582	0
	G7			G8			G9		
	0	0.285	0.372	0	0.514	0.572	0.372	0.512	0
	G10			G11			G12		
	0.582	0	0	0	0.352	0	0	0	0.235
	G13			G14			G15		
	0.682	0.235	0	0	0.523	0	0.5325	0	0
	G16			G17					
	0.5244	0.483	0	0	0.602	0.662			
P2	G1			G2			G3		
	0.155	0.682	0.352	0.706	0.243	0.923	0.729	0	0
	G4			G5			G6		
	0.654	0.198	0.233	0	0.877	0.294	0.683	0.153	0
	G7			G8			G9		
	0	0.542	0.242	0	0.269	0.664	0.756	0.426	0
	G10			G11			G12		
	0.233	0	0	0	0.738	0	0	0	0.154
	G13			G14			G15		
	0.337	0.768	0	0	0.352	0	0.673	0	0
	G16			G17					
	0.223	0.822	0	0	0.322	0.682			
P3	G1			G2			G3		
	0.821	0.183	0.313	0.127	0.731	0.317	0.287	0	0
	G4			G5			G6		
	0.833	0.871	0.271	0	0.217	0.831	0.189	0.725	0
	G7			G8			G9		
	0	0.442	0.627	0	0.713	0.286	0.258	0.877	0
	G10			G11			G12		
	0.619	0	0	0	0.294	0	0	0	0.679

	G13			G14			G15		
	0.912	0.228	0	0	0.517	0	0.312	0	0
	G16			G17					
	0.927	0.432	0	0	0.781	0.219			
P4	G1			G2			G3		
	0.427	0.721	0.528	0.812	0.383	0.422	0.731	0	0
	G4			G5			G6		
	0.218	0.523	0.719	0	0.832	0.318	0.842	0.273	0
	G7			G8			G9		
	0	0.423	0.324	0	0.812	0.527	0.572	0.322	0
	G10			G11			G12		
	0.423	0	0	0	0.574	0	0	0	0.351
	G13			G14			G15		
	0.485	0.846	0	0	0.512	0	0.711	0	0
	G16			G17					
	0.128	0.284	0	0	0.284	0.534			
P5	G1			G2			G3		
	0.616	0.223	0.823	0.238	0.823	0.523	0.532	0	0
	G4			G5			G6		
	0.723	0.311	0.412	0	0.721	0.532	0.237	0.823	0
	G7			G8			G9		
	0	0.712	0.214	0	0.121	0.812	0.628	0.238	0
	G10			G11			G12		
	0.628	0	0	0	0.338	0	0	0	0.823
	G13			G14			G15		
	0.396	0.283	0	0	0.812	0	0.217	0	0
	G16			G17					
	0.311	0.418	0	0	0.346	0.772			

4.2.1.3 Perhitungan Nilai Fitness

Setelah didapatkan populasi awal, proses selanjutnya adalah menghitung *fitness*. Proses perhitungan *fitness* menggunakan rumus berikut:

$$fitness = \frac{Klasifikasi\ benar}{jml_data_kasus}$$

Klasifikasi yang benar didapatkan dari proses pendeteksian penyakit infeksi saluran pernafasan menggunakan Dempster-Shafer dengan nilai *belief* setiap individu pada populasi. Berikut ini adalah salah satu perhitungan kasus dari 36 kasus yang ada menggunakan individu P1.

1. Gejala 1: Cough/ Batuk (G2)
   Gejala ini merupakan gejala dari penyakit berikut dengan densitas:
   Pneumnonia = 0.238
   Bronkitis Akut = 0.374

URTI = 0.573

Untuk m_1 nilai densitas yang dipilih adalah yang tertinggi, maka

m_1{P,B,U} = 0.573

m_1{θ} = 1 - m_1{P,B,U}

= 1 - 0.573

= 0.427

2. Gejala : Sputum putih/jernih (G11)

Gejala ini merupakan gejala dari penyakit berikut dengan densitas:

Bronkitis Akut = 0.352

Untuk m_1 nilai densitas yang dipilih adalah yang tertinggi, maka

m_2{B} = 0.352

m_2{θ} = 1 - m_2{B}

= 1 - 0.352

= 0.648

Selanjutnya menghitung nilai densitas baru untuk kombinasi dengan fungsi densitas m_3 yang dapat dilihat pada Tabel 4.3.

Tabel 4.3 Aturan kombinasi untuk m_3

m1		m2			
		{B}	0.352	{θ}	0.648
{P,B,U}	0.573	{B}	0.202	{P,B,U}	0.371
{θ}	0.427	{B}	0.150	{θ}	0.277

Sehingga dapat dihitung

$$m_3\{B\} = \frac{0.202 + 0.150}{1 - 0} = 0.352$$

$$m_3\{P,B,U\} = \frac{0.371}{1 - 0} = 0.371$$

$$m_3\{\theta\} = \frac{0.277}{1 - 0} = 0.277$$

3. Gejala 3: Sore throat (G5)

Gejala ini merupakan gejala dari penyakit berikut dengan densitas:

Bronkitis Akut = 0.347

URTI = 0.474

Untuk m_4 nilai densitas yang dipilih adalah yang tertinggi, maka

m_4{B,U} = 0.474

m_4{θ} = 1 - m_4{B,U}

= 1 – 0.474

= 0.526

Tabel 4.4 Aturan kombinasi untuk m_5

m3		m4			
		{B,U}	0.474	{θ}	0.526
{P,B,U}	0.371	{B,U}	0.176	{P,B,U}	0.195
{B}	0.352	{B}	0.167	{B}	0.185
{θ}	0.277	{B,U}	0.13115	{θ}	0.14554

Sehingga dapat dihitung

$$m_5\{B,U\} = \frac{0.176 + 0.13115}{1 - 0} = 0.30715$$

$$m_5\{B\} = \frac{0.167 + 0.185}{1 - 0} = 0.352$$

$$m_5\{P,B,U\} = \frac{0.195}{1 - 0} = 0.195$$

$$m_5\{\theta\} = \frac{0.14554}{1 - 0} = 0.1454$$

4. Gejala 4: 2-7 days (G17)

Gejala ini merupakan gejala dari penyakit berikut dengan densitas:

Bronkitis Akut = 0.602

URTI = 0.662

Untuk m_4 nilai densitas yang dipilih adalah yang tertinggi, maka

$m_6\{B,U\} = 0.662$

$m_6\{\theta\} = 1 - m_6\{B,U\}$

$= 1 - 0.662$

$= 0.338$

Tabel 4.5 Aturan kombinasi m_7

m5		m6			
		{B,U}	0.662	{θ}	0.338
{P,B,U}	0.195	{B,U}	0.129	{P,B,U}	0.066
{B,U}	0.307	{B,U}	0.203	{B,U}	0.104
{B}	0.352	{B}	0.233	{B}	0.119
{θ}	0.146	{B,U}	0.09635	{θ}	0.04919

Sehingga dapat dihitung

$$m_7\{B,U\} = \frac{0.129 + 0.203 + 0.09635 + 0.104}{1 - 0} = 0.53279$$

$$m_7\{B\} = \frac{0.233 + 0.119}{1 - 0} = 0.352$$

$$m_7\{P,B,U\} = \frac{0.066}{1 - 0} = 0.066$$

$$m_7\{\theta\} = \frac{0.04919}{1 - 0} = 0.04919$$

Dari hasil perhitungan dengan metode Dempster-Shafer, nilai densitas paling tinggi adalah penyakit Bronkitis akut dan URTI. Kemudian hasil diagnosa perhitungan ini dibandingkan dengan hasil diagnosa pakar. Diketahui bahwa hasil perhitungan ini sesuai dengan diagnosa pakar, maka akurasi hasil kasus ini bernilai 1.

Setelah semua kasus sudah dihitung selanjutnya adalah menghitung nilai akurasi yang digunakan sebagai nilai *fitness*. Berikut ini pada Tabel 4.6 adalah nilai *fitness* dari salah satu individu dari 5 individu, yaitu individu P1.

Tabel 4.6 Hasil Akurasi Individu 1

	Pakar	Manual	Akurasi Hasil
Kasus 1	U	B,U	1
Kasus 2	U	B,U	1
Kasus 3	B	B	1
Kasus 4	U	U	1
Kasus 5	P	P,B	1
Kasus 6	P	P	1
Kasus 7	B	B	1
Kasus 8	P	P,B	1
Kasus 9	P	P	1
Kasus 10	P	P	1
Kasus 11	B	B	1
Kasus 12	B	P	0
Kasus 13	B	B	1
Kasus 14	B	B	1
Kasus 15	P	P,B	1
Kasus 16	B	B,U	1
Kasus 17	P	P	1
Kasus 18	P	B	0
Kasus 19	P	B	0
Kasus 20	B/U	B,U	1
Kasus 21	P	B	0
Kasus 22	B/U	B	1
Kasus 23	U	B,U	1
Kasus 24	P/B	P	1
Kasus 25	P/B	P	1
Kasus 26	B	B	1
Kasus 27	P	P	1
Kasus 28	P	P	1
Kasus 29	P	P	1
Kasus 30	U	U	1
Kasus 31	P	P	1
Kasus 32	P	P	1
Kasus 33	B	B	1
Kasus 34	P	P	1
Kasus 35	P	P	1
Kasus 36	P	P	1
		Total benar:	32

Nilai 1 menunjukkan bahwa hasil dari sistem sesuai dengan penilaian pakar, sedangkan nilai 0 menunjukkan bahwa hasil dari sistem tidak sesuai dengan penilaian pakar. Kemudian nilai *fitness* dapat dihitung sebagai berikut:

$$fitness = \frac{32}{36} = 0.8889$$

Tabel 4.7 Hasil perhitungan nilai *fitness*

No	Individu	*Fitness*
1	P1	0.94444444
2	P2	0.83333333
3	P3	0.86111111
4	P4	0.83333333
5	P5	0.86111111

4.2.1.4 Crossover

Metode crossover yang digunakan adalah *extended intermediate crossover.* Metode ini dilakukan dengan mengkombinasikan nilai gen dari kedua induk dengan rumus: $C_1 = P_1 + \alpha(P_2 - P_1)$ dan $C_2 = P_2 + \alpha(P_1 - P_2)$ dengan nilai α diambil secara acak dalam range [-0.25, 1.25] (Mahmudy, 2013).

Hal pertama yan dilakukan pada proses *crossover* adalah menghitung jumlah offspring yang akan dihasilkan. Pada perhitungan ini *crossover rate* yang digunakan adalah 0,8 dengan popSize sejumlah 5.

$$Jumlah\ offspring = 0{,}8 \times 5$$
$$= 4$$

Sehingga pada perhitungan ini jumlah *offspring* yang dihasilkan adalah 4 *offspring* dimana setiap proses crossover menghasilkan dua *offspring* dari dua *parent.* Contoh perhitungan akan ditunjukkan pada tabel berikut.

Tabel 4.8 Perhitungan *Crossover*

Parent	Chromosome								
P2	G1			G2			G3		
	0.155	0.682	0.352	0.706	0.243	0.923	0.729	0	0
	G4			G5			G6		
	0.654	0.198	0.233	0	0.877	0.294	0.683	0.153	0
	G7			G8			G9		
	0	0.542	0.242	0	0.269	0.664	0.756	0.426	0
	G10			G11			G12		
	0.233	0	0	0	0.738	0	0	0	0.154
	G13			G14			G15		
	0.337	0.768	0	0	0.352	0	0.673	0	0
	G16			G17					
	0.223	0.822	0	0	0.322	0.682			
P5	G1			G2			G3		
	0.616	0.223	0.823	0.238	0.823	0.523	0.532	0	0
	G4			G5			G6		
	0.723	0.311	0.412	0	0.721	0.532	0.237	0.823	0
	G7			G8			G9		

	0	0.712	0.214	0	0.121	0.812	0.628	0.238	0
	G10			G11			G12		
	0.628	0	0	0	0.338	0	0	0	0.823
	G13			G14			G15		
	0.396	0.283	0	0	0.812	0	0.217	0	0
	G16			G17					
	0.311	0.418	0	0	0.346	0.772			
C1	G1			G2			G3		
	0.449	0.775	0.784	0.233	0.334	0.431	0.776	0.000	0.000
	G4			G5			G6		
	0.702	0.244	0.294	0.000	0.696	0.548	0.332	0.854	0.000
	G7			G8			G9		
	0.000	0.657	0.221	0.000	0.274	0.709	0.755	0.221	0.000
	G10			G11			G12		
	0.185	0.000	0.000	0.000	0.534	0.000	0.000	0.000	0.793
	G13			G14			G15		
	0.344	0.686	0.000	0.000	0.421	0.000	0.390	0.000	0.000
	G16			G17					
	0.247	0.766	0.000	0.000	0.331	0.763			
C2	G1			G2			G3		
	0.322	0.130	0.391	0.711	0.732	1.015	0.485	0.000	0.000
	G4			G5			G6		
	0.675	0.265	0.351	0.000	0.902	0.278	0.588	0.122	0.000
	G7			G8			G9		
	0.000	0.597	0.235	0.000	0.116	0.767	0.629	0.443	0.000
	G10			G11			G12		
	0.676	0.000	0.000	0.000	0.542	0.000	0.000	0.000	0.184
	G13			G14			G15		
	0.389	0.365	0.000	0.000	0.743	0.000	0.500	0.000	0.000
	G16			G17					
	0.287	0.474	0.000	0.000	0.337	0.691			

Pada contoh perhitungan crossover tersebut, C1 dan C2 merupakan hasil *extended intermediate crossover* dari *parent* 2 dan *parent* 5 dengan nilai alpha (α) yang diambil secara acak dengan range (-0,25; 1,25). Tabel 4.9 menunjukkan nilai alpha yang digunakan untuk setiap gennya.

Tabel 4.9 Tabel nilai alpha

G1			G2			G3		
0.637	-0.204	0.918	1.010	0.157	1.229	-0.239	0.000	0.000
G4			G5			G6		
0.691	0.406	0.341	0.000	1.162	1.069	0.786	1.047	0.000
G7			G8			G9		

0.000	0.677	0.766	0.000	-0.031	0.305	0.011	1.090	0.000
G10			G11			G12		
-0.120	0.000	0.000	0.000	0.509	0.000	0.000	0.000	0.956
G13			G14			G15		
0.114	0.170	0.000	0.000	0.150	0.000	0.620	0.000	0.000
G16			G17					
0.272	0.137	0.000	0.000	0.364	0.898			

4.2.1.5 Mutasi

Metode mutasi yang digunakan adalah *random mutation*. Metode ini dilakukan dengan menambah atau mengurangi nilai gen terpilih dengan bilangan *random* yang kecil. Nilai gen *offspring* dibangkitkan dengan rumus $x'_i = x'_i + r(max_i - min_j)$ dengan range r yang digunakan adalah [-0,1; 0,1] (Mahmudy, 2013).

Hal pertama yan dilakukan pada proses mutasi adalah menghitung jumlah *offspring* yang akan dihasilkan. Pada perhitungan ini mutation_rate yang digunakan adalah 0,2 dengan popSize sejumlah 5.

$$Jumlah\ offspring = 0{,}2 \times 5$$
$$= 1$$

Sehingga pada perhitungan ini jumlah *offspring* yang dihasilkan adalah 1. Contoh perhitungan akan ditunjukkan pada tabel berikut.

Tabel 4.10 Perhitungan Mutasi

Parent	Chromosome								
P3	G1			G2			G3		
	0.821	0.183	0.313	0.127	0.731	0.317	0.287	0	0
	G4			G5			G6		
	0.833	0.871	0.271	0	0.217	0.831	0.189	0.725	0
	G7			G8			G9		
	0	0.442	0.627	0	0.713	0.286	0.258	0.877	0
	G10			G11			G12		
	0.619	0	0	0	0.294	0	0	0	0.679
	G13			G14			G15		
	0.912	0.228	0	0	0.517	0	0.312	0	0
	G16			G17					
	0.927	0.432	0	0	0.781	0.219			
C5	G1			G2			G3		
	0.821	0.183	0.313	0.127	0.731	0.317	0.287	0	0
	G4			G5			G6		
	0.833	0.871	0.271	0	0.200476	0.831	0.189	0.725	0
	G7			G8			G9		
	0	0.442	0.627	0	0.713	0.286	0.258	0.877	0
	G10			G11			G12		

	0.619	0	0	0	0.294	0	0	0	0.679
	G13			G14			G15		
	0.912	0.228	0	0	0.517	0	0.312	0	0
	G16			G17					
	0.927	0.432	0	0	0.781	0.219			

Pada perhitungan ini C5 merupakan hasil mutasi dari *parent* 3 dengan posisi gen yang terpilih adalah gen pada gejala 5 dan nilai r dipilih secara acak dalam interval -0,1 hingga 0,1 yakni -0.05508.

4.2.1.6 Seleksi

Metode seleksi yang digunakan adalah metode *elitism* yang bekerja dengan mengumpulkan semua individu dalam populasi (*parent*) dan *offspring* dalam satu penampungan. *popSize* individu terbaik dalam penampungan ini akan lolos untuk masuk dalam generasi selanjutnya. Metode seleksi ini menjamin individu yang terbaik akan selalu lolos.

Setelah melakukan proses reproduksi (*crossover* dan mutasi), proses selanjutnya adalah proses seleksi. Pada perhitungan ini menggunakan popSize = 5, sehingga terdapat 5 individu. Tabel 4.11 menunjukkan individu dengan *fitness*-nya.

Tabel 4.11 Himpunan populasi yang menjadi parent

No	Individu	*Fitness*
1	P1	0.944444444
2	P2	0.833333333
3	P3	0.861111111
4	P4	0.833333333
5	P5	0.861111111

Dari proses *crossover* dihasilkan 4 *offspring* (C_1, C_2, C_3 dan C_4) dan proses mutasi menghasilkan 1 *offspring* (C_5). Sehingga didapatkan 5 *offspring* seperti yang ditunjukkan Tabel 4.12.

Tabel 4.12 Offspring beserta parent dan *fitness*nya

No.	Offspring	Asal Parent	*Fitness*
1	C1	P2 & P5	0.76471
2	C2	P2 & P5	0.82353
3	C3	P1 & P4	0.94118
4	C4	P1 & P4	0.82353
5	C5	P3	0.88235

Selanjutnya *fitness offspring* dan *fitness parent* diurutkan dari yang paling besar ke yang paling kecil. Kemudian akan dipilih individu dengan nilai *fitness* tertinggi sebanyak *popsize* yang telah ditentukan di awal. Hasil dari proses seleksi pada perhitungan ini ditunjukkan pada Tabel 4.13.

Tabel 4.13 Individu baru

No.	$P_{(t+1)}$	Asal $P_{(t)}$	*Fitness*
1	P_1	P1	0.94118
2	P_2	P2	0.82353
3	P_3	P3	0.88235
4	P_4	C3	0.94118
5	P_5	P5	0.88235

4.2.2 Perhitungan Dempster Shafer

Perhitungan dengan algoritma genetika untuk mencari nilai *belief* yang terbaik telah dilakukan. Kemudian diketahui individu dengan *fitness* terbaik adalah individu satu (P1). Maka untuk menentukan penyakit dari gejala yang di masukkan oleh pengguna dilakukan perhitungan dengan metode Dempster-Shafer. Berikut adalah proses perhitungannya:

1. Gejala 1: Cough (G2)
   Gejala ini merupakan gejala dari penyakit berikut dengan densitas:
   Pneumnonia = 0.238
   Bronkitis Akut = 0.374
   URTI = 0.573
   Untuk m_1 nilai densitas yang dipilih adalah yang tertinggi, maka
   $m_1\{P,B,U\} = 0.573$
   $m_1\{\theta\} = 1 - m_1\{P,B,U\}$
   $= 1 - 0.573$
   $= 0.427$
2. Gejala 2: Sputum: jernih/putih (G11)
   Gejala ini merupakan gejala dari penyakit berikut dengan densitas:
   Bronkitis Akut = 0.352
   Untuk m_1 nilai densitas yang dipilih adalah yang tertinggi, maka
   $m_2\{B\} = 0.352$
   $m_2\{\theta\} = 1 - m_2\{B\}$
   $= 1 - 0.352$
   $= 0.648$

Selanjutnya menghitung nilai densitas baru untuk kombinasi dengan fungsi densitas m_3 yang dapat dilihat pada Tabel 4.14.

Tabel 4.14 Aturan Kombinasi untuk m_3

m1		m2			
		{B}	0.352	{θ}	0.648
{P,B,U}	0.573	{B}	0.202	{P,B,U}	0.371
{θ}	0.427	{B}	0.150	{θ}	0.277

Sehingga dapat dihitung

$$m_3\{B\} = \frac{0.202 + 0.0150}{1 - 0} = 0.352$$

$$m_3\{P,B,U\} = \frac{0.371}{1 - 0} = 0.371$$

$$m_3\{\theta\} = \frac{0.277}{1 - 0} = 0.277$$

3. Gejala 3: More than 10 days (G15)

Gejala ini merupakan gejala dari penyakit berikut dengan densitas:

Pneumonia = 0.5325

Untuk m_1 nilai densitas yang dipilih adalah yang tertinggi, maka

$m_4\{B\}$ = 0.5325

$m_4\{\theta\}$ = 1 - $m_4\{B\}$

= 1 - 0.5325

= 0.4675

Selanjutnya menghitung nilai densitas baru untuk kombinasi dengan fungsi densitas m_3 yang dapat dilihat pada Tabel 4.15.

Tabel 4.15 Aturan Kombinasi untuk m_5

m3		m4			
		{P}	0.5325	{θ}	0.4675
{P,B,U}	0.371	{P}	0.198	{P,B,U}	0.174
{B}	0.352	Ø	0.187	{B}	0.165
{θ}	0.277	{P}	0.147341	{θ}	0.129355

$$m_5\{P\} = \frac{0.198 + 0.147341}{1 - 0.187} = 0.37905$$

$$m_5\{P,B,U\} = \frac{0.174}{1 - 0.187} = 0.21363$$

$$m_5\{B\} = \frac{0.165}{1 - 0.187} = 0.20252$$

$$m_5\{\theta\} = \frac{0.129355}{1 - 0.187} = 0.15919$$

4. Gejala 4: Chest pain (G3)

Gejala ini merupakan gejala dari penyakit berikut dengan densitas:

Pneumonia = 0.468

Untuk m_6 nilai densitas yang dipilih adalah yang tertinggi, maka

$m_6\{P\}$ = 0.468

$m_6\{\theta\}$ = 1 - $m_6\{P\}$

= 1 - 0.468

= 0.532

Selanjutnya menghitung nilai densitas baru untuk kombinasi dengan fungsi densitas m_3 yang dapat dilihat pada Tabel 4.16.

Tabel 4.16 Aturan kombinasi untuk m_7

m5	m6			
	{P}	0.468	{θ}	0.532

{P}	0.379	{P}	0.177	{P}	0.202
{P,B,U}	0.214	{P}	0.100	{P,B,U}	0.114
{B}	0.203	Ø	0.095	{B}	0.108
{θ}	0.159	{P}	0.074503	{θ}	0.084692

$$m_7\{P\} = \frac{0.177 + 0.100 + 0.202 + 0.074503}{1 - 0.095} = 0.57464$$

$$m_7\{P,B,U\} = \frac{0.114}{1 - 0.095} = 0.12555$$

$$m_7\{B\} = \frac{0.108}{1 - 0.095} = 0.11902$$

$$m_7\{\theta\} = \frac{0.084692}{1 - 0.095} = 0.09356$$

5. Gejala 5:Dyspnea (sesak nafas) (G6)

Gejala ini merupakan gejala dari penyakit berikut dengan densitas:

Pneumonia = 0.472

Bronkitis akut = 0.582

Untuk m_8 nilai densitas yang dipilih adalah yang tertinggi, maka

$m_8\{P,B\} = 0.582$

$m_8\{\theta\} = 1 - m_8\{P,B\}$

$= 1 - 0.582$

$= 0.418$

Selanjutnya menghitung nilai densitas baru untuk kombinasi dengan fungsi densitas m_3 yang dapat dilihat pada Tabel 4.17.

Tabel 4.17 Aturan kombinasi untuk m_9

m7		m8			
		{P,B}	0.582	{θ}	0.418
{P}	0.575	{P}	0.334	{P}	0.240
{P,B,U}	0.126	{P,B}	0.073	{P,B,U}	0.052
{B}	0.119	{B}	0.069	{B}	0.050
{θ}	0.094	{P,B}	0.054451	{θ}	0.039108

$$m_9\{P\} = \frac{0.334 + 0.240}{1 - 0} = 0.57464$$

$$m_9\{P,B,U\} = \frac{0.052}{1 - 0} = 0.05248$$

$$m_9\{B\} = \frac{0.069 + 0.050}{1 - 0} = 0.11902$$

$$m_9\{P,B\} = \frac{0.073 + 0.054451}{1 - 0} = 0.12752$$

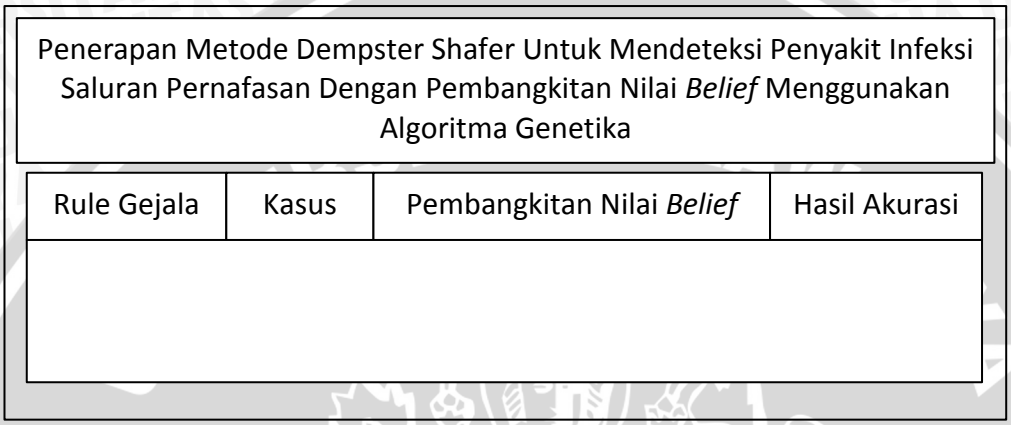
$$m_9\{\theta\} = \frac{0.039108}{1 - 0} = 0.039108$$

Dari hasil perhitungan dengan metode Dempster-Shafer menggunakan nilai *belief* yang telah dioptimasi dengan algoritma genetika, nilai densitas paling

tinggi terdapat pada penyakit pneumonia. Maka dari kasus ini pengguna diindikasikan terserang penyakit pneumonia.

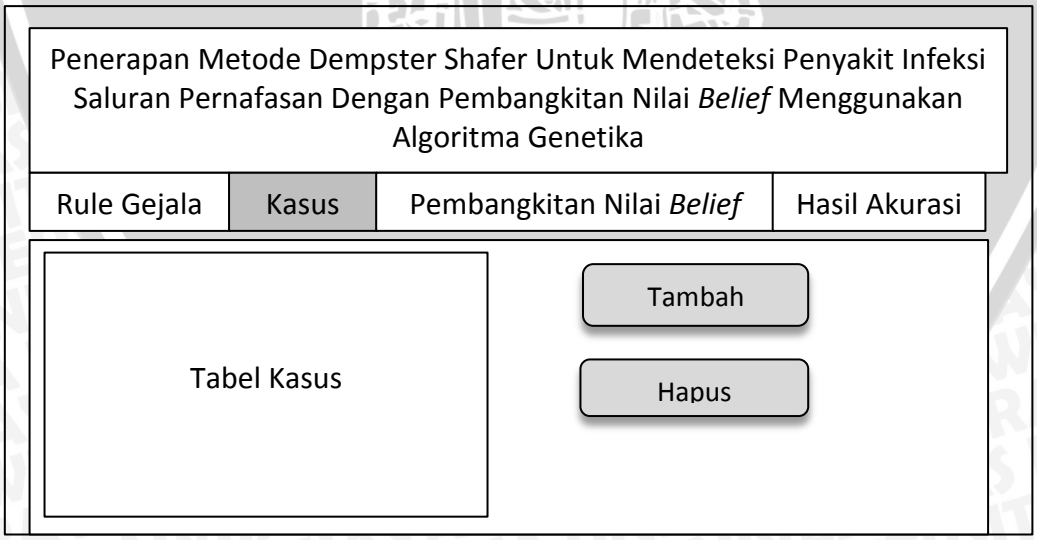
4.3 Perancangan Antarmuka

Perancangan antarmuka dari sistem ini terdiri dari halaman pembangkitan nilai *belief* dan halaman deteksi penyakit. Pada halaman pembangkitan nilai *belief* akan ditampilkan beberapa tab menu, yaitu menu rule gejala, kasus, pembangkitan nilai *belief*,dan hasil akurasi. Perancangan antarmuka halaman pembangkitan nilai *belief* dapat dilihat pada Gambar 4.10.



Gambar 4.10 Antarmuka Halaman Pembangkitan Nilai *Belief*

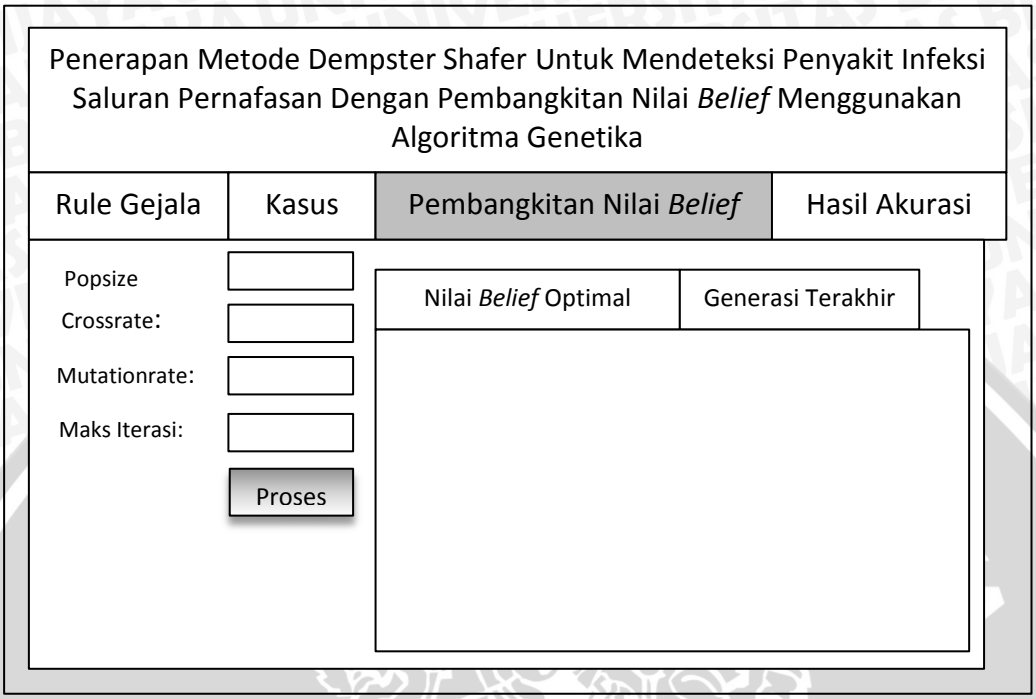
Pada tab menu rule gejala akan ditampilkan tabel aturan (*rule*) dari gejala, yaitu tabel dari akusisi pengetahuan. Pada tab kasus akan ditampilkan data kasus yang digunakan sebagai data latih. Pengguna pun dapat menambah dan menghapus kasus sehingga sistem pun bisa belajar dan meningkatkan akurasi. Perancangan antarmuka halaman kasus dapat dilihat pada Gambar 4.11.



Gambar 4.11 Antarmuka Halaman Kasus

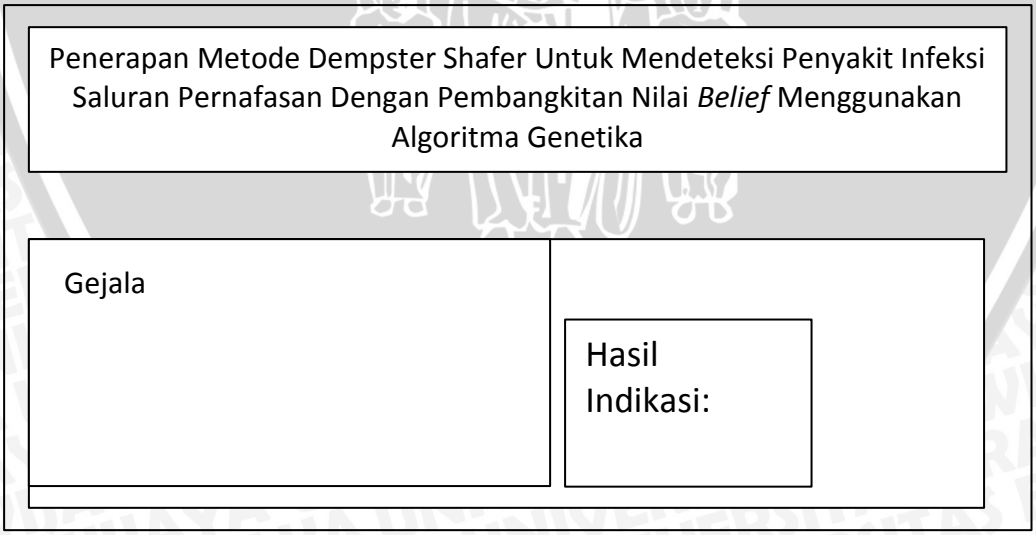
Pada menu pembangkitan nilai *belief*, proses pembangkitan nilai *belief* dilakukan. Pengguna dapat menginputkan jumlah populasi, cr, mr, dan generasi

yang digunakan. Sistem akan menampilkan nilai *belief*, generasi terakhir, serta *fitness*nya. Perancangan antarmuka halaman pembangkitan nilai *belief* dapat dilihat pada Gambar 4.12.



Gambar 4.12 Antarmuka Halaman Pembangkitan Nilai *Belief*

Pada menu diagnosa pengguna dapat menginputkan gejala, kemudian sistem akan menampilkan hasil indikasi penyakit yang dialami pengguna. Perancangan antarmuka halaman diagnosa dapat dilihat pada Gambar 4.13.



Gambar 4.13 Antarmuka Halaman Diagnosa

BAB 5 IMPLEMENTASI APLIKASI

Bab ini membahas mengenai implementasi perangkat lunak berdasarkan hasil yang telah diperoleh dari analisa kebutuhan dan proses perancangan aplikasi yang telah dibuat. Pembahasan terdiri dari penjelasan tentang implementasi spesifikasi sistem dan implementasi program Penerapan Dempster-Shafer Untuk Deteksi Penyakit Pernafasan Dengan Pembangkitan Nilai *Belief* Menggunakan Algoritma Genetika.

5.1 Spesifikasi Aplikasi

Hasil analisis kebutuhan dan perancangan perangkat lunak yang telah diuraikan pada Bab 3 menjadi acuan untuk melakuan implementasi menjadi sistem yang dapat berfungsi sesuai dengan kebutuhan, spesifikasi sistem diimplementasikan pada spesifikasi perangkat keras dan perangkat lunak.

5.1.1 Spesifikasi Perangkat Keras

Pengembangan program Penerapan Dempster-Shafer Untuk Deteksi Penyakit Pernafasan Dengan Pembangkitan Nilai *Belief* Menggunakan Algoritma Genetika menggunakan sebuah komputer dengan minimal spesifikasi perangkat keras yang dijelaskan pada tabel 5.1.

Tabel 5.1 Spesifikasi perangkat keras

Komponen	Spesifikasi
Prosesor	Intel Core i5
Memori (RAM)	8 GB
Kartu Grafis	Nvidia 1 GB
Harddisk	500GB HDD
Layar Monitor	14” LCD

5.1.2 Spesifikasi Perangkat Lunak

Pengembangan program Penerapan Dempster-Shafer Untuk Deteksi Penyakit Pernafasan Dengan Pembangkitan Nilai *Belief* Menggunakan Algoritma Genetika menggunakan laptop dengan minimal spesifikasi perangkat lunak yang dijelaskan pada tabel 5.2.

Tabel 5.2 Spesifikasi perangkat lunak

Sistem Operasi	Microsoft Windows 7 (32 bit)
Bahasa Pemrograman	C#
Tools pemograman	Microsoft Visual Studio 2013

5.2 Implementasi Program

Berdasarkan perancangan proses yang terdapat pada Bab 4, maka pada sub bab ini akan menjelaskan implementasi dari proses-proses tersebut. Pada implementasi program menggunakan bahasa pemrograman C#.

5.2.1 Inisialisasi Populasi Awal

Pada proses inisialisasi populasi awal, kromosom tiap individu dibangkitkan secara acak sesuai aturan pada data gejala. Aturan yang digunakan dapat dilihat pada tabel 3.1 dimana terdapat gejala yang mempengaruhi suatu penyakit dan terdapat gejala yang tidak mempengaruhi suatu penyakit. Jika suatu gejala mempengaruhi penyakit maka dilakukan random dengan range nilai minimum dan nilai maksimum. Namun, jika suatu gejala tidak mempengaruhi suatu penyakit maka gejala tersebut bernilai kosong. Proses ini ditunjukkan pada baris kode ke-18 sampai ke-25 pada *Source code* 5.1.

```
class Density
1   private Random _r = new Random();
2   private List<Gejala> density = new List<Gejala>();
3   private List<Gejala> rule;
4   private double minimum = 0;
5   private double maximum = 1;
6   }
7      private void GenerateRandom()
8      {
9         density.Clear();
10        for (int i = 0; i < rule.Count; i++)
11        {
12           Gejala ge = new Gejala();
13           ge.SetKodeGejala(rule[i].GetKodeGejala());
14           ge.SetNamaGejala(rule[i].GetNamaGejala());
15           double[] p = new double[3];
16           for (int j = 0; j < 3; j++)
17           {
18              if (rule[i].GetPenyakitById(j) == 1)
19              {
                    p[j] =
20  Math.Round(GetRandomNumber(this.minimum, this.maximum), 5);
21              }
22              else
23              {
24                 p[j] = 0;
25              }
26           }
27           ge.SetPenyakit(p);
28           density.Add(ge);
```

```
29              }
30          }
```

***Source Code* 5.1 Inisialisasi Populasi Awal**

5.2.2 Implementasi *Crossover*

Metode *crossover* yang digunakan pada penelitian ini adalah *Extended Intermediate Crossover*. Proses *extended intermediate crossover* ini diawali dengan menentukan jumlah anak (*offspring*) yang dihasilkan. Pada kode baris ke-1 sampai baris ke-2 menunjukkan proses perhitungan jumlah *offspring* yang dihasilkan. Pada kode baris ke-3 sampai baris ke-19 merupakan proses pengambilan data gejala dari database yang dideklarasikan pada variable yang baru. Selanjutnya membangkitkan bilangan random α dengan range -0.25 - 1.25 yang ditunjukkan baris kode ke-21 sampai baris ke-23. Kemudian memilih individu yang akan dicrossover secara acak, ini ditunjukkan pada baris kode ke-27. Berikut merupakan source code untuk proses crossover yang ditunjukkan pada Source Code 5.2.

```
public CrossOver(List<Individu> individu, double crossrate, int
popSize)
 1  double offspring = crossrate * (double) popSize;
 2  offspring = Math.Floor(offspring);
 3  SqliteDatabase db = new SqliteDatabase();
 4  DataTable dt = db.query("SELECT * FROM gejala");
 5
 6  rule = new List<Gejala>();
 7
 8  foreach (DataRow row in dt.Rows)
 9  {
10     Gejala g = new Gejala();
11     double[] p = new double[3];
12     g.SetKodeGejala(Convert.ToString(row["kode_gejala"]));
13     g.SetNamaGejala(Convert.ToString(row["nama_gejala"]));
14     p[0] = Convert.ToDouble(row["pneumonia"]);
15     p[1] = Convert.ToDouble(row["bronkitis"]);
16     p[2] = Convert.ToDouble(row["urti"]);
17     g.SetPenyakit(p);
18     rule.Add(g);
19  }
20
21  Density de = new Density(rule);
22  de.setMinMax(-0.25, 0.25);
23  Individu alpha = new Individu(de.GetDensity());
24
25  crossover = new List<Individu>();
26
27  int[] randomIndividu = generateNumber(individu.Count);
28  int offset = 0;
29  while (crossover.Count < offspring) //?
```

```
{
    List<Gejala> c = new List<Gejala>();
    c = Cross(individu[offset], individu[offset + 1],
alpha);
    Individu v = new Individu(c);
    crossover.Add(v);

    if (crossover.Count < offspring)
    {
        c = Cross(individu[offset + 1], individu[offset],
alpha);
        Individu v2 = new Individu(c);
        crossover.Add(v2);
        offset = offset + 2;
     }
}
```

Source Code* 5.2 Proses *Extended Intermediate Crossover

Proses perhitungan *offspring* menggunakan persamaan 2.4 dan persamaan 2.5 ditunjukkan pada *Source Code* 5.3.

```
private List<Gejala> Cross(Individu a, Individu b, Individu
alpha)
{
List<Gejala> hasilKross = new List<Gejala>();
for (int i = 0; i < rule.Count; i++)
{
    Gejala g = new Gejala();
    double[] aDen = a.find(rule[i].GetKodeGejala());
    double[] bDen = b.find(rule[i].GetKodeGejala());
    double[] alphaden = alpha.find(rule[i].GetKodeGejala());
    double[] hasil = new double[3];
    for (int j = 0; j < 3; j++)
    {
        hasil[j] = aDen[j] + alphaden[j] * (bDen[j] -
aDen[j]);
    }
    g.SetKodeGejala(rule[i].GetKodeGejala());
    g.SetPenyakit(hasil);
    hasilKross.Add(g);
    }
    return hasilKross;
}
```

Source Code* 5.3 Rumus *Extended Intermediate Crossover

5.2.3 Implementasi Mutasi

Metode mutasi yang digunakan pada penelitian ini adalah *Random Mutation*. Dalam membangkitkan *offspring* dari proses mutasi hal-hal yang dilakukan adalah memilih individu yang menjadi induk (*parent*) pada proses mutasi secara acak, membangkitkan bilangan random r, dan membangkitkan

offspring dengan menggunakan Persamaan 2.6. Membangkitkan bilangan random r ditunjukkan pada baris kode ke-14. Untuk pemilihan induk dan perhitungan *offspring* ditunjukkan pada baris kode ke-15 sampai dengan baris kode ke-31. Berikut merupakan source code untuk proses mutasi yang ditunjukkan pada Source Code 5.4.

```
class Mutasi
1  {
2  Random _r = new Random();
3  Individu hasilMutasi;
4  List<Gejala> rule;
   public Mutasi(List<Individu> individu, double mutasiRate, int
5  popSize, double maxmin, List<Gejala> rule)
6  {
7       this.rule = rule;
8       double offspring = mutasiRate * (double)popSize;
9       offspring = Math.Floor(offspring);
10      List<Gejala> g = new List<Gejala>();
11      int randomIndividu = GetRandomIndex(individu.Count);
12      int randomGejala = GetRandomIndex(rule.Count);
13      int randomPenyakit = GetRandomIndex(3);
14      double r = GetRandomNumber(-0.1, 0.1);
        double[] gen =
   individu[randomIndividu].find(rule[randomGejala].GetKodeGejal
15 a());
16     gen[randomPenyakit] = gen[randomPenyakit] + (r * maxmin);
17     for (int i = 0; i < rule.Count; i++)
18     {
19         Gejala gg = new Gejala();
20         gg.SetKodeGejala(rule[i].GetKodeGejala());
21         if (i == randomGejala)
22         {
23            gg.SetPenyakit(gen);
24         }
25         Else
26         {

   gg.SetPenyakit(individu[randomIndividu].find(rule[i].GetKodeG
27 ejala()));
28         }
29         g.Add(gg);
30     }
31     hasilMutasi = new Individu(g);
32 }
33 private int GetRandomIndex(int jumlahIndividu)
34 {
35      int index;
36      index = _r.Next(0, jumlahIndividu - 1);
37      return index;
38 }
```

```
    private double GetRandomNumber(double minimum, double
39  maximum)
40  {
41      return _r.NextDouble() * (maximum - minimum) + minimum;
42  }
43  public Individu GetMutasi()
44  {
45      return this.hasilMutasi;
46  }
47  }}
```

Source Code* 5.4 Proses *Random Mutation

5.2.4 Perhitungan Nilai *Fitness* dan Metode *Dempster Shafer*

Proses perhitungan *fitness* pada penelitian ini dilakukan dengan menggunakan perhitungan akurasi dalam mendiagnosa penyakit dengan menggunakan individu yang telah dibangkitkan sebagai nilai *belief*. Metode yang digunakan untuk mendiagnosa penyakit saluran pernafasan adalah Metode Dempster-Shafer dengan nilai *fitness* yang akan digunakan dalam menentukan baik tidaknya individu.

Perhitungan *fitness* dilakukan dengan mencocokkan diagnosa sistem dengan diagnosa pakar, sehingga perlu dilakukan perhitungan Dempster Shafer untuk mendapatkan nilai *fitness* tersebut. Fungsi perhitungan *fitness* ditunjukkan pada *Source Code* 5.5. Pada baris ke-2 terdapat perhitungan nilai *fitness* yang didapatkan dari hasil bagi jumlah data kasus yang cocok dengan diagnosa pakar dan jumlah data kasus.

```
public double Get Fitness()
1           {
2               fitness = (double) counter / (double) data.Count;
3               return Math.Round(fitness, 5);
4           }
```

Source Code* 5.5 Perhitungan *Fitness

Metode Dempster Shafer digunakan untuk menentukan diagnosa penyakit yang nantinya akan digunakan untuk perhitungan *fitness*. Pada baris ke-16 sampai baris ke-33 merupakan proses pengambilan nilai *belief* sementara untuk diketahui nilai *belief* yang terbesar dari antara gejala yang dipilih. Perhitungan metode Dempster Shafer ditunjukkan pada *Source Code* 5.6.

```
public DempsterShafer(Individu individu, List<Kasus> data)
 1  {
 2    this.individu = individu;
 3    this.data = data;
 4    for (int i = 0; i < data.Count; i++)
 5    {
 6      Console.Write("Kasus " + (i + 1) + " | ");
 7      string[] gejala = data[i].GetGejala();
 8      List<List<Mass>> Masses = new List<List<Mass>>();
```

```
 9        int mCounter = 0;
10        for (int j = 0; j < gejala.Length; j++)
11        {
12            double[] nilaiGejala = individu.find(gejala[j]);
13            double max = GetMaximumDensity(nilaiGejala);
14            if (max == 1)
15            {
16                Mass m = new Mass();
17                m.SetPenyakit(GetKodePenyakit(nilaiGejala));
18                m.SetDensity(max);
19                string[] indikasi = data[i].GetDiagnosa();
20                int ada = 0;
21                foreach (string idk in indikasi)
22                {
23                    if (m.GetPenyakit().Contains(idk))
24                    {
25                        ada++;
26                    }
27                }
28                if (ada != 0)
29                {
30                  counter++;
31                }
32                break;
33            }
34          List<string> sTheta = new List<string>() { "theta" };
35          if (j == 0) { // gejala pertama
36            List<Mass> M = new List<Mass>();
37            Mass m = new Mass();
38            m.SetPenyakit(GetKodePenyakit(nilaiGejala));
39            m.SetDensity(max);
40            M.Add(m);
41            Mass mTheta = new Mass();
42            mTheta.SetPenyakit(sTheta);
43            mTheta.SetDensity(GetTheta(max));
44            M.Add(mTheta);
45            Masses.Add(M); // m1
46          } else { // gejala ke-2 dst
47            List<Mass> M = new List<Mass>();
48            Mass m = new Mass();
49            m.SetPenyakit(GetKodePenyakit(nilaiGejala));
50            m.SetDensity(max);
51            M.Add(m);
52            Mass mTheta = new Mass();
```

```
53              mTheta.SetPenyakit(sTheta);
54              mTheta.SetDensity(GetTheta(max));
55
56              M.Add(mTheta);
57
58              Masses.Add(M);
59              mCounter++;
60
61              List<Mass> matriksKombinasi = new List<Mass>();
62
    matriksKombinasi = CombineMass(Masses[mCounter - 1],
63  Masses[mCounter]);
64           List<Mass> mBaru = GetMassBaru(matriksKombinasi);
65
66           Masses.Add(mBaru);
67           mCounter++;
68          }
69
70         }
71
72         mFinal = Masses[mCounter];
73         int indeksMaks = 0;
74         double terbesar = 0;
75
76         for (int x = 0; x < mFinal.Count; x++)
77         {
78            if (terbesar < mFinal[x].GetDensity())
79            {
80                 terbesar = mFinal[x].GetDensity();
81                 indeksMaks = x;
82            }
83         }
84
85         List<string> d = mFinal[indeksMaks].GetPenyakit();
86
87         DebugString(d);
88         Console.Write(" | Nilai Belief : " + terbesar);
89         Console.WriteLine();
90
91         string[] indi = data[i].GetDiagnosa();
92         int a = 0;
93         if (indi != null)
94         {
95            foreach (string idk in indi)
96            {
```

```
97              if (d.Contains(idk))
98          {
99          a++;
100       }
101     }
102     }
103     if (a != 0)
104     {
105         counter++;
106     }
107 }
108 }
109
110 private double GetTheta(double maximumDensity)
111 {
112     return 1 - maximumDensity;
113 }
114
115 private List<Mass> CombineMass(List<Mass> a, List<Mass> b)
116 {
117     List<Mass> hasilKombinasi = new List<Mass>();
118     for (int i = 0; i < a.Count; i++)
119     {
120         for (int j = 0; j < b.Count; j++)
121         {
122             Mass hasilKali = new Mass();
 hasilKali.SetPenyakit(GetIntersection(a[i].GetPenyakit(),
123 b[j].GetPenyakit()));
    hasilKali.SetDensity(Math.Round((a[i].GetDensity() *
124 b[j].GetDensity()), 3));
125             hasilKombinasi.Add(hasilKali);
126         }
127     }
128
129 return hasilKombinasi;
130 }
131
132 private double GetMaximumDensity(double[] array)
133 {
134     return array.Max();
135 }
136
137 private List<string> GetKodePenyakit(double[] dens)
138 {
139     List<string> kodePenyakit = new List<string>();
```

```
140         if (dens[0] != 0) kodePenyakit.Add("P");
141             if (dens[1] != 0) kodePenyakit.Add("B");
142                 if (dens[2] != 0) kodePenyakit.Add("U");
143                     return kodePenyakit;
144     }
145
146     private List<string> GetIntersection(List<string> a,
    List<string> b)
147     {
148         List<string> intersection = new List<string>();
149         if (!a.Contains("theta") && !b.Contains("theta"))
150         {
151             intersection = a.Intersect(b,
    StringComparer.OrdinalIgnoreCase).ToList();
152             if (intersection.Count == 0)
153             {
154                 intersection.Add("kosong");
155             }
156         }
157         else if(!a.Contains("theta") && b.Contains("theta"))
158         {
159             intersection = a;
160         }
161         else if (a.Contains("theta") && !b.Contains("theta"))
162         {
163             intersection = b;
164         }
165         else if (a.Contains("theta") && b.Contains("theta"))
166         {
167             intersection = a;
168         }
169
170         return intersection;
171     }
172
173     private List<Mass> GetMassBaru(List<Mass> mK)
174     {
175         List<Mass> mBaru = new List<Mass>();
176         List<int> indeksOfMass = new List<int>();
177         double pembagi = 0;
178         double pembilang = 0;
179
180         for (int i = 0; i < mK.Count; i++)
181         {
182             if(indeksOfMass.Contains(i)) continue;
```

```
183         indeksOfMass.Add(i);
184 
185           if (mK[i].GetPenyakit().Contains("kosong"))
186           {
187               pembagi = mK[i].GetDensity();
188           }
189           else
190           {
191               pembilang = mK[i].GetDensity();
192           }
193 
194            for (int j = i + 1; j < mK.Count; j++)
195            {
196                var a = mK[i].GetPenyakit();
197                var b = mK[j].GetPenyakit();
198 
199           if(b.Contains("kosong")){
200                pembagi += mK[j].GetDensity();
201                indeksOfMass.Add(j);
202           } else {
203                if (IsEqual(a, b))
204                {
205                   pembilang += mK[j].GetDensity();
206                   indeksOfMass.Add(j);
207                }
208            }
209            }
210 
211         if (!mK[i].GetPenyakit().Contains("kosong"))
212         {
213             Mass mm = new Mass();
214             mm.SetPenyakit(mK[i].GetPenyakit());
215             double nDensity = pembilang / (1 - pembagi);
216             mm.SetDensity(Math.Round(nDensity, 5));
217 
218             mBaru.Add(mm);
219          }
220     }
221     return mBaru;
222 }
```

Source Code 5.6 Metode Dempster Shafer

Telah ditentukan beberapa kasus yang telah didiagnosa oleh pakar dan disimpan dalam datatabase. Perhitungan *fitness* terdapat dalam fungsi dempster shafer sehingga ketika selesai dilakukan perhitungan kasus dengan dempster

shafer, kemudian dicocokkan dengan kasus pada database. Kemudian terdapat fungsi Get*Fitness*() untuk memanggil nilai *fitness* itu sendiri.

5.2.5 Implementasi Seleksi

Pada proses seleksi metode yang digunakan adalah metode seleksi *Elitism*. Baris kode ke-2 adalah proses menggabungkan isi dari variabel P yang berisi individu *parent* dan variabel C yang berisi individu *offspring.* Proses seleksi dilakukan dengan cara mengurutkan nilai *fitness* mulai yang terbesar sampai terkecil. Proses ini ditunjukkan pada baris kode ke-22 sampai baris kode ke-30. Kemudian akan dipilih individu terbaik sebanyak jumlah *popsize.* Hasil dari pengurutan nilai *fitness* tersebut menjadi individu baru yang akan digunakan untuk proses pada generasi berikutnya. Berikut ini merupakan *souce code* untuk seleksi *Elitism* yang ditunjukkan pada *Source Code* 5.7.

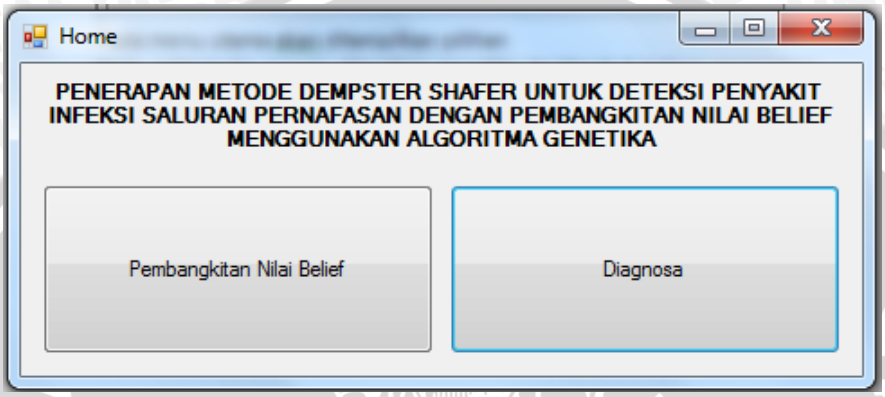
```
public Selection(List<Individu> P, List<Individu> C, int
popSize)
1  {
2    P.AddRange(C);
3    sort(P);
4    iTerbaik = new List<Individu>();
5    for (int i = 0; i < popSize; i++)
6    {
7        int idIndividu = index[i]+1;
8        if ( idIndividu > popSize)
9        {
10          idIndividu -= popSize;
11          Console.Write("C" + idIndividu + " | ");
12       }
13       else
14       {
15           Console.Write("P" + idIndividu + " | ");
16        }
17        Console.WriteLine(P[index[i]].GetFitness());
18        iTerbaik.Add(P[index[i]]);
19   }
20 }
21
22 private void sort(List<Individu> listIndividu)
23 {
24    var std = listIndividu
                .Select((x, i) => new KeyValuePair<Individu,
25 int>(x, i))
26              .OrderByDescending(x => x.Key.GetFitness())
27              .ToList();
28     sorted = std.Select(x => x.Key).ToList();
29     index = std.Select(x => x.Value).ToList();
```

```
30  }
```

Source Code 5.7 Proses *Elitism Selection*

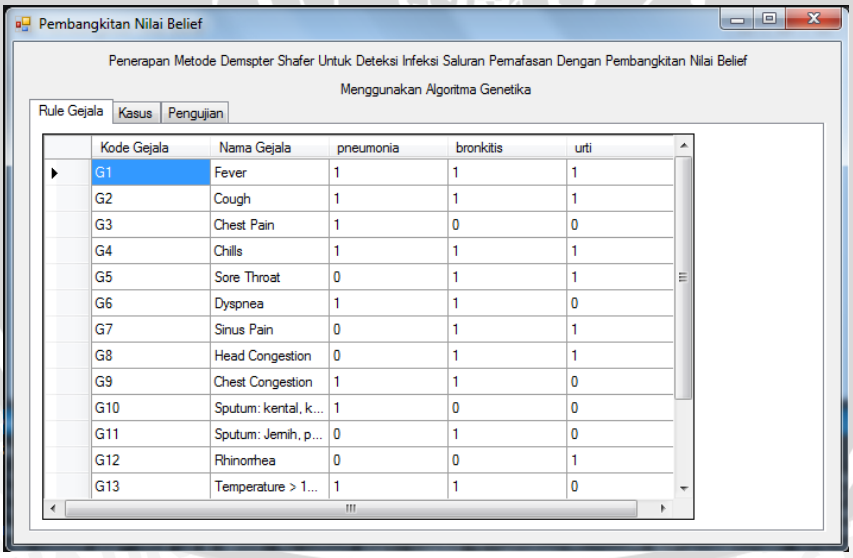
5.3 Implementasi Antarmuka

Antarmuka sistem digunakan oleh pengguna untuk berinteraksi dengan sistem. Antarmuka dibagi menjadi halaman proses algoritma genetika dan halaman deteksi penyakit yang akan ditampilkan pada menu utama. Menu utama dapat dilihat pada Gambar 5.1.



Gambar 5.1 Menu utama

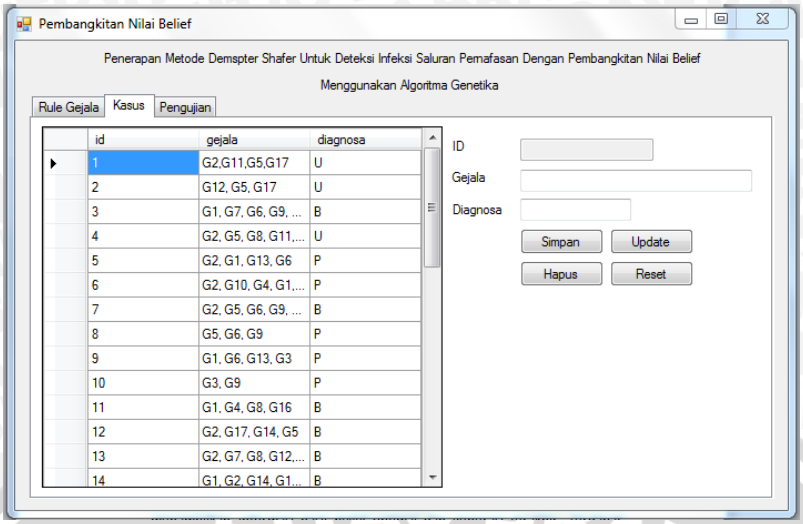
Pada menu pembangkitan nilai *belief* terdapat beberapa tab yang menampilkan rule gejala, kasus, dan proses algoritma genetika. Rule gejala menunjukkan aturan gejala dan penyakit yang saling mempengaruhi. Tampilan menu rule gejala dapat dilihat pada Gambar 5.2.



Kode Gejala	Nama Gejala	pneumonia	bronkitis	urti
G1	Fever	1	1	1
G2	Cough	1	1	1
G3	Chest Pain	1	0	0
G4	Chills	1	1	1
G5	Sore Throat	0	1	1
G6	Dyspnea	1	1	0
G7	Sinus Pain	0	1	1
G8	Head Congestion	0	1	1
G9	Chest Congestion	1	1	0
G10	Sputum: kental, k...	1	0	0
G11	Sputum: Jernih, p...	0	1	0
G12	Rhinorrhea	0	0	1
G13	Temperature > 1...	1	1	0

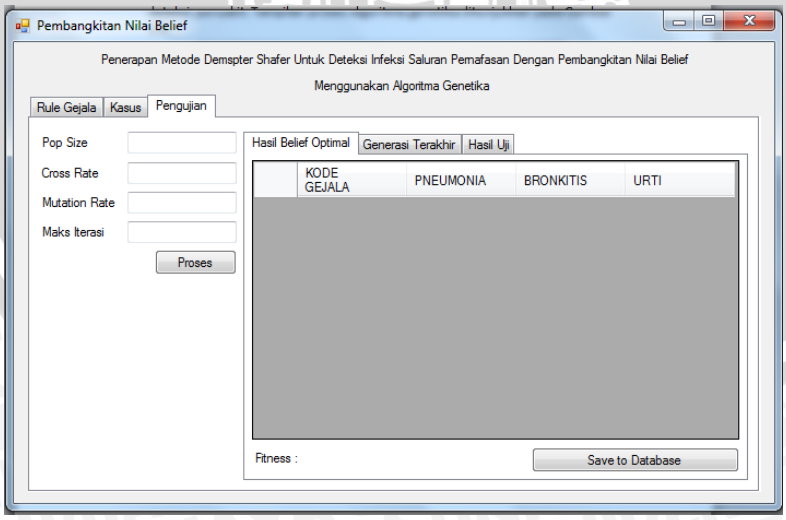
Gambar 5.2 Tampilan Menu Rule Gejala

Pada menu kasus menampilkan sejumlah kasus yang terdiri dari gejala dan hasil diagnosa dari pakar. Terdapat fitur untuk menambah, menghapus, dan meng-*update* kasus. Tampilan menu kasus ditunjukkan pada Gambar 5.3.



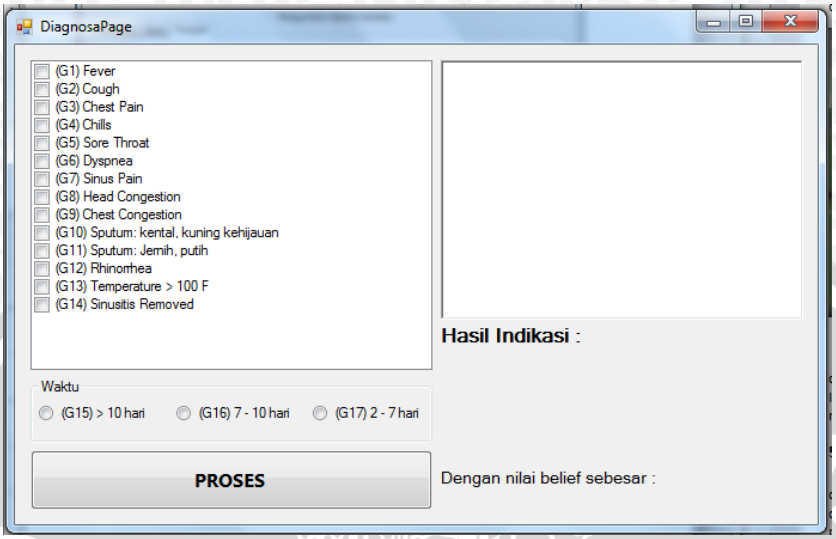
Gambar 5.3 Tampilan Kasus

Pada antarmuka proses algoritma genetika terdapat 4 kolom parameter-parameter algoritma genetika yang harus diisi oleh pengguna. Parameter-parameter tersebut antara lain probabilitas *crossover*, probabilitas mutasi, jumlah generasi, dan ukuran populasi. Setelah tombol proses ditekan dan sistem melakukan proses optimasi, sistem akan menampilkan informasi hasil *belief* optimal, generasi terakhir, dan hasil uji. Pada tab hasil belief optimal akan ditunjukkan hasil nilai belief dari gejala setiap penyakit dan terdapat tombol Save to Database yang dapat digunakan ketika pengguna ingin menyimpan hasil nilai *belief* pada database sehingga bisa digunakan untuk deteksi penyakit. Pada tab generasi terakhir akan ditampilkan populasi terakhir sebanyak *popsize* dan besar *fitness*-nya. Pada tab hasil uji akan menampilkan hasil akurasi dari pengujian terhadap 17 kasus uji. Tampilan menu proses algoritma genetika ditunjukkan pada Gambar 5.4.



Gambar 5.4 Tampilan Proses Algoritma Genetika

Pada menu deteksi penyakit terdapat gejala yang dapat dipilih pengguna untuk diproses kemudian akan ditampilkan hasil penyakitnya. Tampilan menu deteksi penyakit ditunjukkan pada Gambar 5.5.



Gambar 5.5 Tampilan Deteksi Penyakit

BAB 6 PENGUJIAN DAN ANALISIS

Bab ini menjelaskan mengenai hasil pengujian yang telah dilakukan dalam penerapan metode Dempster-Shafer untuk deteksi penyakit infeksi saluran pernafasan dengan pembangkitan nilai *belief* menggunakan algoritma genetika. Pengujian dilakukan untuk menguji parameter jumlah populasi, *crossover rate* (*Cr*), *mutation rate* (*Mr*), jumlah generasi pada algoritma genetika agar mendapatkan nilai *fitness* terbaik, serta mengetahui pengaruh algoritma genetika terhadap optimasi nilai *belief* pada metode Dempster-Shafer.

6.1 Pengujian Parameter Algoritma Genetika

Pengujian parameter algoritma genetika meliputi pengujian ukuran populasi, pengujian jumlah generasi, serta pengujian nilai *crossover rate* (*cr*) dan *mutation rate* (*mr).* Pengujian yang dilakukan adalah membandingkan perubahan nilai parameter terhadap nilai *fitness* yang dihasilkan

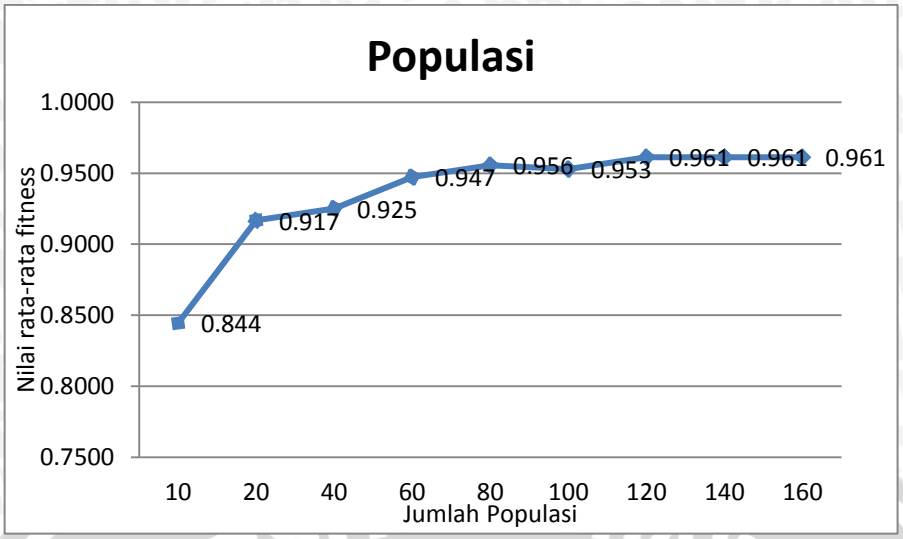
6.1.1 Pengujian Jumlah Populasi (*Popsize*) Terhadap Nilai *Fitness*

Pengujian ini dilakukan untuk mengetahui seberapa banyak jumlah populasi dengan nilai *fitness* yang terbesar. Pada pengujian jumlah populasi (*popsize*) banyaknya populasi yang diujikan sebanyak 10, 20, 40, 60, 80, 100, 120, 140, dan 160 dengan 10 kali pengujian. Pada pengujian ini, nilai parameter generasi yaitu 20 generasi dan nilai Cr dan Mr 0,5. Hasil pengujian terhadap jumlah populasi dapat ditunjukkan pada Tabel 6.1.

Tabel 6.1 Hasil pengujian jumlah populasi

Populasi	Nilai *Fitness* Pengujian ke-i										Rata-rata *fitness*
	1	2	3	4	5	6	7	8	9	10	
10	0.861	0.861	0.861	0.861	0.778	0.861	0.833	0.861	0.833	0.833	0.844
20	0.917	0.889	0.917	0.889	0.889	0.917	0.944	0.944	0.944	0.917	0.917
40	0.889	0.861	0.944	0.944	0.972	0.972	0.917	0.917	0.917	0.917	0.925
60	0.944	0.917	0.944	0.944	0.944	0.944	0.972	0.944	0.944	0.972	0.947
80	0.944	0.944	0.972	0.944	0.917	0.972	0.972	0.944	0.972	0.972	0.956
100	0.944	0.944	0.944	0.972	0.944	0.972	0.972	0.944	0.944	0.944	0.953
120	0.972	0.972	0.944	0.944	0.972	0.972	0.972	0.944	0.944	0.972	0.961
140	0.972	0.972	0.972	0.972	0.972	0.972	0.944	0.944	0.944	0.944	0.961
160	0.944	0.944	0.944	0.972	0.972	0.944	0.972	0.972	0.972	0.972	0.961

Berdasarkan hasil yang ditunjukkan pada Tabel 6.1 dapat terlihat bahwa nilai rata-rata *fitness* terbesar didapatkan pada populasi 120 dengan nilai *fitness*-nya 0,961 dan pada ukuran populasi selanjutnya terjadi konvergensi. Grafik pengujian jumlah populasi ditunjukkan pada Gambar 6.1.



Gambar 6.1 Grafik Pengujian Jumlah Populasi Terhadap Nilai *Fitness*

Grafik pada Gambar 6.1 dapat dilihat bahwa ukuran populasi berpengaruh terhadap hasil algoritma yang terlihat pada nilai *fitness*-nya yang mengalami kenaikan dengan semakin besarnya ukuran populasi. Nilai *fitness* terendah terdapat pada populasi 10 dikarenakan ukuran populasi masih sedikit sehingga area eksplorasi dengan algoritma genetika masih sempit dan solusi yang ditemukan tidak terlalu baik. Namun ukuran populasi yang terlalu besar memungkinkan terjadi konvergensi dan membutuhkan waktu proses yang lama. Dari hasil grafik diperoleh 120 populasi memiliki jumlah populasi dengan hasil optimal.

6.1.2 Pengujian Jumlah Generasi Terhadap Nilai *Fitness*

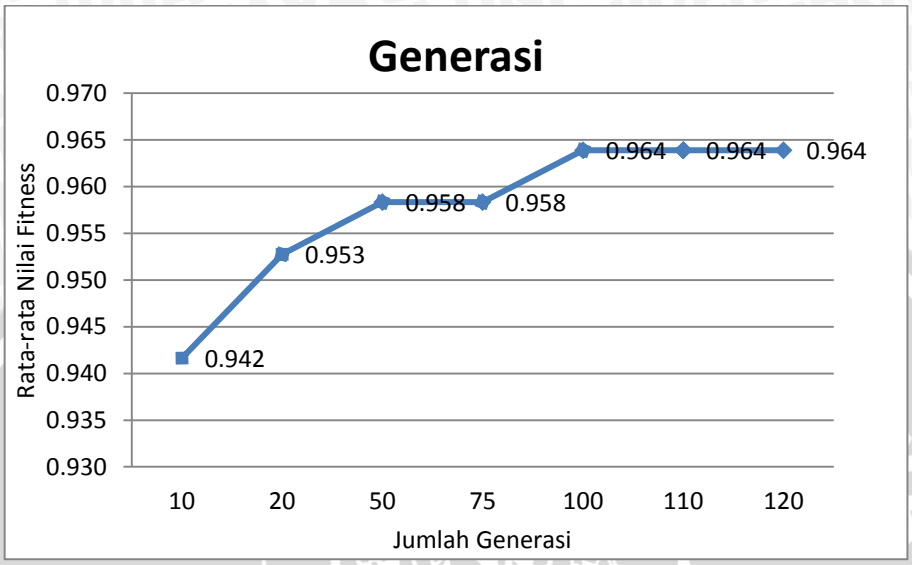
Pengujian yang kedua adalah untuk mengetahui pengaruh perubahan jumlah generasi terhadap besar nilai *fitness* maksimum yang dihasilkan. Pada pengujian ini menggunakan parameter 120 populasi yang merupakan hasil dari pengujian pertama serta menggunakan cr dan mr yang bernilai 0,5. Pada setiap pengujian satu jumlah generasi dilakukan 10 kali percobaan lalu diambil nilai rata-rata *fitness* yang kemudian akan dibandingkan dengan setiap nilai generasi yang lain. Hasil uji coba dapat dilihat pada tabel 5.2 berikut ini.

Tabel 6.2 Pengujian Jumlah Generasi

Generasi	Nilai *Fitness*										Rata-rata *fitness*
	Pengujian ke-i										
	1	2	3	4	5	6	7	8	9	10	
10	0.917	0.944	0.944	0.944	0.944	0.944	0.944	0.944	0.944	0.944	0.942
20	0.944	0.944	0.944	0.944	0.944	0.972	0.944	0.944	0.972	0.972	0.953
50	0.944	0.972	0.944	0.972	0.972	0.944	0.972	0.972	0.944	0.944	0.958
75	0.972	0.972	0.944	0.944	0.944	0.944	0.972	0.944	0.972	0.972	0.958
100	0.944	0.944	0.972	0.972	0.972	0.972	0.972	0.972	0.972	0.944	0.964
110	0.972	0.972	0.944	0.944	0.972	0.972	0.972	0.944	0.972	0.972	0.964

120	0.972	0.972	0.972	0.972	0.944	0.944	0.972	0.944	0.972	0.972	0.964

Berdasarkan Tabel 5.2 dapat dikatakan bahwa *fitness* rata-rata terbesar yang dihasilkan saat jumlah generasi sebesar 100 yaitu 0,964. Grafik pengaruh perubahan jumlah generasi terhadap nilai *fitness* ditunjukkan pada Gambar 5.2.



Gambar 6.2 Grafik Pengujian Jumlah Generasi Terhadap Nilai *Fitness*

Pada grafik Gambar 5.2 menunjukkan bahwa jumlah generasi berbanding lurus dengan nilai *fitness* yang dihasilkan, semakin banyak jumlah generasi berpengaruh terhadap meningkatnya kemampuan algoritma genetika dalam mencari solusi. Pada jumlah generasi 100 merupakan titik optimum karena tidak terjadi kenaikan rata-rata fitness yang signifikan setelah generasi 100 dan selanjutnya nilai *fitness* menunjukkan konvergensi.

6.1.3 Pengujian Probabilitas Crossover dan Probabilitas Mutasi Terhadap Nilai *Fitness*

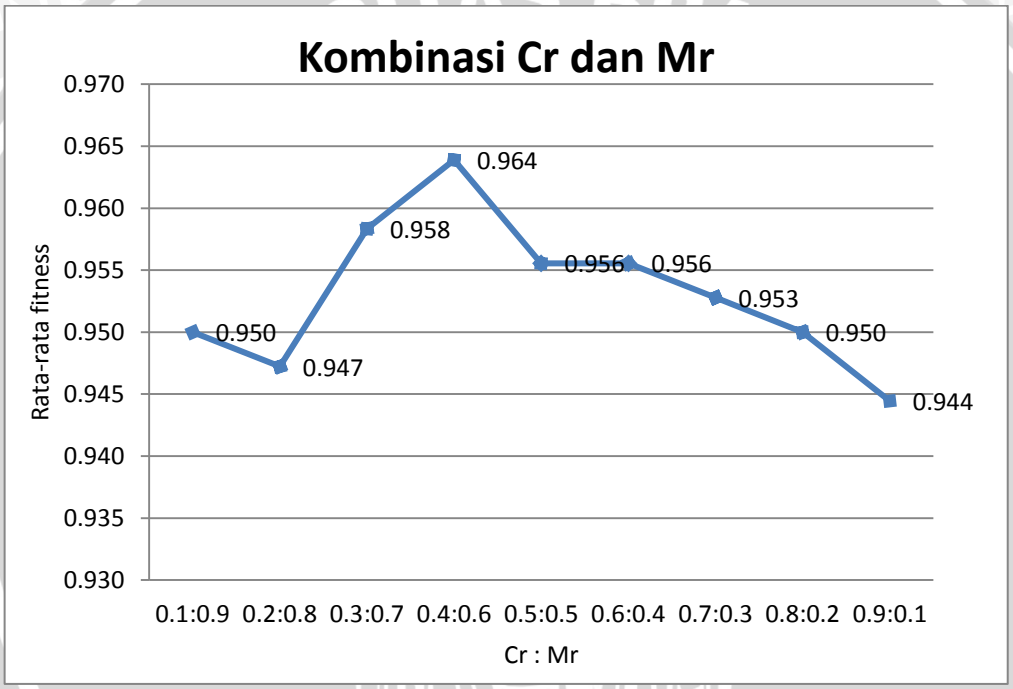
Pengujian ini dilakukan untuk mengetahui apakah perubahan nilai probabilitas *crossover* (cr) dan probabilitas mutasi (mr) berpengaruh terhadap perubahan nilai *fitnes* serta mengetahui nilai probabilitas *crossover* (cr) dan probabilitas mutasi (mr) yang tepat untuk menghasilkan solusi optimal dalam kasus ini. Dalam pengujian ini menggunakan parameter jumlah populasi 120 dan jumlah generasi 100. Hasil pengujian terhadap kombinasi Cr dan Mr dapat dilihat pada Tabel 6.3 berikut ini.

Tabel 6.3 Hasil pengujian kombinasi cr dan mr

Cr : *Mr*	Nilai *Fitness*										Rata-rata *fitness*
	Pengujian ke-i										
	1	2	3	4	5	6	7	8	9	10	
0.1:0.9	0.944	0.944	0.944	0.944	0.972	0.944	0.972	0.972	0.944	0.917	0.950
0.2:0.8	0.972	0.944	0.944	0.944	0.944	0.944	0.944	0.944	0.944	0.944	0.947

0.3:0.7	0.944	0.944	0.972	0.972	0.972	0.972	0.944	0.944	0.944	0.972	0.958
0.4:0.6	0.944	0.972	0.972	0.944	0.972	0.972	0.972	0.972	0.944	0.972	0.964
0.5:0.5	0.972	0.944	0.972	0.972	0.944	0.944	0.944	0.944	0.972	0.944	0.956
0.6:0.4	0.944	0.944	0.944	0.972	0.972	0.972	0.944	0.944	0.944	0.972	0.956
0.7:0.3	0.944	0.944	0.944	0.944	0.972	0.944	0.944	0.944	0.972	0.972	0.953
0.8:0.2	0.944	0.944	0.972	0.944	0.944	0.972	0.944	0.944	0.944	0.944	0.950
0.9:0.1	0.972	0.944	0.944	0.944	0.944	0.944	0.944	0.944	0.944	0.917	0.944

Berdasarkan Tabel 6.3 nilai *fitness* rata-rata terbesar adalah 0,964 yang terdapat pada nilai cr 0,4 dan nilai mr 0,6 Grafik pengaruh perubahan nilai probabilitas *crossover* dan probabilitas mutasi terhadap nilai *fitness* dapat dilihat pada gambar berikut.



Gambar 6.3 Grafik Pengujian Nilai Probabilitas *Crossover* dan Probabilitas Mutasi Terhadap Nilai *Fitness*

Rata-rata nilai *fitness* yang didapatkan sangat bervariasi. Belum ada ketetapan nilai *Cr* maupun *Mr* yang digunakan untuk memperoleh solusi optimal. Suatu permasalahan sangat mempengaruhi nilai kombinasi yang tepat (Mahmudy, 2013). Nilai *fitness* terendah terdapat pada kombinasi nilai *Cr* 0,9 dan nilai *Mr* 0,1 yang disebabkan oleh nilai probabilitas mutasi (*Mr)* yang terlalu kecil sehingga variasi nilai yang dihasilkan pun semakin sedikit. Pada pengujian ini menghasilkan *fitness* rata-rata terbaik adalah 0,964, yaitu pada kombinasi *crossover rate* 0,4 dan *mutation rate* 0.6.

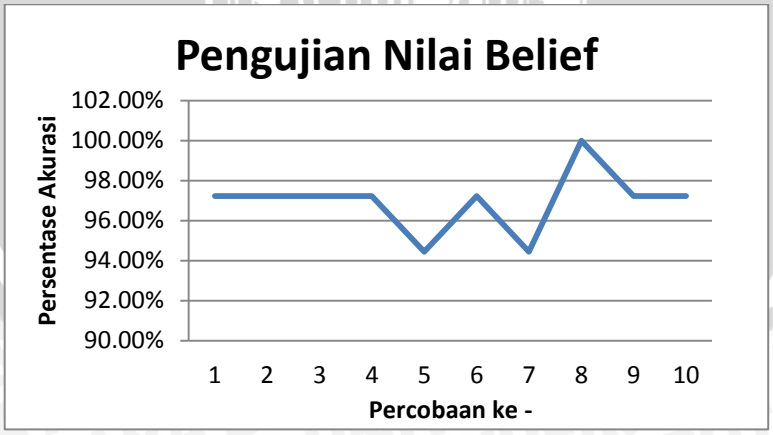
6.2 Pengujian Nilai *Belief* Menggunakan Parameter Algoritma Genetika

Pengujian ini bertujuan untuk melihat besar persentase akurasi dari nilai belief yang dihasilkan dari proses algoritma genetika menggunakan parameter optimal yang telah diuji sebelumnya. Pengujian ini menggunakan data latih dan dilakukan sebanyak 10 kali, kemudian hasil persentase akan dirata-rata. Parameter yang digunakan adalah parameter-parameter yang telah diuji sebelumnya, yaitu ukuran populasi 120, jumlah generasi 100, probabilitas *crossover* 0.4, dan probabilitas mutasi 0,6. Hasil 10 kali pengujian akurasi ditunjukkan pada Tabel 6.4.

Tabel 6.4 Pengujian Akurasi

Percobaan ke-*i*	Hasil Persentase Akurasi
1	97.22%
2	97.22%
3	97.22%
4	97.22%
5	94.44%
6	97.22%
7	94.44%
8	100.00%
9	97.22%
10	97.22%

Hasil pengujian akurasi yang ditunjukkan pada Tabel 6.7 menunjukkan rata-rata akurasi yang dihasilkan dari 10 kali percobaan adalah 96,94%. Grafik pengujian akurasi menggunakan nilai *belief* yang telah dioptimasi ditunjukkan pada Gambar 6.4.



Gambar 6.4 Grafik Pengujian Akurasi

Nilai belief yang dihasilkan dari 10 kali percobaan ini menunjukkan hasil rata-rata yang mendekati 100%. Pada percobaan ke-8 persentase akurasi nilai belief mencapai 100%.

Nilai belief ini didapatkan menggunakan parameter algoritma genetika yang telah diuji dan paling optimal, yaitu probabilitas crossover 0,4, probabilitas mutasi 0,6, ukuran populasi 120 dan jumlah generasi 100. Nilai belief optimal terdapat pada percobaan ke-8 dan setiap satu kali percobaan waktu yang dibutuhkan adalah kurang lebih 1 menit. Lama proses komputasi kasus ini menunjukkan waktu yang cukup cepat dengan jumlah generasi sebanyak 100 generasi.

6.3 Nilai *Belief* Optimal

Setelah dilakukan pengujian dengan menggunakan parameter algoritma yang dianggap paling baik, sistem telah menghasilkan nilai *belief* optimal. Tabel 6.5 menunjukkan nilai *belief* yang optimal dengan nilai *fitness* 1.

Tabel 6.5 Nilai *belief* optimal

	Pneumonia	Bronkitis Akut	URTI
G1	0.94797	0.21422	0.35317
G2	0.42682	0.64908	0.91603
G3	0.94625	0	0
G4	0.95425	0.41291	0.16521
G5	0	0.38087	0.06988
G6	0.57159	0.79116	0
G7	0	0.8188	0.64475
G8	0	0.03981	0.19268
G9	0.36919	0.73912	0
G10	0.52216	0	0
G11	0	0.24564	0
G12	0	0	0.2815
G13	0.88744	0.89944	0
G14	0	0.36865	0
G15	0.25977	0	0
G16	0.10791	0.10072	0
G17	0	0.89731	0.66624

6.4 Pengujian Akurasi

Pengujian hasil akurasi nilai *belief* pakar dilakukan untuk mengetahui hasil perbandingan antara nilai *belief* yang langsung diberikan oleh pakar dan nilai *belief* yang dioptimasi dengan algoritma genetika dalam mendeteksi penyakit melalui gejalanya. Data yang digunakan dalam pengujian ini sebanyak 17 data kasus yang dilampirkan pada Lampiran D.

Tabel 6.6 Hasil Perbandingan Diagnosa

Kasus	Diagnosa Pakar	Hasil Diagnosa dengan Nilai Belief dari Pakar	Nilai Belief dari Pakar	Hasil Diagnosa dengan Nilai Belief Hasil Optimasi	Nilai Belief dari Hasil Optimasi
1	Pneumonia	Bronkitis Akut	0.78	Pneumonia/Bronkitis Akut	0.947
2	Bronkitis Akut/URTI	Bronkitis Akut/URTI	0.46	Bronkitis Akut/URTI	0.644
3	Bronkitis Akut/URTI	URTI	0.75	Pneumonia/Bronkitis Akut/URTI	0.577
4	Pneumonia	Bronkitis Akut	0.223	Pneumonia	0.35831
5	Pneumonia	Pneumonia	0.46	Pneumonia/Bronkitis Akut/URTI	0.49004
6	Bronkitis Akut	Bronkitis Akut	0.9456	Bronkitis Akut/URTI	0.5
7	Bronkitis Akut	Bronkitis Akut	0.43573	Pneumonia/Bronkitis Akut/URTI	0.48718
8	Bronkitis Akut/URTI	Bronkitis Akut	0.6	Bronkitis Akut/URTI	0.58
9	Bronkitis Akut	Bronkitis Akut	0.45	Bronkitis Akut/URTI	0.35897
10	Pneumonia	Pneumonia	0.871	Pneumonia	0.93851
11	Pneumonia	Bronkitis Akut	0.54	Bronkitis Akut	0.605
12	URTI	Bronkitis Akut	0.74783	-	0.43077
13	Pneumonia	Bronkitis Akut	0.799	Pneumonia/Bronkitis Akut	0.768
14	Pneumonia	Pneumonia/Bronkitis Akut	0.99	Bronkitis Akut	0.71
15	URTI	Bronkitis Akut	0.5	Bronkitis Akut/URTI	0.706
16	URTI	Bronkitis Akut	0.5	Bronkitis Akut/URTI	0.716
17	Pneumonia	Bronkitis Akut	0.891	Pneumonia/Bronkitis Akut	0.585

Berdasarkan Tabel 6.6 berikut adalah analisa hasil pengujian, antara lain:

1. Pada kasus 3, 5, dan 7 sistem menghasilkan hasil diagnosa ketiga penyakit yang ada. Hal ini dapat terjadi karena penyakit dalam penelitian ini hanya 3 dan gejala pada kasus tersebut tidak banyak sehingga hasilnya kurang spesifik.

2. Pada kasus 12 sistem tidak menghasilkan hasil diagnosa karena nilai belief terbesar terdapat pada nilai *thetha* (θ). Dan masih terjadi kesalahan diagnosa pada kasus 11 dan 14. Hal ini terjadi karena sistem belum mempunyai pengetahuan untuk kasus ini sehingga diperlukan data latih yang lebih banyak.
3. Berdasarkan konsultasi dengan pakar diagnosa yang menghasilkan 3 penyakit sekaligus dinyatakan benar. Dalam mendiagnosa, dokter dapat menemui masalah ketika diagnosa penyakit belum bisa ditetapkan secara pasti. Hal ini dapat terjadi ketika gejala yang dijabarkan tidak lengkap sehingga diperlukan pemeriksaan lanjut untuk mendapatkan diagnosa yang pasti.

Hasil pengujian akurasi terhadap 17 data kasus penyakit infeksi saluran pernafasan setelah dilakukan optimasi pada nilai *belief* metode Dempster-Shafer menggunakan algoritma genetika adalah 82,35%. Sedangkan hasil pengujian menggunakan nilai *belief* yang diberikan pakar adalah 52,94%. Ini menunjukkan bahwa metode Dempster Shafer dapat digunakan dalam mendiagnosa penyakit infeksi saluran pernafasan dan pembangkitan nilai belief menggunakan algoritma genetika dapat meningkatkan nilai akurasi.

Pengujian ini bernilai benar jika hasil diagnosa sistem sama dengan hasil diagnosa pakar. Pengujian ini juga bernilai benar jika hasil diagnosa pakar terdapat dalam hasil diagnosa sistem walaupun sistem mengeluarkan hasil diagnosa lebih dari satu penyakit. Hal ini menjadi penyebab terjadi peningkatan akurasi walaupun hasil diagnosa pada sistem kurang spesifik.

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan hasil pengujian dan analisis yang dilakukan pada Implementasi Algoritma Genetika Untuk Optimasi Nilai Densitas Metode Dempster-Shafer pada Deteksi Penyakit Infeksi Saluran Pernafasan, maka dapat diambil kesimpulan sebagai berikut:

1. Implementasi algoritma genetika terhadap optimasi nilai *belief* pada metode Dempster Shafer dilakukan dengan mengubah nilai *belief* menjadi kromosom. Kemudian dilakukan optimasi dengan menerapkan menerapkan metode *Extended Intermediate Crossover*, metode *Random Mutation* dan metode *Elitism Selection*. Selanjutnya nilai *belief* ini digunakan pada perhitungan Dempster Shafer untuk mendeteksi penyakit infeksi saluran pernafasan.
2. Parameter Algoritma Genetika yang terbaik untuk mengoptimasi nilai yaitu probabilitas crossover 0,4, probabilitas mutasi 0,6, ukuran populasi 120 dan jumlah generasi 100. Berdasarkan pengujian populasi menunjukkan bahwa semakin besar jumlah populasi maka akan menghasilkan nilai *fitness* yang tinggi. Hal ini juga berlaku pada ukuran jumlah generasi, bertambahnya jumlah generasi mengakibatkan meningkatnya nilai *fitness* yang dihasilkan.
3. Akurasi yang didapat oleh sistem ini dalam melakukan deteksi penyakit infeksi saluran pernafasan terhadap data uji adalah sebesar 82,35%. Penerapan algoritma genetika terhadap optimasi nilai *belief* pada metode Dempster-Shafer menunjukkan hasil yang lebih baik dibandingkan sistem yang menerapkan metode Dempster Shafer menggunakan nilai *belief* yang diperoleh dari pakar yang hanya menghasilkan akurasi sebesar 52,94%

7.2 Saran

Berdasarkan penelitian yang sudah dilakukan, optimasi nilai densitas pada metode Dempster-Shafer masih memiliki kekurangan. Saran yang dapat diberikan untuk pengembangan sistem ini lebih lanjut antara lain:

1. Inisialisasi awal menggunakan nilai belief yang diberikan pakar untuk mempercepat waktu perhitungan.
2. Untuk meningkatkan tingkat akurasi dapat dilakukan pengujian multiclass untuk memastikan kebenaran diagnosa ketika hasil diagnosa yang didapatkan lebih dari satu diagnosa.
3. Mengembangkan metode crossover, metode mutasi, dan seleksi yang lain sehingga dapat dianalisa pengaruh metode yang digunakan apakah dapat menghasilkan nilai *fitness* lebih baik atau tidak.
4. Menggunakan metode optimasi lainnya seperti Ant Colony Algorithm.
5. Menambah data kasus yang ada.

DAFTAR PUSTAKA

Asthma and Respiratory Foundation of New Zealand (Inc.). (2010). Retrieved from What is Acute Bronchitis?: www.asthmafoundation.org.nz

Upper Respiratory Tract Infection. (2012, November 12). *EMIS*, pp. 1-3.

Betz, C. L. (2002). *Buku Saku Keperawatan Pediatrik edisi 3.* EGC.

Evertsen, J., Baumgardner, D., Regnery, A., & Banerjee, I. (2010). Primary Care Respiratory Journal. *Diagnosis and management of pneumonia and bronchitis in outpatient primary care practices*, 237-241.

Heckerling, P. S., Tape, Tape, T. G., & Wigton, R. S. (2004). Use of Genetic Algorithm for Neural Network to Predict Community-Acquired Pneumonia. *Artificial Intelligence in Medicine 30*, 71-84.

Indonesia, Perhimpunan Dokter Paru. (2003). *PEDOMAN DIAGNOSIS & PENATALAKSANAAN DI INDONESIA.*

Kurniawati, D. P. (2014). *Implementasi Metode Dempster Shafer Pada Sistem Pakar Untuk Diagnosa Jenis-jenis Penyakit Diabetes Melitus.* Semarang: Fakultas Ilmu Komputer, Universitas Dian Nuswantoro.

Mahmudy, W. F. (2013). *Algoritma Evolusi.* Malang: Universitas Brawijaya.

Melbye, H. (1992). Diagnosis of Pneumonia in Adults in General Practice. Norway.

Muscari. (2005). *Panduan Belajar Keperawatan Pediatrik edisi 3.* EGC.

Raviglione, M. (2008). Practical Approach to Lung Health. Geneva, Switzerland.

Rohilla, A., Sharma, V., & Kumar, S. (2013). UPPER RESPIRATORY TRACT INFECTIONS: AN OVERVIEW. *International Journal of Current Pharmaceutical Research Vol.5*, 1-3.

Sulistyohati, A., & Hidayat, T. (2008). Aplikasi Sistem Pakar Diagnosa Penyakit Ginjal Dengan Metode Dempster-Shafer. *Seminar Nasional Aplikasi Teknologi Informasi (SNATI 208)*, E1-E6.

UNICEF/WHO. (2006). Pneumonia: The Forgotten Killer of Children.

Wahyuni, E. G., & Prijodiprojo, W. (2013). Prototype Sistem Pakar untuk Mendeteksi Tingkat Resiko Penyakit Jantung Koroner dengan Metode Dempster-ShaferPrototype Sistem Pakar untuk Mendeteksi Tingkat Resiko Penyakit Jantung Koroner dengan Metode Dempster-Shafe. *IJCCS*, 133-144.

Widowati, U. (2015, Mei 14). Dipetik Juli 13, 2015, dari CNN Indonesia: http://www.cnnindonesia.com/gaya-hidup/20150513163407-255-53129/10-penyakit-paling-mematikan-di-indonesia/

Zamani, A. M., Amaliah, B., & Munif, A. (2012). Implementasi Algoritma Genetika pada Struktur Backpropagation Neural Network untuk Klasifikasi Kanker Payudara. *Jurnal Teknik ITS Vol.1*, 222-227.

Sutojo, T., Mulyanto, E., Suhartono, V., 2011. *Kecerdasan Buatan*. Penerbit ANDI, Yogyakarta.

LAMPIRAN A DAFTAR NILAI *BELIEF* DARI PAKAR

Kode	Gejala	Pneumonia	Bronkitis Akut	URTI
G1	Fever	0.9	0.5	0.5
G2	Cough	0.9	0.5	0.25
G3	chest pain	1	0	0
G4	Chills	0.8	0.3	0.2
G5	sore throat	0	0.5	0.9
G6	Dyspnea (sesak napas)	0.9	0.5	0
G7	sinus pain	0	0.6	0.2
G8	head congestion	0	0.5	0.8
G9	chest congestion	0.9	0.5	0
G10	Sputum: kental, kuning kehijauan	0.9	0	0
G11	Sputum: Jernih, putih	0	0.5	0
G12	Rhinorrhea (runny nose)	0	0	0.75
G13	temp > 100F	0.75	0.4	0
G14	sinusitis removed	0	0.6	0
G15	> 10days	1	0	0
G16	7-10 days	0.75	0.9	0
G17	2-7 days	0	0.75	0.9

LAMPIRAN B DAFTAR GEJALA

Kode	Gejala	Keterangan
G1	Fever	Demam
G2	Cough	Batuk
G3	Chest pain	Ketidaknyamanan atau rasa sakit yang dirasakan di sepanjang bagian depan tubuh antara leher dan perut bagian atas. (Chest pain)
G4	Chills	Menggigil yang mengacu pada rasa dingin setelah berada di lingkungan yang dingin (Chills)
G5	sore throat	Gejala sakit tenggorokan termasuk pada rasa sakit, terbakar atau scratching sensation (sensasi gatal) di bagian belakang tenggorokan, nyeri saat menelan, dan nyeri di leher. (Sore throat)
G6	Dyspnea (sesak napas)	Sulit atau nampak susah bernafas; sesak napas; Saat bernafas membutuhkan tenaga lebih. (Dyspnea)
G7	sinus pain	Sakit atau tekanan yang terjadi pada sekitar daerah sinus (Sinus pain)
G8	head congestion	Rasa sakit dan atau tekanan di kepala dan wajah (Head congestion)
G9	chest congestion	Gejala respirasi yang terjadi oleh karena banyaknya produksi mucus/dahak yang diakibatkan radang/infeksi. Saat bernafas, nafas terasa tidak bersih (Chest congestion)
G10	Sputum: kental, kuning kehijauan	Dahak yang keluar bersamaan dengan batuk dengan tekstur kental dan warna kuning kehijauan (Sputum: kental, kuning kehijauan)
G11	Sputum: Jernih, putih	Dahak yang keluar bersamaan dengan batuk dengan warna putih atau jernih (Sputum: Jernih, putih)
G12	Rhinorrhea (runny nose)	Pilek (Rhinorrhea/runny nose)
G13	Temp > 100F	Suhu badan saat diukur thermometer lebih dari 100˚F atau 37,8˚C (Temperature > 100˚F)
G14	sinusitis removed	Kondisi saat sinus telah dihilangkan (Sinusitis removed)
G15	> 10days	Gejala yang dialami berlangsung lebih dari 10 hari
G16	7-10 days	Gejala yang dialami berlangsung antara 7 sampai 10 hari
G17	2-7 days	Gejala yang dialami berlangsung antara 2 sampai 7 hari

LAMPIRAN C DATA LATIH

No.	Gejala	Diagnosa Pakar
1	G2 Cough	URTI
	G11 Sputum: jernih	
	G5 Sore Throat	
	G17 2 days	
2	G12 Rhinorrhea	URTI
	G5 Sore throat	
	G17 2 days	
3	G1 Fever	Bronkitis Akut
	G7 Sinus pain	
	G6 Dyspnea	
	G9 Chest congestion	
	G8 Head Congestion	
4	G2 Cough	URTI
	G5 Sore throat	
	G8 Head congestion	
	G11 Sputum: jernih	
	G17 a week	
5	G2 cough	Pneumonia
	G1 Fever	
	G13 Temp > 100F	
	G6 Dyspnea	
6	G2 Cough	Pneumonia
	G10 Sputum: kental	
	G4 Chills	
	G1 Fever	
	G13 Temp > 100F	
7	G2 Cough	Bronkitis Akut
	G5 Sore throat	
	G6 Dyspnea	
	G9 Chest congestion	
	G8 Head congestion	
8	G5 sore throat	Pneumonia
	G6 dyspnea	
	G9 chest congestion	
9	G1 Fever	Pneumonia
	G6 Dyspnea	
	G13 Temp>100F	
	G3 Chest pain	

10	G3 Chest pain G9 Chest congetion	Pneumonia
11	G1 Fever G4 Chills G8 Head Congestion G16 7-10 days	Bronkitis Akut
12	G2 Cough G17 7 days G14 Sinusitis removed G5 Sore throat	Bronkitis Akut
13	G2 Cough G7 Sinus pain G8 Head congestion G12 Rhinorrhea G16 7-10 days G11 Sputum: jernih	Bronkitis Akut
14	G1 Fever G2 Cough G10 Sputum: Kental, kuning G14 Sinusitis removed G17 7 days	Bronkitis Akut
15	G1 Fever G4 Chills G13 Temps>100F G6 Dyspnea G12 Rhinorrhea G16 10 days	Pneumonia
16	G2 Cough G8 Head Congestion G11 sputum: jernih, putih	Bronkitis Akut
17	G2 Cough G6 Dyspnea G15 more than 3 weeks G10 sputum kental	Pneumonia
18	G1 Fever G6 Dyspnea G9 Chest congestion G8 Head Congestion G16 7-10 days	Pneumonia
19	G1 Fever G6 Dyspnea G17 2-7 days	Pneumonia
20	G2 Cough	Bronkitis Akut /URTI

	G5 Sore throat	
	G17 2-7 days	
21	G1 Fever	Pneumonia
	G4 Chills	
	G12 Rhiniorrhea	
	G16 7-10 days	
22	G2 Cough	Bronkitis Akut/URTI
	G11 Sputum: putih, jernih	
	G16 7-10 days	
23	G12 Rhinorerea	URTI
	G4 chills	
	G8 head congestion	
24	G1 fever	Pneumonia/Bronkitis Akut
	G2 cough	
	G11 sputum: jernih putih	
	G13 temp >100f	
	G17 2-7 days	
25	G7 sinus pain	Pneumonia/ Bronkitis Akut
	G10 sputum kental kuning	
	G17 2-7days	
	G2 Cough	
26	G1 fever	Bronkitis Akut
	G5 sore throat	
	G7 sinus pain	
	G14 sinusitis removed	
27	G3 chest pain	Pneumonia
	G15 3weeks	
	G2 Cough	
	G11 sputum jernih putih	
28	G1 fever	Pneumonia
	G4 chills	
	G6 dyspnea	
	G17 2-7days	
	G13 temp >100f	
29	G4 chills	Pneumonia
	G5 sore throat	
	G2 Cough	
	G10 sputum kental kuning	
30	G8 head congestion	URTI
	G11 sputum jernih	
	G12 rhinorrhea	
	G16 7-10days	

31	G2 cough	Pneumonia
	G7 sinus pain	
	G9 chest congestion	
32	G3 Chest pain	Pneumonia
	G2 Cough	
	G10 sputum kental kuning	
	G17 2-7 days	
33	G7 sinus pain	Bronkitis Akut
	G1 fever	
	G8 head congestion	
34	G2 cough	Pneumonia
	G3 chest pain	
	G12 rhinorrhea	
	G15 3 weeks	
35	G12 rhinorrhea	Pneumonia
	G15 3weeks	
	G3 chest pain	
	G5 sore throat	
	G16 10 days	
36	G1 fever	Pneumonia
	G5 sore throat	
	G6 dyspnea	
	G2 cough	
	G11 sputum jernih	
	G8 head congestion	

LAMPIRAN D DATA UJI

No	Kasus	Diagnosa Pakar
1	G9 chest congestion G1 fever G13 >100f G4 chills	Pneumonia
2	G7 sinus pain G2 cough G11 sputum jernih G8 head congestion	Bronkitis Akut/URTI
3	G2 cough G12 rhinorrhea G8 head congestion G4 chills	Bronkitis Akut/URTI
4	G14 sinusitis removed G6 dyspnea G10 sputum kental G8 head congestion	Pneumonia
5	G1 fever G14 sinusitis removed G15 >10 days	Pneumonia
6	G16 13 days G14 Sinusitis removed G5 sore throat G2 Cough G7 Sinus Pain	Bronkitis Akut
7	G2 Cough G11 Sputum: jernih G12 Rhinorrhea G16 10 days	Bronkitis Akut
8	G14 sinusitis removed G8 head congestion G17 2-7 days	Bronkitis Akut/URTI
9	G5 sore throat G8 head congestion G12 Rhinorrhea (runny nose) G15 >10 days	Bronkitis Akut
10	G3 chest pain G9 chest congestion G14 sinusitis removed G15 >10 days	Pneumonia
11	G2 cough	Pneumonia

	G7 sinus pain G9 chest congestion	
12	G8 head congestion G11 sputum jernih G12 rhinorrhea G16 7-10days	URTI
13	G1 Fever G6 Dyspnea G9 Chest congestion G8 Head Congestion G16 7-10 days	Pneumonia
14	G1 Fever G6 Dyspnea G17 2-7 days	Pneumonia
15	G2 Cough G11 Sputum: jernih G5 Sore Throat G17 2 days	URTI
16	G2 Cough G5 Sore throat G8 Head congestion G11 Sputum: jernih G17 a week	URTI
17	G5 sore throat G6 dyspnea G9 chest congestion	Pneumonia

Malang, 5 Desember 2015

dr. Triwahju Astuti M.Kes, Sp.P(K)