OPTIMASI PENJADWALAN PERAWAT MENGGUNAKAN ALGORITMA GENETIKA

SKRIPSI

KONSENTRASI KOMPUTASI CERDAS DAN VISUAL

Diajukan untuk memenuhi persyaratan memperoleh gelar sarjana komputer

Disusun oleh :

RIFQY ROSYIDAH ILMI

NIM. 115060801111045

PROGRAM STUDI INFORMATIKA / ILMU KOMPUTER

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2015

LEMBAR PERSETUJUAN

OPTIMASI PENJADWALAN PERAWAT MENGGUNAKAN ALGORITMA GENETIKA

SKRIPSI

KONSENTRASI KOMPUTASI CERDAS DAN VISUAL

Diajukan untuk memenuhi persyaratan memperoleh gelar sarjana komputer

Disusun Oleh :

RIFQY ROSYIDAH ILMI

NIM. 115060801111045

Telah diperiksa dan disetujui oleh

Dosen Pembimbing

Pembimbing I,	**Pembimbing II,**
Wayan F. Mahmudy, S.Si, MT, Ph.D	**Dian Eka Ratnawati, S.Si, M.Kom**
NIP. 19720919 199702 1 001	**NIP. 19731906 200212 2 001**

LEMBAR PENGESAHAN

OPTIMASI PENJADWALAN PERAWAT MENGGUNAKAN ALGORITMA GENETIKA

SKRIPSI

KONSENTRASI KOMPUTASI CERDAS DAN VISUAL

Diajukan untuk memenuhi persyaratan memperoleh gelar sarjana komputer

Disusun Oleh :

RIFQY ROSYIDAH ILMI

NIM. 115060801111045

Skripsi ini telah diuji dan dinyatakan lulus pada tanggal 22 Mei 2015

Penguji I,	**Penguji II,**
Suprapto, ST., MT	**Drs. Achmad Ridok, M.Kom**
NIP. 19710727 199603 1 001	**NIP. 19680825 199403 1 002**

Penguji III,

Barlian Henryranu P, ST., MT.
NIK. 82102406110254

Mengetahui,

Ketu Program Studi Informatika / Ilmu Komputer

Drs. Marji, MT.
NIP. 196708011992031001

PERNYATAAN ORISINALITAS SKRIPSI

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 dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata di dalam 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, 26 Mei 2015

Mahasiswa

Rifqy Rosyidah Ilmi

NIM. 115060801111045

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah SWT yang telah melimpahkan segala Rahmat, Karunia dan Hidayah-Nya sehingga Penulis dapat menyelesaikan skripsi dengan judul ”Optimasi Penjadwalan Perawat Menggunakan Algoritma Genetika“.

Skripsi ini diajukan sebagai syarat ujian skripsi dalam rangka untuk memperoleh gelar Sarjana Komputer di Program Teknologi Informasi Dan Ilmu Komputer (PTIIK), Program Studi Informatika/Ilmu Komputer, Universitas Brawijaya Malang. Atas terselesaikannya skripsi ini, penulis mengucapkan terima kasih kepada:

1. Bapak Wayan Firdaus Mahmudy, S.Si, MT, Ph.D, selaku dosen pembimbing skripsi pertama yang telah meluangkan waktu dan juga memberikan pengarahan bagi penulis.
2. Ibu Dian Eka Ratnawati, S.Si, M.Kom, selaku dosen pembimbing skripsi kedua yang telah meluangkan waktu dan juga memberikan pengarahan bagi penulis.
3. Drs. Marji, MT selaku ketua Program Studi Informatika/Ilmu Komputer Program Teknologi Informasi & Ilmu Komputer Universitas Brawijaya
4. Ir. Sutrisno, MT., selaku Ketua Program Teknologi Informasi & Ilmu Komputer Universitas Brawijaya.
5. Orang tua dan saudara penulis yang telah memberi dukungan dan semangat dalam menyelesaikan skripsi ini.
6. Sahabat penulis serta teman – teman Teknik Informatika angkatan 2011 yang memberi dukungan dan semangat dalam menyelesaikan skripsi ini.
7. Segenap Bapak dan Ibu dosen yang telah mendidik dan mengajarkan ilmunya kepada penulis selama menempuh pendidikan di Program Teknologi Informasi & Ilmu Komputer Universitas Brawijaya.
8. Segenap staff dan karyawan di Program Studi Teknologi Informasi & Ilmu Komputer Universitas Brawijaya yang telah banyak membantu dalam hal administrasi penulis dalam pelaksanaan penyusunan skripsi ini.

9. Penulis menyadari bahwa skripsi ini tentunya tidak terlepas dari berbagai kekurangan dan kesalahan. Oleh karena itu, segala kritik dan saran yang bersifat membangun sangat Penulis harapkan dari berbagai pihak demi penyempurnaan penulisan skripsi ini.

Akhirnya penulis berharap agar skripsi ini dapat memberikan sumbangan dan manfaat bagi semua pihak yang berkepentingan.

Malang, Mei 2015

Penulis,

Rifqy Rosyidah Ilmi

NIM. 115060801111045

ABSTRAK

Rifqy Rosyidah Ilmi. 2015 : Optimasi Penjadwalan Perawat Menggunakan Algoritma Genetika. Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya, Malang.

Dosen Pembimbing : Wayan Firdaus Mahmudy, S.Si., M.T., Ph.D , Dian Eka Ratnawati, S.Si., M.Kom.

Penjadwalan perawat merupakan suatu masalah kritis pada suatu rumah sakit terutama pada ruang Intensive Care Unit (ICU) dengan pasien yang membutuhkan perawatan khusus. Dampak dari panjangnya jam kerja perawat serta tugas yang banyak dikhawatirkan akan mempengaruhi kualitas kinerja, kondisi fisik maupun kehidupan sosial. Dalam penelitian ini, diterapkan algoritma genetika untuk menyelesaikan permasalahan penjadwalan perawat. Digunakan representasi permutasi bilangan integer dengan panjang kromosom 360 yang setiap angka pada gennya merepresentasikan nomor id perawat. Metode crossover yang digunakan yaitu *one cut-point crossover*, metode mutasi *reciprocal exchange mutation* dan diseleksi dengan *elitism selection*. Dari hasil pengujian yang dilakukan diperoleh parameter optimal yaitu ukuran populasi sebesar 300 individu dengan rata-rata *fitness* sebesar 0,26853, 105 generasi dengan rata-rata *fitness* sebesar 0,56568 dan kombinasi cr = 0.6 dan mr 0.4 dengan rata-rata *fitness* sebesar 0,56568. Hasil akhir berupa jadwal jaga perawat pada ruang Intensive Care Unit (ICU) selama 1 bulan.

Kata kunci : Algoritma Genetika, Penjadwalan Perawat

ABSTRACT

Rifqy Rosyidah Ilmi. 2015 : ***Optimization of Nurse Scheduling with Genetic Algorithms. Information Technology and Computer Science Program, Brawijaya University, Malang.***

***Advisor* : Wayan Firdaus Mahmudy, S.Si., M.T., Ph.D , Dian Eka Ratnawati, S.Si., M.Kom.**

Nurse scheduling is a critical issue in the management of Intensive Care Unit (ICU) department. Under the intense work environment, it is imperative to make quality nurse schedules in a most costand time effective way. In this research, genetic algorithm is applied to optimize the nurse scheduling problem (NSP). The length of chromosome is 360, each gen is represented by id number of nurse. The method of crossover is one cut-point crossover, mutation method is reciprocal exchange mutation and selection is using elitsm method. Optimal parameter of popsize is 300 with average fitness number of 0,26853, 105 generation with average fitness number of 0,56568 and combination of Cr = 0.6 and Mr = 0.4 with average fitness number of 0,56568. The final result from this research is a month of nurse schedule in Intensive Care Unit (ICU) department.

Keywords*: Genetic Algorithms, Nurse Scheduling*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

Bab ini berisi pembahasan yang berkaitan dengan latar belakang permasalahan topik yang diteliti, permasalahan yang akan dibahas dari topik, batasan permasalahan yang dibahas, tujuan dan manfaat dari topik yang akan dibahas serta sistematika penulisan topik.

1.1 Latar Belakang

Perawat didefinisikan sebagai seseorang yang memiliki pengetahuan, keterampilan dan kewenangan untuk memberikan asuhan keperawatan pada orang lain berdasarkan ilmu dan kiat yang dimilikinya dalam batas-batas kewenangan yang dimilikinya (UU RI No. 23 tahun 1992). Perawat sebagai ujung tombak dalam pelayanan di rumah sakit, mempunyai tugas memberikan asuhan keperawatan antara lain mengkaji kebutuhan pasien, merencanakan tindakan keperawatan, melaksanakan rencana tindakan, mengevaluasi hasil asuhan keperawatan, mendokumentasikan asuhan keperawatan dan berperan serta dalam melakukan penyuluhan (Sabarulin, 2013). Dampak dari panjangnya jam kerja perawat serta tugas yang banyak dikhawatirkan akan mempengaruhi kualitas kinerja, kondisi fisik maupun kehidupan sosial. Salah satu cara yang dapat diusahakan pihak managemen rumah sakit adalah membuat kebijakan penjadwalan kerja yang bisa membagi panjangnya jam kerja secara adil kepada seluruh perawat yang tersedia. Penjadwalan dengan shift menjadi lebih disukai dibanding non-shift.

Penjadwalan daftar jaga perawat merupakan sebuah proses penjadwalan yang mengatur waktu kerja perawat di sebuah rumah sakit. Dalam proses penjadwalan daftar jaga perawat ada beberapa komponen yang digunakan yaitu perawat, ruang, hari, dan shift (Tjajo, 2008) . Permasalahan penjadwalan perawat atau dikenal dengan *Nurse Scheduling Problem* (NSP) menjadi hal yang menantang karena dengan jumlah perawat yang relatif terbatas dibanding banyaknya pasien dan shift kerja, penjadwal dituntut untuk mendapatkan jadwal dengan beban kerja seadil mungkin untuk setiap perawat serta memenuhi batasan-

batasan penjadwalan yang ada. Pemenuhan semua batasan penjadwalan seringkali terhambat ketika satu batasan terpenuhi, namun ternyata batasan lain terlanggar. Misalnya batasan jumlah maksimal libur per perawat mungkin terlanggar ketika batasan jumlah minimal perawat dalam satu shift kerja terpenuhi.

Berdasarkan permasalahan penjadwalan perawat di atas maka diperlukan suatu algoritma untuk memecahkan masalah penjadwalan yang ada. Algoritma merupakan kumpulan perintah untuk menyelesaikan suatu masalah. Pada penelitian sebelumnya Atmasari (2010) menerapkan suatu formulasi matematika dengan menggunakan metode *goal programming* (GP) untuk membuat sistem penjadwalan perawat UGD yang lebih optimal. Hasil penelitian menggunakan metode *goal programming* tersebut menghasilkan jadwal lebih optimal dibandingkan jadwal manual namun jumlah shift perawat tidak merata (Atmasari, 2010).

Untuk permasalahan penjadwalan perawat, algoritma genetika dapat digunakan sebagai metode pengembangan sistem yang dapat membantu mengoptimalkan daftar jaga perawat. Permasalahan dengan model matematika yang kompleks atau bahkan sulit dibangun dapat diselesaikan menggunakan algoritma genetika (Mahmudy, 2013). Penelitian tentang penjadwalan perawat telah dilakukan oleh Ulya dengan algoritma genetik menggunakan pemodelan dua tingkat dalam permasalahan penjadwalan perawat pada unit gawat darurat rumah sakit umum xyz surabaya (Ulya, 2010). Pemodelan dirancang untuk menghasilkan solusi berupa jadwal perawat yang bisa memenuhi batasan – batasan penjadwalan yang berlaku pada setiap tingkat pemodelan, membagi hari libur dan shift kerja secara lebih merata, serta mendekati jadwal kerja yang diinginkan perawat (Ulya, 2010). Penelitian serupa tentang penjadwalan perawat juga dilakukan oleh (Tjatjo, 2008) menggunakan algoritma genetika pada penjadwalan perawat. Pada penelitian (Tjatjo, 2008) representasi kromosom berupa pengkodean biner, seleksi dengan *roulette wheel*. Dalam penelitian tersebut dihasilkan populasi terbaik yaitu 50 dan konvergensi nilai pada generasi ke 50 dengan probabilitas *crossover* terbaik 0,6 dan probabilitas mutasi terbaik 0,4.

Pada skripsi ini, diterapkan representasi kromosom berupa permutasi integer, metode *crossover one-cut point*, metode mutasi *reciprocal exchange mutation*, metode seleksi *elitsm* serta studi kasus yang berbeda dengan penelitian sebelumnya. Algoritma genetika digunakan untuk mendapatkan hasil kombinasi kromosom yang optimal sebagai penentuan komposisi jadwal serta perawat yang tepat sehingga didapat hasil penjadwalan yang sesuai kriteria.

1.2 Rumusan Masalah

Rumusan masalah dalam skripsi ini adalah :

1. Bagaimana representasi kromosom dari solusi yang diinginkan.
2. Bagaimana mengimplementasikan metode Algoritma Genetika untuk menyelesaikan masalah penjadwalan perawat.
3. Bagaimana mengukur kebaikan solusi yang dihasilkan Algoritma Genetika.

1.3 Batasan Masalah

Dari permasalahan yang dirumuskan diatas, maka batasan permasalahan dalam skripsi ini adalah :

1. Penjadwalan dibuat untuk periode satu bulan (akumulasi 30 hari).
2. Penjadwalan tidak memperhatikan variable biaya.
3. Data yang digunakan merupakan data pada Rumah Sakit Umum Daerah Ibnu Sina Gresik.
4. Jumlah shift kerja perawat adalah 3 shift kerja dalam sehari, yaitu shift pagi, sore dan malam.

1.4 Tujuan

Tujuan dari skripsi ini adalah :

1. Merepresentasikan data-data penjadwalan ke dalam bentuk kromosom yang akan mewakili solusi permasalahan.
2. Mengimplementasikan metode Algoritma Genetika untuk mendapatkan solusi optimal dalam menyelesaikan permasalahan penjadwalan perawat.
3. Mengukur kebaikan solusi yang dihasilkan Algoritma Genetika.

1.5 Manfaat

Manfaat dari skripsi ini adalah :

1. Optimasi penjadwalan perawat menggunakan algoritma genetika dapat menjadi solusi atas permasalahan penjadwalan perawat seperti ketidakadilan pembagian jadwal sehingga dapat menambah kualitas kerja perawat dalam melayani pasien.

1.6 Sistematika Penulisan

Pembuatan skripsi ini dilakukan dengan sistematika penulisan sebagai berikut :

1. BAB I PENDAHULUAN
   Berisi latar belakang, permasalahan, tujuan, batasan masalah, manfaat serta sistematika penulisan skripsi.
2. BAB II KAJIAN PUSTAKA DAN DASAR TEORI
   Berisi teori tentang kajian teori yang berhubungan dalam penelitian skrispsi ini serta terori dasar tentang perawat, penjadwalan, algoritma genetika dan teori-teori yang berhubungan dalam skripsi ini.
3. BAB III METODOLOGI PENELITIAN
   Berisi algoritma-algoritma yang digunakan dalam pembuatan sistem optimasi penjadwalan perawat menggunakan algoritma genetika.

4. BAB IV IMPLEMENTASI

   Berisi tentang penjelasan implementasi dari sistem, bagaimana user interface sistem dan *source code* untuk mengembankan sistem

5. BAB V PENGUJIAN DAN ANALISIS

   Berisi tentang penjelasn proses pengujian dan hasil pengujian serta analisis dari pengujian tersebut.

6. BAB VI PENUTUP

   Berisi kesimpulan yang diperoleh dari hasil pengujian dan saran-saran untuk pengembangan.

BAB II
KAJIAN PUSTAKA DAN DASAR TEORI

Pada bab ini dibahas kajian pustaka dan dasar teori yang digunakan untuk menunjang penulisan skripsi. Pada kajian pustaka dibahas penelitian yang telah ada serta penelitian yang akan diusulkan. Pada dasar teori dibahas teori algoritma genetika serta penjadwalan perawat.

2.1 Kajian Pustaka

Penelitian sebelumnya tentang penjadwalan yaitu penjadwalan perawat unit gawat darurat dengan menggunakan goal programming dilakukan oleh Atmasari (2010). Baik buruknya penjadwalan perawat yang dilakukan oleh menejemen rumah sakit memegang peranan penting dalam mempengaruhi kinerja rumah sakit dimata pengguna jasa rumah sakit. Oleh sebab itu diperlukan suatu penjadwalan perawat yang baik, sehingga pelayanan perawat terhadap pasien akan menjadi baik pula.

Penerapan metode *Goal Programmig* diterapkan untuk membuat model penjadwalan perawat UGD di Rumah Sakit Umum Haji Surabaya. Model yang dibuat didasarkan pada peraturan- yang berlaku di rumah sakit dan preferensi dari perawat (keinginan perawat misalnya dalam hal pembagian shift secara adil dan hari libur kerja). Disamping itu juga dipertimbangkan kebijakan dari rumah sakit. Preferensi perawat diambil dari survey yang dilakukan untuk kepentingan penelitian yang meliputi pertimbangan keadilan dalam hal pembagian shift malam dan hari libur kerja. Pengembangan model penjadwalan yang sudah ada sebelumnya serta menggunakan bantuan program komputer LINGO memberikan hasil yang dapat memberikan informasi mengenai bagaimana membuat model penjadwalan perawat yang efektif dan efisien dibandingkan dengan penjadwalan manual (Atmasari, 2010).

Penelitian tentang penjadwalan juga dilakukan oleh Ulya dengan judul penggunaan algoritma genetik dengan pemodelan dua tingkat dalam permasalahan penjadwalan perawat pada unit gawat darurat rumah sakit umum xyz surabaya

(Ulya, 2010). Pada penelitian ini dilakukan pemecahan masalah penjadwalan perawat pada UGD dengan menggunakan Genetic Algorithm (GA) dengan pemodelan dua tingkat. Pada tingkat pertama, pemodelan dilakukan untuk menjadwalkan hari libur perawat. Sedangkan pada tingkat kedua, pemodelan dilakukan untuk menjadwalkan shift kerja perawat. Pemodelan dirancang untuk menghasilkan solusi berupa jadwal perawat yang bisa memenuhi batasan – batasan penjadwalan yang berlaku pada setiap tingkat pemodelan, membagi hari libur dan shift kerja secara lebih merata, serta mendekati jadwal kerja yang diinginkan perawat. Solusi dengan nilai *fitness* terbaik didapat dari skenario ujicoba ke-6, dengan nilai maksimal generasi 100, jumlah gen yang dimutasi sebanyak 2 gen, serta jumlah titik potong pada *crossover* sebanyak 5 titik. Sedangkan untuk pembobotan fungsi *fitness*, bobot yang menghasilkan solusi dengan nilai *fitness* yang paling seimbang adalah bobot 1 untuk varians dan 1,5 untuk rasio (Ulya, 2010).

Penelitian serupa juga dilakukan oleh (Tjatjo, 2008) menggunakan algoritma genetika pada penjadwalan perawat. Penjadwalan dilakukan pada suatu rumah sakit di Palu. Sejumlah perawat akan dijadwalkan pada ruang dan shift yang berbeda setiap harinya. Pengkodean yang dilakukan pada penjadwalan ini yaitu pengkodean nilai, *crossover* menggunakan metode *one cut point* , mutasi dilakukan dengan *exchange mutation*. Pada penelitian ini *crossover rate* terbaik adalah 0,6 dan *mutation rate* 0,4. Terjadi konvergensi nilai pada generasi ke 50.

Penelitian yang dilakukan penulis juga optimasi penjadwalan perawat menggunakan algoritma genetika. Data yang digunakan dalam penelitian ini adalah data perawat pada bagian ICU RSUD Ibnu Sina Gresik. Parameter yang digunakan adalah jumlah perawat, jumlah hari dan daftar shift perawat. Pengkodean kromosom menggunakan representasi permutasi nomor perawat. Metode *crossover* yang digunakan yaitu *one cut point* , mutasi dilakukan dengan metode *reciprocal exchange mutation*, seleksi dilakukan dengan metode *elitsm.* Untuk mengukur kebaikan solusi algoritma genetika pada kasus penjadwalan perawat ini akan dilakukan pengujian terhadap jumlah populasi, nilai *crossover rate* serta nilai *mutation rate.*

Pada tabel 2.1 ditunjukkan perbandingan input dan output penelitian sebelumnya serta penelitian yang akan dilakukan.

Tabel 2.1 Kajian Pustaka

No	Judul	Objek	Metode	Input	Proses	Output	Hasil Penelitian
1.	Optimasi Penjadwalan Daftar Jaga Perawat Menggunakan Algoritma Genetika Studi Kasus Pada Rumah Sakit "Undata" Palu. (Tjatjo, 2008)	Penjadwalan perawat	Algoritma Genetika	Bulan, tahun, jumlah individu, jumlah generasi, Pc, Pm.	Inisialisai parameter awal, pembentukan populasi awal, populasi baru, *crossover one cut point*, perhitungan *fitness*, seleksi, penentuan kromosom terbaik	Hasil seleksi generasi akhir, hasil kromosom terbaik, jadwal jaga perawat dengan ruang berbeda setiap hasil.	- Populasi dan generasi optimal sebanyak 50. - *crossover rate* terbaik 0,6 dan *mutation rate* terbaik 0,4.
2.	Penggunaan Algoritma Genetik Dengan Pemodelan Dua Tingkat Dalam Permasalahan Penjadwalan Perawat Pada Unit Gawat Darurat Rumah Sakit Umum Xyz Surabaya (Ulya, 2010)	Penjadwalan perawat	Algoritma Genetika	jumlah generasi, Pc, Pm	Inisialisai parameter awal, pembentukan populasi awal, populasi baru, n-point *crossover*, perhitungan *fitness*, seleksi *tournament*, penentuan kromosom terbaik	Hasil seleksi generasi akhir, hasil kromosom terbaik, dan jadwal jaga perawat di UGD.	- nilai maksimal generasi 100, - jumlah gen yang dimutasi sebanyak 2 gen, - titik potong pada *crossover* sebanyak 5 titik. - pembobotan fungsi *fitness*, bobot 1 untuk varians dan 1,5 untuk rasio
3.	Optimasi Penjadwalan Perawat Menggunakan Algoritma Genetika (Perancangan)	Penjadwalan perawat	Algoritma Genetika	jumlah generasi, Cr, Mr	Inisialisai parameter awal, pembentukan populasi awal, populasi baru, *crossover*	Hasil seleksi generasi akhir, hasil kromosom terbaik, jadwal jaga	-representasi kromosom permutasi dengan banyak gen 360 -penjadwalan untuk ruang ICU

					one cut point, perhitungan *fitness*, seleksi *elitsm*, penentuan kromosom terbaik	perawat pada ruang ICU	

2.2 Algoritma Genetika

Pada sub bab ini akan dibahas definisi algoritma genetika serta struktur algoritma genetika.

2.2.1 Pengertian Algoritma Genetika

Algoritma Genetika (AG) adalah algoritma pencarian yang didasarkan pada mekanisme seleksi alamiah dan genetika alamiah. Algoritma genetika digunakan sebagai algoritma pencarian prameter – prameter optimal. Dalam perkembangannya AG dapat diaplikasikan dalam berbagai masalah seperti learning, peramalan, pemrograman otomatis, dan sebagainya (Suyanto, 2011).

Menurut Mahmudy (2013) dalam bidang industri manufaktur, AG digunakan untuk perencanaan dan penjadwalan produksi. AG juga bisa diterapkan untuk kompresi citra, optimasi penugasan mengajar bagi dosen, penjadwalan dan alokasi ruang ujian, optimasi penjadwalan kuliah, optimasi multi travelling salesman problem (M-TSP), dan penyusunan rute dan jadwal kunjungan wisata yang efisien

Menurut (Michalewicz 1996) dalam (Mahmudy 2013) algoritma genetika diilhami oleh ilmu genetika, karena itu istilah yang digunakan dalam algoritma genetika banyak diadopsi dari ilmu tersebut. Apabila dibandingkan dengan prosedur pencarian dan optimasi biasa, algoritma genetika berbeda dalam beberapa hal sebagai berikut:

- Manipulasi dilakukan terhadap kode dari himpunan parameter (biasa disebut kromosom), tidak secara langsung terhadap parameternya sendiri.

- Proses pencarian dilakukan dari beberapa titik dalam satu populasi, tidak dari satu titik saja.
- Proses pencarian menggunakan informasi dari fungsi tujuan.
- Pencariannya menggunakan stochastic operators yang bersifat probabilistik, tidak menggunakan aturan deterministik.

Kelebihan AG sebagai metode optimasi adalah sebagai berikut (Mahmudy, 2013):

a. AG merupakan algoritma yang berbasis populasi yang memungkinkan digunakan pada optimasi masalah dengan ruang pencarian (search space) yang sangat luas dan kompleks. Properti ini juga memungkinkan AG untuk melompat keluar dari daerah optimum lokal
b. Individu yang ada pada populasi bisa diletakkan pada beberapa sub-populasi yang diproses pada sejumlah komputer secara paralel. Hal ini bisa mengurangi waktu komputasi pada masalah yang sangat kompleks. Penggunaan sub-populasi juga bisa dilakukan pada hanya satu komputer untuk menjaga keragaman populasi dan meningkatkan kualitas hasil pencarian.
c. AG menghasilkan himpunan solusi optimal yang sangat berguna pada peyelesaian masalah dengan banyak obyektif.
d. AG dapat digunakan untuk menyelesaikan masalah yang kompleks dengan banyak variabel. Variabel tersebut bisa kontinyu, diskrit atau campuran keduanya.
e. AG menggunakan kromosom untuk mengkodekan solusi sehingga bisa melakukan pencarian tanpa memperhatikan informasi derivatif yang spesifik dari masalah yang diselesaikan.
f. AG bisa diimplementasikan pada berbagai macam data seperti data yang dibangkitkan secara numerik atau menggunakan fungsi analitis.
g. AG cukup fleksibel untuk dihibridisasikan dengan algoritma lainnya. Beberapa penelitian membuktikan bahwa hybrid AG sangat efektif untuk menghasilkan solusi yang lebih baik.

2.2.2 Komponen – komponen Utama Algoritma Genetika

Terdapat 6 komponen utama di dalam algoritma genetika menurut (Kusumadewi, 2003) dalam (Faris, 2013) :

2.2.2.1 Teknik Pengkodean

Menurut Desiani dalam (Faris, 2013) pengkodean adalah suatu teknik untuk menyatakan populasi awal sebagai calon solusi suatu masalah kedalam suatu kromosom sebagai suatu kunci pokok persoalan ketika menggunakan algoritma genetika (Faris, 2013). Teknik pengkodean meliputi pengkodean gen dan kromosom. Gen merupakan bagian dari kromosom yang dapat direpresentasikan dalam bentuk string bit, array bilangan real, daftar aturan, elemen permutasi, elemen program, atau representasi lain yang dapat diimplementasikan untuk operator genetika.

2.2.2.2 Prosedur Inisialisasi

Prosedur inisialisasi atau pembangkitan populasi awal sejumlah individu dilakukan secara acak atau melalui prosedur tertentu. Ukuran populasi tergantung pada masalah yang akan dipecahkan dan jenis operator genetika yang akan diimplementasikan. Setelah ukuran populasi ditentukan kemudian dilakukan inisialisasi terhadap kromosom yang terdapat pada populasi tersebut. Inisalisasi kromosom dilakukan secara acak dengan memperhatikan domain solusi dan kendala permasalahan yang ada.

2.2.2.3 Fungsi Evaluasi

Suatu individu dievaluasi berdasarkan suatu fungsi tertentu sebagai ukuran performansinya. Dalam algoritma genetika, individu yang memiliki nilai *fitness* tinggi pada kromosomnya akan dipertahankan, sedangkan individu yang memiliki nilai *fitness* rendah pada kromosomnya akan diganti. Fungsi *fitness* tergantung pada permasalahan tertentu dari representasi yang digunakan.

Fungsi *fitness* dapat ditunjukkan pada persamaan 2.1 berikut (Rohmansyah, 2014):

$$Fitness = \frac{1}{1+penalti} \quad \text{.....(2.1)}$$

Dari persamaan 2.1 nilai *fitness* ditentukan oleh penalti yang menunjukkan jumlah pelanggaran kendala pada suatu kromosom. Nilai penalti berbanding terbalik dengan nilai *fitness*, semakin kecil nilai penalti (jumlah pelanggaran) maka semakin besar nilai *fitness*nya seperti ditunjukkan pada persamaan 2.2 berikut :

$$Fitness = \frac{1}{1+\Sigma Bp+\Sigma Np} \quad \text{.....(2.2)}$$

Keterangan :

Bp = Bobot pelanggaran

Np = Indikator pelnggaran

2.2.2.4 Seleksi

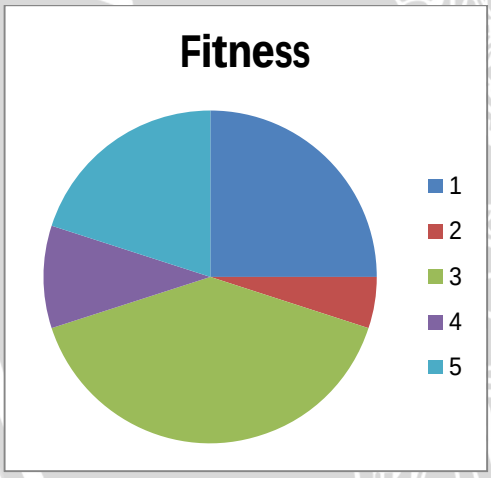
Setiap kromosom yang terdapat dalam populasi akan melalui proses seleksi untuk dipilih menjadi induk (*parent*) yang akan digunakan untuk proses persilangan *(crossover)* dan mutasi. Induk yang baik diharapkan menghasilkan keturunan yang baik. Terdapat beberapa metode seleksi yaitu (Sharma, 2013) :

a. Seleksi Elite (elitism)

Metode Elite yaitu pemilihan kromosom yang memiliki *fitness* yang paling baik di dalam satu populasi. Pada pemilihan kromosom dalam suatu populasi yang terdiri dari 10 kromosom, pemilhan kromosom itu tidak mengurangi jumlah kromosom dalam satu populasi, melainkan metode elite ini berfungsi mirip dengan penggandaan dan kemudian menyimpan hasil penggandaan itu. Kromosom yang terpilih dalam metode elite ini tidak melewati urutan proses seperti seleksi, *crossover*, dan mutasi, tetapi kromosom yang terpilih akan menggantikan secara langsung kromosom pada generasi selanjutnya, yang memiliki nilai *fitness* paling kecil (Sharma, 2013).

b. Seleksi Roulette Wheel

Seleksi Roulette merupakan salah satu teknik seleksi tradisional algoritma genetika. Roulette wheel adalah teknik pilihan yang paling sederhana. Dalam teknik ini, semua kromosom dalam populasi ditempatkan pada roda roulette sesuai dengan nilai *fitness* mereka. Setiap individu diberikan sebuah segmen roda roulette yang ukurannya sebanding dengan nilai *fitness* individu. Semakin besar nilai *fitness*, maka semakin besar segmen. Kemudian, roda roulette maya diputar. Individu yang sesuai dengan segmen roda roulette berhenti kemudian dipilih. Proses ini diulang sampai jumlah yang diinginkan dari individu yang dipilih. Individu dengan *fitness* yang lebih tinggi memiliki lebih banyak probabilitas seleksi. Seleksi ini dapat melewatkan individu-individu terbaik dari populasi pada waktu tertentu. Tidak ada jaminan bahwa individu yang baik akan menemukan jalan mereka ke generasi berikutnya (Sharma, 2013).



Populasi	*Fitness*
1	25.0
2	5.0
3	40.0
4	10.0
5	20.0

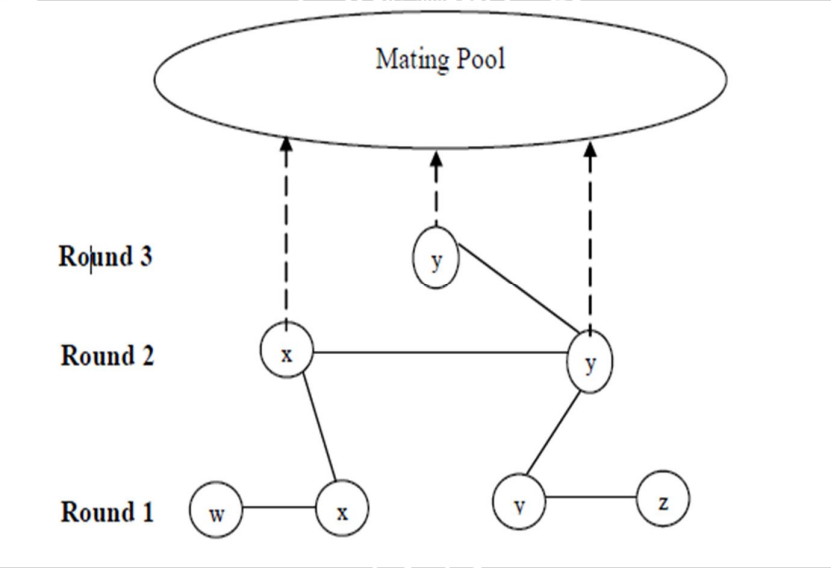
Gambar 2. 1 Seleksi *roulette wheel*

Sumber : (Sharma, 2013)

c. Seleksi *Tournament*

Algoritma genetika menggunakan mekanisme seleksi untuk memilih individu-individu dari populasi untuk memasukkan ke dalam *mating pool.* Individu dari *mating pool* yang digunakan untuk menghasilkan keturunan yang baru, dengan keturunan yang dihasilkan membentuk dasar dari generasi berikutnya. Mekanisme seleksi tournament hanyalah sebuah proses yang mendukung pemilihan individu yang lebih baik dalam populasi *mating pool.*

Seleksi ini mendorong untuk meningkatkan niali *fitness* generasi bru. Tingkat konvergensi dari AG sangat ditentukan oleh tekanan seleksi, dengan tekanan seleksi yang lebih tinggi sehingga tingkat konvergensi yang lebih tinggi. Namun, jika tekanan seleksi terlalu rendah, tingkat konvergensi akan lambat, dan algoritma genetika akan memakan waktu lebih lama untuk menemukan solusi optimal. Jika tekanan seleksi terlalu tinggi, peningkatan algoritma genetika prematur konvergen ke solusi yang salah (suboptimal). Seleksi turnamen memberikan tekanan seleksi dengan mengadakan turnamen di antara pesaing, dengan *s* menjadi ukuran turnamen. Pemenang turnamen adalah individu dengan *fitness* tertinggi dari pesaing *s* turnamen. Pemenangnya kemudian dimasukkan ke dalam *mating pool*. *Mating pool*, yang terdiri dari pemenang turnamen, memiliki *fitness* rata-rata lebih tinggi daripada rata-rata *fitness* populasi lainnya (Sharma, 2013).



Gambar 2. 2 Seleksi *tournament*

Sumber : (Sharma, 2013)

Pemenang kompetisi selalu memiliki kromosom paling kuat (niali *fitness* paling tinggi) dalam populasi. Semakin sukses kromosom dalam kompetisi maka semakin sering muncul dalam *mating pool*.

2.2.2.5 Operator Genetika

Algoritma genetika merupakan proses pencarian heuristik dan acak sehingga penekana pemilihan operator yang digunakan sangat menentukan keberhasilan algoritma genetika dlam menemukan solusi permasalahan yang diberikan. Hal yang harus diperhatikan adalah konvergensi prematur, yaitu pencapaian solusi optimum sebelum waktunya, solusi yang diperoleh adalah hasil optimum lokal. Terdapat dua operator genetika yaitu :

a. Persilangan *(Crossover)*

Salah satu komponen yang paling penting dalam algoritma genetika adalah persilangan (*crossover*). Persilangan merupakan proses pada algoritma genetika yang bekerja untuk menggabungkan dua kromosom induk (*parent*) menjaadi kromosom baru (*offspring*) pada suatu waktu. Terdapat beberapa jenis *crossover*, antara lain :

1. *One- cut- point*

   Metode one-cut-point, yang secara acak memilih satu titik potong dan menukarkan bagian kanan dari tiap induk untuk menghasilkan *offspring*.

   Misalkan yang terpilih sebagai induk adalah P1 dan P2. Titik potong (cut point) adalah titik 2. Maka akan dihasilkan dua *offspring* (C1 dan C2) sebagai berikut :

   titik potong ↓

   P1 : [00 01110000010001010101011000101110 1]

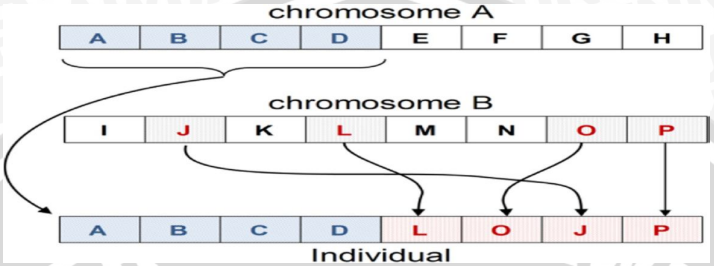
   P2 : [10 01111011110111111111101101000111 11]

   C1 : [00 01111011110111111111101101000111 11]

   C2 : [10 01110000010001010101011000101110 1]

2. *Order Crossover*

Titik potong dipilih secara acak dari induk. Sisi kiri *offspring* berasal dari titik potong dari induk pertama dan sisanya diambil secara random dari induk kedua(Faraji, 2014). Sebgaimana pada Gambar 2.3 berikut :



Gambar 2. 3 *Order crossover*

Sumber : (Faraji, 2014)

3. *Extended Intermediate Crossover*

Crossover dilakukan dengan memilih dua induk (parent) secara acak dari populasi. Metode *crossover* extended intermediate *crossover* menghasilkan *offspring* dari kombinasi nilai dua induk. Misalkan P1 dan P2 adalah dua kromosom yang telah diseleksi untuk melakukan *crossover*, maka *offspring* C1 dan C2 bisa dibangkitkan sebagai berikut:

C1 = P1 + a (P2 – P1)

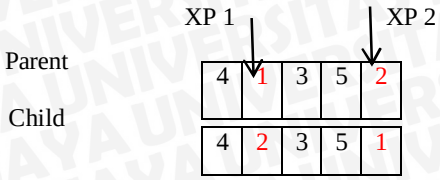
C2 = P2 + a (P1 – P2)

a dipilih secara acak pada interval tertentu (Mahmudy , 2013).

b. Mutasi

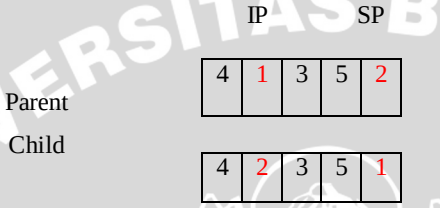
Mutasi diperlukan untuk mengembalikan informasi bit yang hilang akibat *crossover*. Mutasi diterapkan dengan probabilitas yang sangat kecil. Mutasi pada tingkat kromosom yaitu semua gen dalam kromosom berubah. Terdapat 3 metode mutasi diantaranya yaitu :

1. Metode mutasi yang paling sederhana adalah *reciprocal exchange mutation*. Metode ini bekerja dengan memilih dua posisi (exchange point / XP) secara random kemudian menukarkan nilai pada posisi tersebut seperti pada Gambar 2.4 berikut :



Gambar 2. 4 *Reciprocal exchange mutation*

2. *Insertion mutation* bekerja dengan memilih satu posisi (selected point / SP) secara random kemudian mengambil dan menyisipkan nilainya pada posisi lain (insertion point / IP) secara random seperti pada Gambar 2.5:



Gambar 2. 5 *Insertion mutation*

2.2.2.6 Parameter Kontrol

Parameter kontrol genetika diperlukan untuk mengendalikan operator – oprator seleksi. Terdapat dua parameter dasar dari algoritma genetika yaitu*crossover rate* (Cr) dan *mutation rate* (Mr) (Desiani, 2006) dalam (Faris, 2013) :

1. *Crossover Rate* (Cr)

Crossover rate akan mengendalikan operator *crossover* setiap generasi dalam populasi yang mengalami *crossover*. Semakin besar nilai *crossover rate*, akan semakin cepat struktur individu baru terbentuk ke dalam populasi. Akan tetapi, bila nilai *crossover* terlalu besar, indvidu yang merupakan kandidat solusi terbaik dapat hilang lebih cepat pada generasi selanjutnya.

2. *Mutation Rate* (Mr)

Mutation rate akan mengendalikan operator mutasi pada setiap generasi dimana peluang mutasi yang digunakan lebih kecil dari peluang *crossover*. Pada seleksi alami murni, mutasi jarang sekali muncul sehingga pada algoritma genetika juga tidak selalu terjadi. Untuk itulah nilai peluang mutasi dibuat lebih kecil untuk setiap generasi.

2.2.3 Struktur Algoritma Genetika

Proses dalam algoritma genetika diawali dengan **inisialisasi**, yaitu menciptakan individu-individu secara acak yang memiliki susunan gen (kromosom) tertentu. Kromosom ini mewakili solusi dari permasalahan yang akan dipecahkan. **Reproduksi** dilakukan untuk menghasilkan keturunan (*offspring*) dari individu-individu yang ada di populasi. **Evaluasi** digunakan untuk menghitung kebugaran (*fitness*) setiap kromosom. Semakin besar *fitness* maka semakin baik kromosom tersebut untuk dijadikan calon solusi. Seleksi dilakukan untuk memilih individu dari himpunan populasi dan *offspring* yang dipertahankan hidup pada generasi berikutnya. Fungsi probabilistis digunakan untuk memilih individu yang dipertahankan hidup. Individu yang lebih baik (mempunyai nilai kebugaran/*fitness* lebih besar) mempunyai peluang lebih besar untuk terpilih (Mahmudy, 2013).

Setelah melewati sekian iterasi (generasi) akan didapatkan individu terbaik. Individu terbaik ini mempunyai susunan kromosom yang bisa dikonversi menjadi solusi yang terbaik (paling tidak mendekati optimum). Dari sini bisa disimpulkan bahwa algoritma genetika menghasilkan suatu solusi optimum dengan melakukan pencarian di antara sejumlah alternatif titik optimum berdasarkan fungsi probabilistic (Mahmudy, 2013).

2.3 Penjadwalan Perawat

Penjadwalan merupakan proses dalam menyusun jadwal atau urutan proses yang diperlukan dalam suatu permasalahan. Contoh permasalahan penjadwalan diantaranya adalah penjadwalan pelajaran, penjadwalan ujian, penjadwalan karyawan serta penjadwalan perawat. Penjadwalan daftar jaga perawat merupakan sebuah proses penjadwalan yang mengatur waktu kerja perawat di sebuah rumah sakit. Dalam proses penjadwalan daftar jaga perawat ada beberapa komponen yang digunakan yaitu perawat, ruang, hari, dan shift (Tjajo, 2008).

Banyaknya jumlah pasien yang membutuhkan pelayanan kesehatan sangat kontras dengan jumlah perawat dan dokter yang ada pada rumah sakit. Hal ini

mengakibatkan pihak rumah sakit perlu melakukan pengaturan jadwal yang efisien untuk setiap sumber daya manusia yang ada (termasuk perawat dan pasien) agar semua pasien dapat terlayani dengan baik (Atmasari, 2010).

Terdapat Karakteristik Penjadwalan perawat menurut Warner (1976) seperti yang dikutip oleh Atmasari (2010) penjadwalan perawat memiliki karakteristik yang penting, antara lain:

a. Coverage

Coverage merupakan istilah untuk menunjukkan jumlah perawat dengan berbagai tingkat yang akan ditugaskan sesuai jadwal berkenaan dengan pemakaian minimum personel perawat tersebut.

b. Quality

Quality atau kualitas merupakan sebuah alat untuk menilai keadaan pola jadwal.

c. Stability

Stability atau stabilitas dapat dicapai ketika seseorang perawat mengetahui kepastian jadwal libur masuk untuk beberapa hari mendatang dan supaya mereka mempunyai pandangan bahwa jadwal ditetapkan oleh suatu kebijaksanaan yang stabil dan konsisten, seperti weekend policy, rotation policy.

d. Flexibility

Flexibility merupakan kemampuan jadwal untuk mengantisipasi setiap perubahan-perubahan seperti pembagian fulltime, part time, rotasi shift dan permanen shift.

e. Fairness

Fairness merupakan suatu pernyataan bahwa tiap-tiap perawat akan merasa diberlakukan sama.

f. Cost

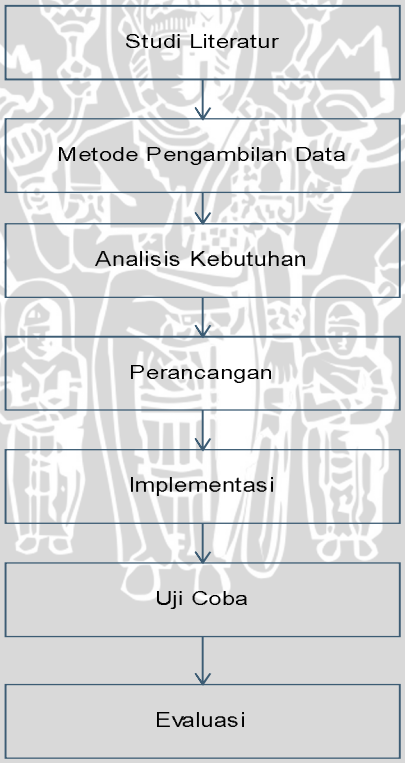
Cost atau biaya merupakan jumlah resource yang dikonsumsi untuk penyusunan maupun operasional penjadwalan.

BAB III
METODOLOGI PENELITIAN

Pada bab ini dibahas tahapan penelitian, kebutuhan sistem, formulasi permasalahan serta siklus penyelesaian permasalahan menggunakan algoritma genetika.

3.1 Tahapan Penelitian

Pada sub bab metode penelitian, akan dibahas metode yang digunakan dalam penjadwalan perawat serta langkah- langkah dalam mengimplementasikan metode pada penjadwalan perawat. Langkah penelitian digambarkan melalui diagram pada Gambar 3.1:



Gambar 3. 1 Diagram Alir Penelitian

Sumber : Perancangan

3.1.1 Studi Literatur

Langkah pertama pada penelitian ini adalah mempelajari metode yang digunakan. Langkah ini dilakukan untuk mendapatkan dasar teori dan sumber untuk merancang penjadwalan perawat menggunakan algoritma genetika. Informasi dan pustaka yang berkaitan dengan skripsi ini didapat dari buku, situs internet, penjelasan dari dosen pembimbing, dan rekan – rekan mahasiswa serta referensi lain yang dapat digunakan untuk menyelesaikan skripsi ini. Teori yang dipelajari yaitu penjadwalan perawat serta metode algoritma genetika untuk menyelesaikan kasus penjadwalan perawat.

3.1.2 Metode Pengambilan Data

Metode yang digunakan dalam penjadwalan perawat pada skripsi ini adalah algoritma genetika. Variabel penelitian skripsi ini adalah bagaimana penyusunan jadwal jaga perawat yang sesuai pada aturan yang terdapat pada instansi. Hipotesis dari penelitian ini adalah algoritma genetika bisa diimplementasikan pada penjadwalan perawat yang sebelumnya masih dilakukan secara manual dan menyebabkan ketidakadilan pada pembagian jadwal yang berakibat pada kurang optimalnya kinerja perawat.

Berdasarkan cara pengumpulan data untuk kegiatan penelitian terdapat 2 jenis data yaitu data sekunder dan data primer. Data sekunder adalah data yang telah dikumpulkan oleh orang lain dan tidak dipersiapkan untuk kegiatan peneitian, tetapi dapat digunakan untuk tujuan penelitian. Data primer adalah data yang didapatkan langsung dari responden penelitian. Metode pengumpulan data primer yang bersifat kuantitatif dapat menggunakan instrument kuesioner dan wawancara.

Data yang digunakan pada penelitian ini yaitu data sekunder untuk mendapatkan jadwal jaga perawat yang akan diproses dengan algoritma genetika. Data primer untuk mengetahui kriteria serta batasan jadwal jaga perawat yang sesuai dengan aturan pada instansi. Data primer dan sekunder didapatkan dari metode wawancara dan survei secara langsung pada instansi.

3.1.3 Analisis Kebutuhan

Analisis kebutuhan bertujuan untuk menganalisis dan mendapatkan kebutuhan yang diperlukan dalam sistem penjadwalan perawat.

3.1.4 Perancangan

Perancangan implementasi metode bertujuan untuk mengetahui komponen apa saja yang dibutuhkan dalam mengimplementasikan algoritma genetika untuk penjadwalan perawat.

3.1.5 Implementasi

Implementasi metode penjadwalan perawat dilakukan sesuai perancangan yang telah dibuat.

3.1.6 Uji Coba

Pengujian metode dilakukan setelah proses implementasi selesai dilakukan. Langkah ini bertujuan untuk mengetahui seberapa baik metode yang digunakan dalam menyelesaikan permasalahan penjadwalan perawat.

3.1.7 Evaluasi

Berdasarkan pengujian metode yang digunakan, langkah terakhir dalam penelitian ini adalah melakukan evaluasi hasil uji coba metode. Langkah ini bertujuan untuk mengetahui nilai parameter metode yang terbaik dalam penjadwalan perawat.

3.2 Kebutuhan Sistem

Pada sub bab kebutuhan sistem akan dibahas deskripsi umum sistem, batasan implementasi sistem serta spesifikasi kebutuhan sistem.

3.2.1 Deskripsi Umum Sistem

Sistem ini dibangun untuk penjadwalan perawat menggunakan metode algoritma genetika. Data masukan dalam sistem ini merupakan data jadwal jaga perawat yang meliputi perawat, ruang dan hari. Data masukan akan dikodekan sebagai populasi awal calon solusi jadwal jaga perawat. Representasi algoritma genetika dalam penjadwalan perawat ditunjukkan dalam Tabel 3.1:

Tabel 3. 1 Representasi algoritma genetika dalam jadwal jaga perawat

Populasi	Kumpulan sejumlah solusi jadwal
Kromosom	1 jadwal yang terdiri dari perawat dalam shift dan tanggal tertentu
Gen	Atribut dari jadwal yaitu perawat, tanggal serta shift
Fitness	Nilai yang menunjukkan kualitas suatu jadwal.

Populasi solusi jadwal akan dibentuk secara random setelah kromosom dikodekan. Sejumlah solusi jadwal yang telah terbentuk akan dievaluasi nilai *fitnees* nya. Evaluasi nilai *fitness* pada setiap solusi jadwal akan menunjukkan kualitas jadwal. Semakin tinggi nilai *fitness* pada suatu solusi jadwal, maka jadwal tersebut memiliki kualitas yang semakin baik. Kualitas jadwal yang baik yaitu jadwal yang tidak melanggar *constraint* yang telah ditentukan.

Soft constraint dalam penelitian ini terdapat 2 macam. *Soft constraint* pertama yaitu perawat tidak dapat berjaga pada shift yang berurutan, misalnya berjaga pada shift pagi setelah berjaga pada shift malam sebelumnya. *Soft constraint* kedua yaitu perawat berjaga pada shift malam dengan pembagian tertentu agar proporsional dan adil. *Hard constraint* pada penelitian ini terdapat 2 macam. *Hard constraint* pertama adalah 1 orang perawat tidak boleh muncul 2x dalam 1 shift. *Hard Constraint* kedua adalah seorang perawat tidak boleh mendapat jadwal jaga pada saat sedang mengajukan cuti. Tahapan sistem yang akan dibangun menggunakan algoritma genetika adalah sebagai berikut :

1. Representasi Kromosom
2. Evaluasi dengan menghitung *fitness* dari masing-masing kromosom
3. *Crossover* untuk mendapatkan individu baru

4. Proses mutasi untuk meningkatkan variasi populasi
5. Proses seleksi untuk membentuk populasi baru

3.2.2 Batasan Implementasi Sistem

Batasan implementasi sistem adalah sebagai berikut :

1. Data yang digunakan adalah data jadwal jaga perawat pada RSUD Ibnu Sina Gresik bagian ICU.
2. Metode seleksi yang digunakan adalah *elitsm.*
3. Metode *crossover* yang digunakan adalah *one cut point crossover.*

3.2.3 Spesifikasi Kebutuhan Sistem

Spesifikasi kebutuhan perangkat yang digunakan dalam pembuatan sistem ini meliputi:

a. Spesifikasi Kebutuhan *Hardware*
   - ✓ Komputer Laptop dengan sistem operasi Windows 7 Home Premium. Prosesor Intel (R) Core i5-2430 M CPU @ 2.40 GHz. RAM 4,00 GB 64-bit Operating System

b. Spesifikasi Kebutuhan Software
   - ✓ Sistem operasi menggunakan Windows (Windows 7 , Windows 8).
   - ✓ Bahasa pemrograman Java.
   - ✓ Database untuk pengolahan data menggunakan database management system (DBMS) MySQL.

3.3 Formulasi Permasalahan

Algoritma genetika tidak beroperasi dengan penyelesaian asli dari suatu masalah tetapi beroperasi dengan penyelesaian yang telah direpresentasikan. Representasi kromosom merupakan proses pengkodean dari penyelesaian asli suatu permasalahan. Pengkodean kromosom yang diperlukan pada penelitian ini adalah pengkodean perawat dan shift setiap hari dalam satu bulan.

Jumlah perawat dalam penelitian ini berjumlah 16 yaitu yang terdapat pada ruang ICU. Tabel 3.2 berikut menunjukkan data ruang, jumlah perawat serta kebutuhan perawat dalam 1 shift.

Tabel 3. 2 Data ruang dan perawat

No	Ruang	Jumlah Perawat	Jumlah Perawat Dalam 1 Shift
1	Anggrek	12	4
2	Cempaka	16	4
3	Dahlia	12	4
4	Flamboyan	16	4
5	Gardena	12	4
6	Helicona	12	4
7	Wijaya Kusuma	12	4
8	Bersalin	20	5
9	ICU	16	4
10	NICU	16	4
11	IGD	20	5
	Jumlah	164	46

Data perawat yang bertugas pada ruang ICU ditunjukkan pada Tabel 3.3 berikut :

Tabel 3.3 Data perawat

No.	Nama	Ruang
1	M. Yusuf	Icu
2	Dewi Masruroh	Icu
3	Hijrah S	Icu
4	Husni H	Icu
5	Imam S	Icu
6	Sumiah	Icu
7	Esti A	Icu
8	Iin S	Icu
9	Firdaus Z	Icu
10	Ulifah	Icu
11	Novi I	Icu
12	Izzatur R	Icu
13	Kiswanto	Icu
14	Rumiati	Icu
15	Paramitha S	Icu
16	Santi Q	Icu

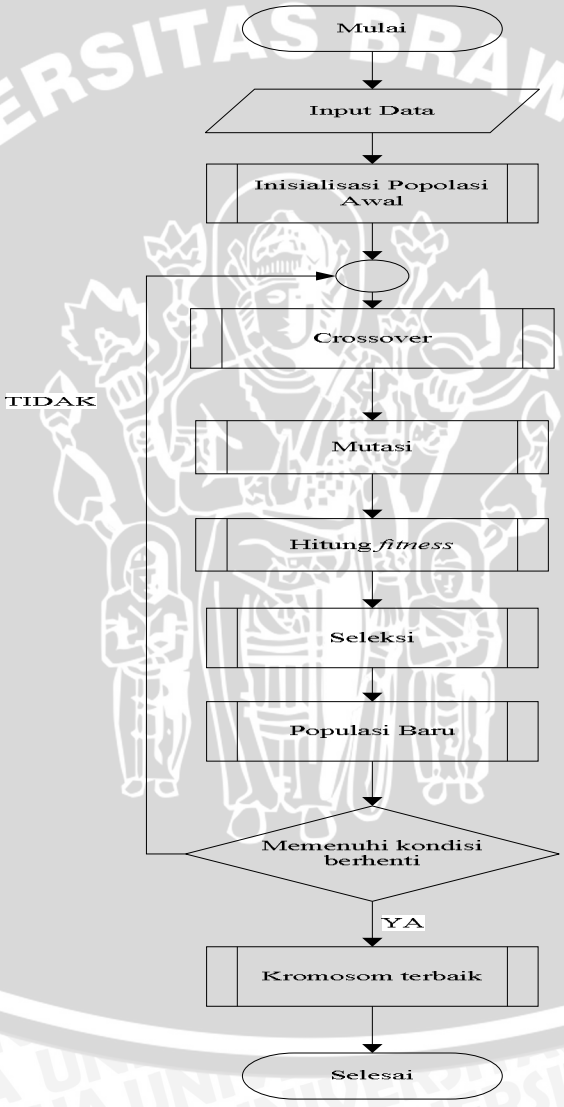
Jumlah shift dalam penelitian ini berjumlah 3. Data waktu shift ditunjukkan pada Tabel 3.4.

Tabel 3. 4 Data shift

No.	Shift	Waktu
1	Pagi	07.00 S/D 14.00
2	Sore	14.00 S/D 20.00
3	Malam	20.00 S/D 07.00

3.4 Siklus Penyelesaian Masalah Menggunakan Algoritma Genetika

Proses penyelesaian masalah penjadwalan perawat menggunakan algoritma genetika dapat dilihat pada flowchart berikut :



Gambar 3. 2 Flowchart Proses Implementasi

Sumber : Perancangan

1. Inisialisasi parameter awal

   a. Data penjadwalan perawat yang berupa jumlah perawat, shift, tanggal, nama perawat, ruang.

   b. Parameter Algoritma genetika, meliputi :

   - Jumlah generasi
   - Ukuran Populasi (*popsize*)
   - *Crossover rate* (Cr)
   - *Mutation rate* (Mr)

2. Bangkitkan populasi awal sebanyak jumlah populasi yang ditentukan.

3. Membentuk populasi baru dengan langkah-langkah sebagai berikut :

   - Melakukan proses *crossover* pada induk yang terpilih berdasarkan Cr untuk mendapatkan anak (*offspring*) dengan metode one cut-point *crossover*.
   - Melakukan proses mutasi pada induk yang terpilih berdasarkan Mr untuk mendapatkan anak (*offspring*) dengan metode *reciprocal exchange mutation.*
   - Menghitung nilai *fitness* untuk masing-masing kromosom atau individu
   - Melakukan seleksi elitism untuk memilih individu sebanyak jumlah populasi awal dari gabungan individu induk dan anak untuk dijadikan populasi pada generasi selanjutnya.

4. Jika kondisi akhir terpenuhi, maka iterasi berhenti dan solusi terbaik adalah populasi yang terpilih pada generasi tersebut, jika kondisi akhir tidak terpenuhi maka lanjut ke-iterasi generasi selanjutnya.

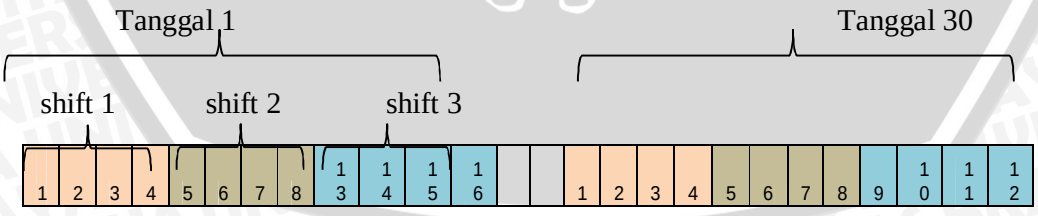
Pada sub bab 3.3.1 – 3.3.4 akan dibahas contoh perhitungan manual. Misalkan dilakukan proses penjadwalan terhadap 16 perawat yang berjaga pada ruang ICU. Jadwal dibuat untuk 30 hari, setiap hari terdapat 3 shift yaitu pagi, sore dan malam. Jumlah popoulasi (*Popsize*) yang digunakan sebanyak 10 dengan *crossover rate* (Cr) 0,2 dan *mutation rate* (Mr) 0,1. Iterasi dilakukan sebanyak 2 kali.

3.4.1 Representasi Kromosom dan Perhitungan *Fitness*

Representasi permutasi berbasis integer digunakan dalam penelitian ini. Sesuai kebutuhan perawat tiap shift pada bagian ICU dalam satu bulan, maka dibutuhkan 4 perawat tiap shift sehingga kebutuhan perawat tiap hari adalah :

$jumlah\ shift\ x\ jumlah\ kebutuhan\ perawat\ dalam\ shift = 3\ x\ 4 =$ 12 orang perawat. Selama satu bulan (akumulasi 30 hari) maka kromosom penyusun algoritma genetika sebanyak 12 x 30 = 360 gen. Gen dalam kromosom direpresentasikan dengan angka integer yang menunjukkan nomor perawat. Dalam ruang ICU terdapat 16 orang perawat sehingga angka integer penyusun gen yaitu 1- 16. Dalam periode 30 hari setiap perawat mendapat jadwal jaga sebanyak $\frac{360}{16} = 22{,}5 \sim 23\ kali$.

Setiap 4 kolom gen dalam kromosom menunjukkan shift , sehingga 4 kolom pertama menunjukkan shift 1, 4 kolom kedua menunjukkan shift 2, 4 kolom ketiga menunjukkan shift 3 sehingga 12 kolom menunjukkan kebutuhan perawat dalam satu hari seterusnya sampai hari ke 30 dan berjumlah 360 (Wong, 2014). Representasi kromosom digambarkan pada Gambar 3.3 berikut.



Gambar 3.3 Representasi Kromosom

Berikut Tabel 3.5 merupakan hasil konversi dari kromosom pada penelitian ini :

Tabel 3. 5 Hasil konversi dari kromosom ke jadwal

TGL	REPRESENTASI KROMOSOM											
	SHIFT 1				SHIFT 2				SHIFT 3			
1	1	2	3	4	5	6	7	8	13	14	15	16
2	1	2	3	4	5	6	7	8	13	14	15	16
3	1	2	3	4	9	10	11	12	13	14	15	16
4	5	6	7	8	1	2	3	4	9	10	11	12
5	5	6	7	8	1	2	3	4	9	10	11	12
6	5	6	7	8	1	2	3	4	13	14	15	16
7	9	10	11	12	5	6	7	8	1	2	3	4
8	9	10	11	12	13	14	15	16	1	2	3	4
9	9	10	11	12	13	14	15	16	1	2	3	4
10	13	14	15	16	9	10	11	12	5	6	7	8
11	13	14	15	16	9	10	11	12	5	6	7	8
12	9	10	11	12	13	14	15	16	5	6	7	8
13	9	10	12	11	13	15	14	16	5	6	7	8
14	13	15	16	14	11	10	9	12	3	2	1	4
15	14	13	16	15	9	11	12	10	4	2	3	1
16	1	2	3	4	13	14	15	16	5	6	7	8
17	2	3	1	4	15	14	13	16	7	6	5	8
18	1	2	3	4	14	15	16	13	10	9	11	12
19	8	6	7	5	3	2	1	4	14	16	13	15
20	5	6	7	8	1	2	3	4	13	14	15	16
21	5	6	7	8	1	2	3	4	13	14	15	16
22	9	10	11	12	5	6	7	8	1	2	3	4
23	9	10	11	12	5	6	7	8	1	2	3	4
24	9	10	11	12	5	6	7	8	1	2	3	4
25	13	14	15	16	9	10	11	12	5	6	7	8
26	13	14	15	16	9	10	11	12	5	6	7	8
27	13	14	15	16	1	2	3	4	5	6	7	8
28	5	6	7	8	1	2	3	4	9	10	11	12
29	13	14	15	16	1	2	3	4	9	10	11	12
30	1	2	3	4	5	6	7	8	9	10	11	12

Ket :

Tanda blok kuning : pelanggaran *soft constraint*

Setiap kromosom dalam satu individu akan dievaluasi nilai *fitness* nya. *Fitness* dinilai berdasarkan jumlah pelanggaran yang terjadi. Rumus perhitungan nilai *fitness* pada penelitian ini sesuai dengan persamaan berikut (Rohmansyah, 2014) :

$$Fitness = \frac{100}{1+ (5\Sigma P1 + 5\Sigma P2 + 20\Sigma P3 + 50\Sigma P4)} \quad (3\text{-}1)$$

Keterangan :

P1 = Pelanggaran nomor 1

P2 = Pelanggaran nomor 2

P3 = Pelanggaran nomor 3

P4 = Pelanggaran nomor 4

Nilai 100 dipilih untuk memudahkan dalam proses evaluasi. Konstanta pelanggaran dibedakan berdasarkan jenis pelanggaran. Konstanta pelanggaran *soft constraint* memiliki nilai lebih kecil dibandingkan dengan konstanta pelanggaran *hard constraint*. Pelanggaran P4 merupakan penalti cuti yang memiliki urgensi terbesar sehingga diberi nilai konstanta 50. Nilai pelanggaran dihitung berdasaran kemunculan pelanggaran pada kromosom. Setiap kemunculan dihitung 1 pelanggaran. Jenis pelanggaran beserta nilai pelanggaran dalam penelitian ini dapat dilihat pada Tabel 3.6 berikut.

Tabel 3. 6 Jenis pelanggaran

No. Pelanggaran	Keterangan	Jenis Pelanggaran	Konstanta Pelanggaran	Nilai Pelanggaran
P1	Perawat tidak dapat berjaga pada shift yang berurutan, misalnya berjaga pada shift pagi setelah berjaga pada shift malam sebelumnya	*soft constraint.*	5	1
P2	Seorang perawat berjaga pada shift malam sesuai dengan porsi yang telah ditentukan	*soft constraint.*	5	1
P3	Seorang perawat tidak boleh muncul 2x atau lebih dalam 1 shift	*hard constraint.*	20	1
P4	Seorang perawat tidak boleh mendapat jadwal jaga pada saat sedang mengajukan cuti	*hard constraint.*	50	1

Soft constraint pelanggaran 2 dibuat agar jadwal jaga perawat pada shift malam dapat dibagi secara adil. Pembagian dilakukan dengan perhitungan membagi jumlah perawat jaga dalam 1 bulan dengan jumlah shift. Kebutuhan perawat jaga pada shift malam = $\frac{jumlah\ jadwal\ jaga\ perawat\ (30\ hari)}{jumlah\ shift} = \frac{23}{3} = 7{,}67 \sim 8$ kali. Pelanggaran terjadi saat perawat mendapat jadwal jaga shift malam melebihi 8 kali dalam 1 bulan. Pada perhitungan nilai *fitness* individu P2 dimisalkan perawat dengan id 5 melakukan cuti pada tanggal 11.

TGL	REPRESENTASI KROMOSOM											
	SHIFT 1				SHIFT 2				SHIFT 3			
1	1	2	3	4	5	6	7	8	13	14	15	16
2	1	2	3	4	5	6	7	8	13	14	15	16
3	1	2	3	4	9	10	11	12	13	14	15	16
4	5	6	7	8	1	2	3	4	9	10	11	12
5	5	6	7	8	1	2	3	4	9	10	11	12
6	5	6	7	8	1	2	3	4	13	14	15	16
7	9	10	9	12	5	6	7	8	1	2	3	4
8	9	10	11	12	13	14	15	16	1	2	3	4
9	9	10	11	12	13	14	15	16	1	2	3	4
10	13	14	15	16	9	10	11	12	5	6	7	8
11	13	14	15	16	9	10	11	12	5	6	7	8
12	9	10	11	12	13	14	15	16	5	6	7	8
13	9	10	12	11	13	15	14	16	5	6	7	8
14	1	2	3	2	14	15	16	13	10	9	11	12
15	8	6	7	5	3	2	1	4	14	16	13	15
dst	5	6	7	8	1	2	3	4	13	14	15	16
24	9	10	11	12	5	6	7	8	1	2	3	4
25	13	14	15	16	9	10	11	12	5	6	7	10
26	13	14	15	16	9	10	11	12	13	14	15	16
27	13	14	15	16	1	2	3	4	5	6	7	8
28	5	6	7	8	1	2	3	4	9	10	11	12
29	13	14	15	16	1	2	3	4	9	10	11	12
30	1	2	3	4	5	6	7	8	9	10	11	12

Ket :

Tanda blok kuning : pelanggaran *soft constraint*

Tanda blok merah : pelanggaran *hard constraint*

Pada individu P2 ditemukan :

- P1 (*soft constraint*) sebanyak 8
- P2 (*soft constraint*) sebanyak 4
- P3 (*hard constraint*) sebanyak 2
- P4 (*hard constraint*) sebanyak 1

Maka perhitungaan *fitness* nya adalah :

$$Fitness = \frac{100}{1+ (5\Sigma P1 + 5\Sigma P2 + 20\Sigma P3 + 50\Sigma P4)}$$

$$Fitness = \frac{100}{1+ (5.8 + 5.4 + 20.2 + 50.1)}$$

$$Fitness = \frac{100}{1+ 150}$$

$$Fitness = \frac{100}{151}$$

$$Fitness = 0{,}66225$$

3.4.2 Inisialisasi Populasi Awal

Pembentukan populasi awal dilakukan untuk mencari penyelesaian yang optimal. Dalam penelitian ini representasi populasi awal merupakan solusi jadwal yang dibuat secara random. Berikut merupakan contoh populasi awal ditunjukkan pada Tabel 3.7 yang dibangkitkan secara random.

Tabel 3. 7 Populasi awal

INDIVIDU 1

TGL	REPRESENTASI KROMOSOM											
	SHIFT 1				SHIFT 2				SHIFT 3			
1	13	14	15	16	9	10	11	12	5	6	7	8
2	13	14	15	16	1	2	3	4	5	6	7	8
3	13	14	15	16	1	2	3	4	9	10	11	12
4	13	14	15	16	1	2	3	4	9	10	11	12
5	1	2	3	4	5	6	7	8	9	10	11	12
Dst	5	6	7	8	1	2	3	4	13	14	15	16
30	9	10	11	12	5	6	7	8	1	2	3	4

..... INDIVIDU N

3.4.3 Reproduksi

Pada proses reproduksi terdapat proses *crossover* dan mutasi. *Crossover* merupakan operasi yang bekerja unrtuk menggabungkan dua kromosom induk menjadi kromosom baru (*offspring*). Jumlah solusi jadwal yang mengalami *crossover* ditentukan oleh parameter yang disebut dengan *crossover rate.* Metode *crossover* yang digunakan dalam penelitian ini adalah one cut point *crossover* (Kaya, 2008). Pada contoh perhitungan manual ini, proses *crossover* dilakukan dengan inisialisai *crossover rate* (Cr) 0,2. Langkah pertama yaitu memilih 2 induk yang akan melalui proses *crossover* dengan random.

Individu terpilih yaitu individu 4 dan 10. Titik *crossover* dipilih secara random dan menghasilkan cut point pada titik 10.

Parent 1	1	2	3	4	5	6	7	8	9	10	11	12	9	10	11	12	1	2	3	4	13	14	15	16

Parent 2	13	14	15	16	9	10	11	12	5	6	7	8	5	6	7	8	1	2	3	4	13	14	15	16

Child 1	1	2	3	4	5	6	7	8	9	10	7	8	5	6	7	8	1	2	3	4	13	14	15	16

Child 2	13	14	15	16	9	10	11	12	5	6	11	12	9	10	11	12	1	2	3	4	13	14	15	16

Setelah *crossover* dilakukan, langkah selanjutnya adalah mutasi. Mutasi merupakan proses pengubahan gen keturunan secara *random.* Dalam penjadwalan perawat, proses mutasi dilakukan dengan metode *reciprocal exchange mutation* yaitu memilih dua posisi (exchange point / XP) perawat dalam suatu shift dan hari tertentu secara random kemudian menukarkan nilai pada posisi tersebut. Pada contoh perhitungan manual ini, proses *crossover* dilakukan dengan inisialisai *mutation rate* (Mr) 0,1. Individu dipilih secara random.

Parent 1	1	2	3	4	5	6	7	8	9	10	11	12	9	10	11	12	1	2	3	4	13	14	15	16
Child 1	1	2	3	4	5	6	7	8	9	4	11	12	9	10	11	12	1	2	3	10	13	14	15	16

3.4.4 Evaluasi dan Seleksi

Proses evaluasi dilakukan dengan melakukan perhitungan *fitness* terhadap setiap individu dalam populasi. Seleksi dilakukan dengan metode *elitsm* yaitu memilih individu dengan nilai *fitness* terbaik. Perhitungan nilai *fitness* dilakukan pada setiap individu yang terdapat dalam populasi. Didapatkan nilai *fitnes* sebanyak 13 yaitu 10 dari individu awal, 2 dari proses *crossover* dan 1 dari proses mutasi. Hasil perhitungan *fitness* dapat dilihat pada Tabel 3.8 berikut :

Tabel 3. 8 Hasil perhitungan *fitness*

Individu	*Fitness*
P1	0,090909
P2	0,02439
P3	0,090909
P4	0,090909
P5	0,02439
P6	0,008264
P7	0,006623
P8	0,090909
P9	0,009901
P10	0,090909
C1	0,090909
C2	0,090909
C3	0,02439

Proses seleksi *elitsm* dilalukan melalui proses sorting. Nilai *fitness* individu terbaik sebanyak jumlah populasi akan dipilih sebagai populasi baru. Pseudo- code seleksi *elitsm* dapat dilihat pada Gambar 3.3 :

```
PROCEDURE ElitismSelection
Input:
POP: himpunan individu pada populasi
pop_size: ukuran populasi
OS: himpunan individu anak (offspring) hasil reproduksi menggunakan crossover and mutasi
Output:
POP: himpunan individu pada populasi setelah proses seleksi selesai
/* gabungkan individu pada POP dan OS ke dalam TEMP */
TEMP ← Merge (POP,OS)
/* urutkan individu berdasarkan fitness secara ascending */
OrderAscending (Temp)
/* copy pop_size individu terbaik ke POP */
POP ← CopyBest (Temp, pop_size)

END PROCEDURE
```

Gambar 3. 3 Pseudo code seleksi *elitsm*

Individu baru yang terpilih dapat dilihat pada Tabel 3.9 berikut :

Tabel 3. 9 Hasil seleksi *elitsm*

Individu	*Fitness*	Asal
P1	0,090909	P1
P2	0,090909	P3
P3	0,090909	P4
P4	0,090909	P8
P5	0,090909	P10
P6	0,090909	C1
P7	0,090909	C2
P8	0,02439	P2
P9	0,02439	P5
P10	0,02439	C3

Iterasi dilanjutkan sampai iterasi ke dua dan menghasilkan solusi jadwal dengan nilai *fitness* pada Tabel 3.10 berikut :

Tabel 3. 10 Hasil seleksi *elitsm*

Individu	*Fitness*	Asal
P1	1	P1
P2	0,090909	P2
P3	0,090909	P3
P4	0,090909	P4
P5	0,090909	P5
P6	0,090909	P6
P7	0,090909	C1
P8	0,032258	P7
P9	0,032258	C2
P10	0,02439	P8

Maka solusi jadwal jaga perawat yang diambil yaitu P1. Jadwal jaga dapat dilihat pada Tabel 3.11.

Tabel 3. 11 Jadwal jaga perawat

TGL	No Perawat											
	SHIFT 1				SHIFT 2				SHIFT 3			
1	1	2	3	4	13	14	15	16	5	6	7	8
2	2	3	1	4	15	14	13	16	7	6	5	8
3	1	2	3	4	14	15	16	13	10	9	11	12
4	8	6	7	5	9	10	11	12	14	16	13	15
5	5	6	7	8	1	2	3	4	13	14	15	16
6	5	6	7	8	1	2	3	4	13	14	15	16
7	1	2	3	4	5	6	7	8	9	10	11	12
8	16	13	14	15	5	6	7	8	1	2	3	4
9	9	10	11	12	5	6	7	8	1	2	3	4
10	13	14	15	16	9	10	11	12	5	6	7	8
11	13	14	15	16	9	10	11	12	5	6	7	8
12	13	14	15	16	1	2	3	4	5	6	7	8
13	5	6	7	8	1	2	3	4	9	10	11	12
14	13	14	15	16	1	2	3	4	9	10	11	12
15	1	2	3	4	5	6	7	8	9	10	11	12
16	1	2	3	4	5	6	7	8	13	14	15	16
17	1	2	3	4	5	6	7	8	13	14	15	16
18	1	2	3	4	9	10	11	12	13	14	15	16
19	5	6	7	8	1	2	3	4	9	10	11	12
20	5	6	7	8	1	2	3	4	9	10	11	12
21	5	6	7	8	1	2	3	4	13	14	15	16
22	9	10	11	12	5	6	7	8	1	2	3	4
23	9	10	11	12	13	14	15	16	1	2	3	4
24	9	10	11	12	13	14	15	16	1	2	3	4
25	13	14	15	16	9	10	11	12	5	6	7	8
26	13	14	15	16	9	10	11	12	5	6	7	8
27	9	10	11	12	13	14	15	16	5	6	7	8
28	9	10	12	11	13	15	14	16	1	2	3	4
29	13	15	16	14	11	10	9	12	3	2	1	4
30	14	13	16	15	9	11	12	10	4	2	3	1

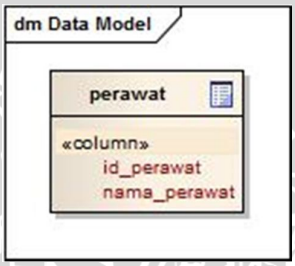
Solusi jadwal yang dihasilkan pada tabel 3.11 menggambarkan perawat dengan id 1,2,3,4 mendapat jadwal jaga pada shift 1 tanggal 1. Perawat tersebut adalah Yusuf, Dewi, Hijrah, dan Husni. Pada solusi jadwal tersebut tidak terdapat pelanggaran baik *soft constraint* maupun *hard constraint.*

BAB IV
PERANCANGAN

Pada bab ini dibahas perancangan database, *user interface* serta perancangan uji coba dan evaluasi sistem.

4.1 Perancangan Tabel Data Perawat

Pada sub bab ini dibahas perancangan data yang digunakan dalam proses implementasi. Pada aplikasi ini terdapat 1 tabel yang digunakan untuk menyimpan data perawat. Perancangan database ditunjukkan pada Gambar 4.1 berikut :



Gambar 4.1 Perancangan tabel data perawat

Atribut dari tabel perawat yaitu dapat dilihat pada tabel 4.1 berikut :

Tabel 4.1 Atribut Tabel Data Perawat

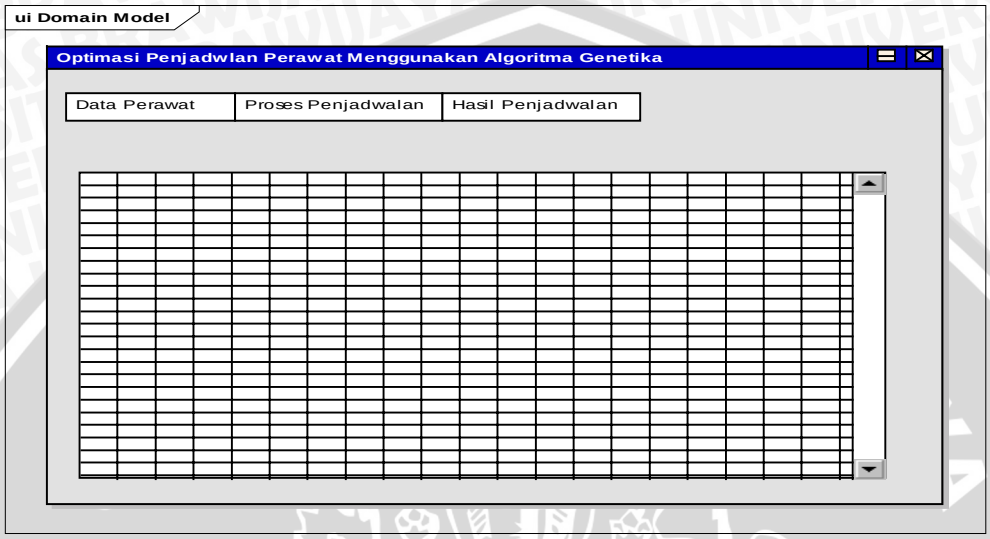
Nama Atribut	Tipe	Keterangan
id_perawat	Integer	Menyimpan id perawat
nama_perawat	Varchar	Menyimpan nama perawat

4.2 Perancangan User Interface

Pada perancangan *user interface* dari aplikasi penjadwalan perawat ini terdiri dari beberapa halaman yaitu halaman input data perawat, halaman penyusunan jadwal, serta halaman hasil proses algoritma genetika.

1. Rancangan *User Interface* Halaman Data Perawat

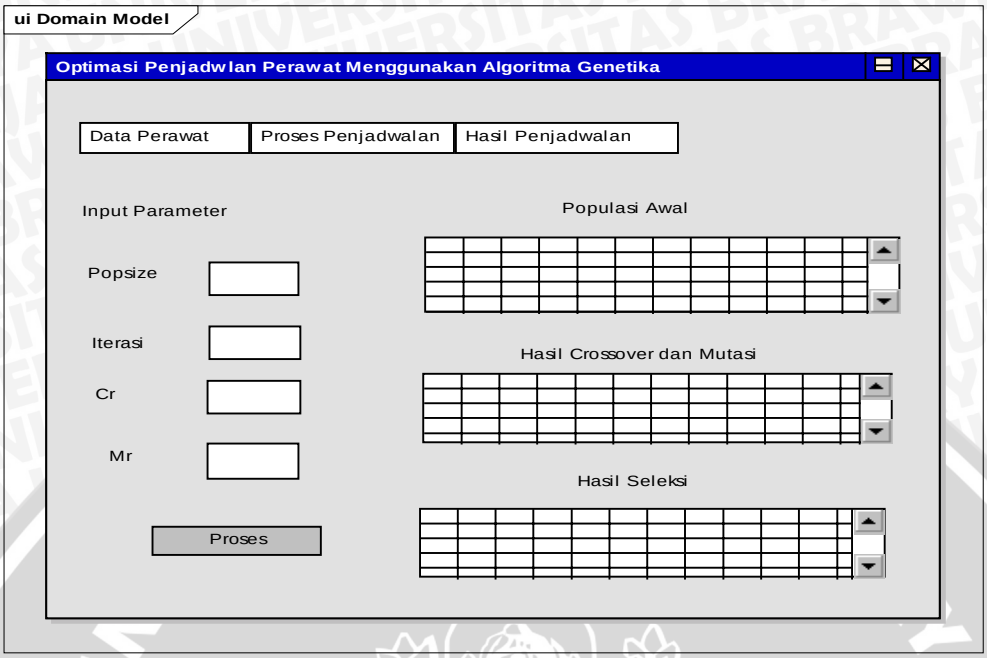
   Halaman data perawat menampilkan data perawat yang bertugas pada ruang ICU. Pada Gambar 4.2 ditampilkan rancangan *user interface* halaman input data perawat.



Gambar 4. 2 Halaman Data Perawat

2. Rancangan *User Interface* Halaman Proses Penjadwalan

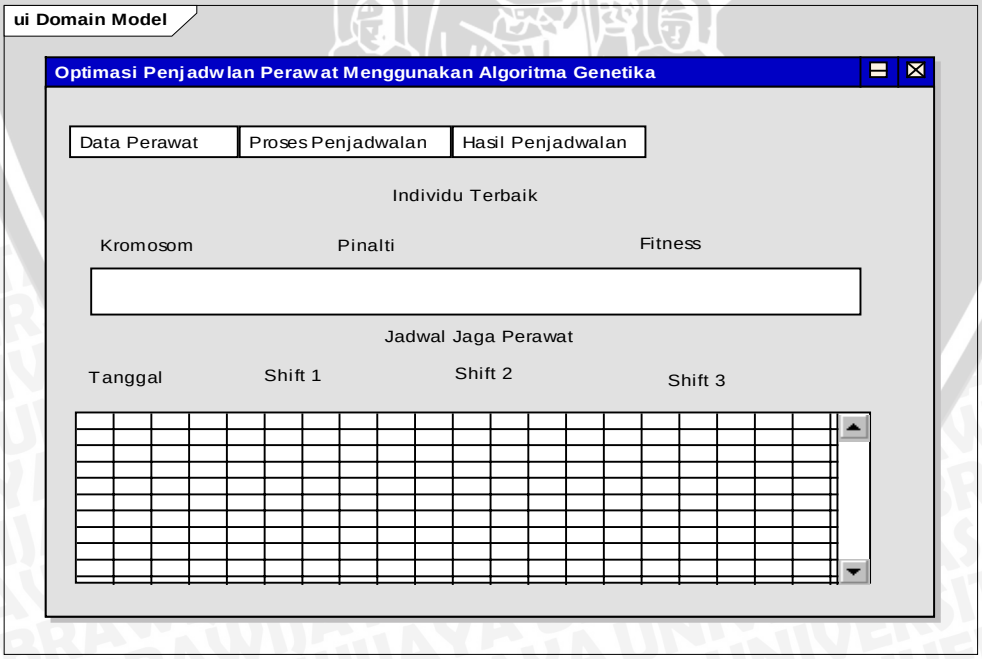
   Halaman proses penjadwalan digunakan untuk proses penjadwalan perawat. Pada Gambar 4.3 ditampilkan rancangan *user interface* halaman proses penjadwalan.



Gambar 4. 3 Halaman Proses Penjadwalan

3. Rancangan *User Interface* Halaman Hasil Penjadwalan

   Hasil akhir proses penjdwalan perawat menggunakan algoritma genetika ditampilkan pada halaman hasil. Pada Gambar 4.4 ditampilkan rancangan *user interface* halaman hasil.



Gambar 4. 4 Halaman Hasil Proses Penjadwalan

4.3 Perancangan Uji Coba dan Evaluasi

Pengujian yang dilakukan dalam sistem ini dilakukan melalui kualitas hasil implementasi algoritma genetika pada penjadwalan perawat dibandingkan dengan penjadwalan manual yang ada. Pada penelitian ini dilakukan pengujian dengan mengetahui nilai parameter algoritma genetika dalam menemukan jadwal jaga perawat yang paling optimal, akan dilakukan beberapa percobaan antara lain:

1. Uji coba untuk menentukan ukuran iterasi yang optimal untuk proses algoritma genetika penjadwalan.
2. Uji coba untuk menentukan ukuran populasi (*PopSize*) yang optimal untuk proses algoritma genetika penjadwalan.
3. Uji coba untuk mencari kombinasi *crossover rate* dan *mutation rate* untuk menghasilkan solusi yang paling optimum.

4.3.1 Data Uji

Spesifikasi data yang diujikan yaitu :

1. Jumlah perawat yang bertugas pada ruang ICU sejumlah 16 orang.
2. Jumlah shift dalam satu hari adalah 3 shift.
3. Jumlah penjadwalan sebanyak 30 hari.

4.3.2 Uji Coba Ukuran Populasi

Pada tahap ini dilakukan uji coba jumlah individu dalam populasi (*popSize*) untuk menemukan solusi jadwal perawat yang paling optimum. Hipotesa pada pengujian ukuran populasi ini adalah semakin besar nilai populasi maka semakin bervariasi solusi jadwal yang ada sehingga akan menghasilkan solusi jadwal yang lebih baik dengan mengetahui nilai *fitness* yang dihasilkan. Uji coba akan dilakukan sebanyak 10 kali dengan banyak gen 100. Sebagai contoh ukuran populasi yang akan diuji yaitu 20, 50, 100, 150, 200, 250, 300, 350, 400. Tabel hasil uji coba terhadap ukuran popsize bisa dilihat pada Tabel 4.1

Tabel 4. 1 Rancangan uji coba ukuran populasi

Ukuran Populasi	Nilai *Fitness* Percobaan										Nilai *Fitness* Rata rata
	1	2	3	4	5	6	7	8	9	10	
20											
50											
100											
150											
200											
250											
300											
350											
400											

4.3.3 Uji Coba Ukuran Generasi

Pada tahap ini dilakukan uji coba terhadap program untuk mengetahui ukuran generasi yang optimal dalam menemukan solusi jadwal yang terbaik. Dalam pengujian ini terdapat hipotesa dimana semakin tinggi ukuran generasi maka semakin tinggi pula kemampuan algoritma genetika untuk mencari solusi terbaik. Pengujian ini bertujuan untuk mengetahui pengaruh perubahan jumlah iterasi terhadap nilai *fitness* yang dihasilkan. Uji coba akan dilakukan sebanyak 10 kali dan digunakan generasi mulai dari 20, 50, 100, 105, 110, dan 125. Ukuran populasi yang digunakan adalah hasil terbaik dari percobaan sebelumnya. Setiap pengujian akan diambil nilai *fitness* masing – masing percobaan kemudian akan dibandingkan dengan setiap nilai generasi yang lain untuk dianalisa. Tabel hasil uji coba terhadap ukuran iterasi bisa dilihat pada Tabel 4.2.

Tabel 4. 2 Rancangan uji coba ukuran generasi

Ukuran Iterasi	Nilai *Fitness* Percobaan ke -										Rata – rata *Fitness*
	1	2	3	4	5	6	7	8	9	10	
20											
50											
100											
105											
110											
125											

4.3.4 Uji Coba Kombinasi *Crossover Rate* dan *Mutation Rate*

Pada tahap ini dilakukan uji coba ini pada nilai *crossover rate* (Cr) dan nilai *mutation rate* (Mr) yang sesuai agar menemukan solusi jadwal yang optimal. Hipotesa dari pengujian ini adalah semakin tinggi nilai Cr dan Mr maka semakin

optimal hasil yang diperoleh. Misalkan nilai *crossover rate* dan *mutation rate* yang diuji yaitu 0, 0.1, 0.3, 0.5, 0.7, 0.9. Pada penelitian ini jumlah generasi dan Pop *size* diambil dari popsize terbaik pada pengujian sebelumnya. Setiap pengujian dilakukan 10 kali percobaan kemudian diambil nilai *fitness*-nya. Tabel hasil uji coba terhadap *crossover rate* dan *mutation rate* bisa dilihat pada Tabel 4.3.

Tabel 4. 3 Rancangan uji coba *crossover rate dan mutation rate*

No.	Kombinasi		Nilai *Fitness* Percobaan										Nilai *Fitness* Rata- rata
	Cr	Mr	1	2	3	4	5	6	7	8	9	10	
1.	0	1											
2.	0.9	0.1											
3.	0.8	0.2											
4.	0.7	0.3											
5.	0.6	0.4											
6.	0.5	0.5											
7.	0.4	0.6											
8.	0.3	0.7											
9.	0.2	0.8											
10.	0.1	0.9											
11.	0	1											

BAB V
IMPLEMENTASI

Pada bab ini akan dibahas implementasi program serta implementasi *user interface.*

5.1 Implementasi Program

Implementasi program dilakukan sesuai dengan metodologi dan perancangan yang telah dibuat. Dalam sistem ini, bahasa pemrograman yang digunakan adalah JAVA.

5.1.1 Tampilan Data Perawat

Tampilan data perawat diambil dari database MySQL yang berisikan data id perawat serta nama perawat. Proses menampilkan data perawat dapat dilihat pada *source code* 5.1

```
public void tampilan_perawat() {
        Object[] row = {"ID Perawat", "Nama Perawat"};
        model = new DefaultTableModel(null, row);
        tabel_perawat.setModel(model);
        model.getDataVector().removeAllElements();
        model.fireTableDataChanged();
        try {
            Class.forName("com.mysql.jdbc.Driver");
            Connection con =
DriverManager.getConnection("jdbc:mysql://localhost/perawat",
"root", "");
            Statement stmt = con.createStatement();
            ResultSet set = stmt.executeQuery("select * from
perawat");
            while (set.next()) {
                Object[] a = new Object[2];
                a[0] = set.getString("idPerawat");
                a[1] = set.getString("Nama_Perawat");
                model.addRow(a);
            }
            set.close();
        } catch (Exception a) {
            System.err.println(a);
        }
    }
```

Source code 5.1 Tampilan Data Perawat

5.1.2 Pembangkitan Populasi Awal

Proses pembangkitan populasi awal dilakukan dengan melakukan random angka sebagai simbol representasi permutasi. Angka random menggambarkan id perawat yaitu 1- 16. Proses pembangkitan populasi awal dapat dilihat pada *source code* 5.2.

```
String chromosome[][] = new String [iterasi + 2][populasi +
(offspringcross * 2) + offspringmut];
int gen[][] = new int[90][4];
for (int ulang = 0; ulang < populasi; ulang++) {
            chromosome[0][ulang] = "";
            for (int k = 0; k < 16; k++) {
                perawat[k] = 0;
            }
            for (int i = 0; i < 90; i++) {
            for (int j = 0; j < 4; j++) {
                do {
                    random = (int) (Math.random() * 16 + 1);
                    if (perawat[random - 1] < 23) {
                        cek = false;
                    } else {
                        cek = true;
                    }
                } while (cek == true);
                perawat[random - 1]++;
                gen[i][j]=random;
                chromosome[0][ulang] += gen[i][j];
                chromosome[0][ulang] += " ";
            }
            }
```

Source code 5.2 Pembangkitan populasi awal

5.1.3 Perhitungan Nilai Penalti

Perhitungan nilai penalti diperoleh dari jumlah pelanggaran yang terjadi pada setiap poin penalti 1 hingga 4. Tabel pelanggaran dapat dilihat pada tabel 3.6. Proses perhitungan penalti dapat dilihat pada source code 5.3.

```
public int pinalti1(String a){
        //antar shift
        int gen[][] = new int[90][4];
        String[] kromosom = a.split(" ");
        int cek[];
        cek = new int[kromosom.length];
        int pinalti1 = 0;
        int pinaltia = 0;
        int pinaltib = 0;
        for(int i = 0;i < kromosom.length;i++)
            {
```

```
                cek[i] = Integer.parseInt(kromosom[i]);
            }

            int cek1=0;
            for(int i=0;i<90;i++){
                for(int j=0;j<4;j++){
                    gen[i][j]=cek[cek1];
                                        cek1++;
                }
            }
          for (int i = 1; i < 90; i++) {
                Arrays.sort(gen[i]);
                for (int j = 0; j <4; j++) {
                if(Arrays.binarySearch(gen[i],gen[i-1][j])>0){
                    pinaltia=pinaltia+1;
                }
                }
            }
          int k = 0;
          int l = 0;
          int gen2[][][]=new int[30][3][5];
          for(int i = 0; i < 90; i++){
            if(l==3){
                k++;
                l=0;
            }
            for(int j = 0; j < 4; j++){
                gen2[k][l][j+1]=gen[i][j];
            }
            l++;
          }
          Arrays.sort(gen2[0][0]);
          for(k = 0; k < 30; k++){
                              for(int j = 1; j < 5; j++){
                  if(Arrays.binarySearch(gen2[k][0],
gen2[k][2][j])>0){

                  }
                  else{
                  }
              }
          }

          pinalti1=pinaltia+pinaltib;
          return pinalti1;
    }
public int pinalti2(String a){
        int gen[][] = new int[90][4];
        int gen1[][] = new int[90][4];
        int perawat[] = new int [16];
        int jmlpinalti[] = new int [16];
        String[] kromosom = a.split(" ");
        int cek[] = new int[kromosom.length];
        int pinalti=0;

        for(int i = 0;i < kromosom.length;i++)
            {

                cek[i] = Integer.parseInt(kromosom[i]);
```

```
            }
           int cek1=0;
            for(int i=0;i<90;i++){
                for(int j=0;j<4;j++){
                    gen[i][j]=cek[cek1];
                    //System.out.print(gen[i][j]);
                cek1++;
                }
            }
        int cek2=0;
        for (int i = 2; i < 90; i=i+3) {
           //System.out.println("gen");
                for (int j = 0; j < 4; j++) {
               gen1[i][j]=gen[i][j];
                //System.out.print(gen1[i][j]+" ");
                cek2++;

                }
                //System.out.println("");
            }
         for (int i = 2; i < 90; i=i+3) {
                for (int j = 0; j < 4; j++) {
                    for(int k = 0; k < 16; k++){
                        int h=k+1;
                        if(h==gen1[i][j]){
                            perawat[k]++;
                        }
                    }
                }
            }
         for(int k = 0; k < 16; k++){
             //System.out.println(perawat[k]);
             if (perawat[k]>8){
                 jmlpinalti[k]=perawat[k]-8;
             }
             else{
                 jmlpinalti[k]=0;
             }
             pinalti=pinalti+jmlpinalti[k];
         }
    }
     public int pinalti3(String a){
        int gen[][] = new int[90][4];
        int gen1[][] = new int[90][4];
        String[] kromosom = a.split(" ");
        int cek[] = new int[kromosom.length];
        int pinalti3=0;
        for(int i = 0;i < kromosom.length;i++){
            cek[i] = Integer.parseInt(kromosom[i]);
        }
        int cek1=0;
        for(int i=0;i<90;i++){
           for(int j=0;j<4;j++){
               gen[i][j]=cek[cek1];
               gen1[i][j]=gen[i][j];
                //System.out.print(gen[i][j]);
                cek1++;
            }
        }
```

```
        for(int i=0;i<90;i++){
        for(int j=0;j<4;j++){
          // System.out.print(gen[i][j] + " ");
            for(int k=j+1;k<4;k++){
                 if(gen[i][j]==gen1[i][k]) {
                    pinalti3++;
                }
            }
        }
        }
        }
       public int pinalti4(String a, String []idP, String []tgl){
        String idPer[]=new String[idP.length];
        int temp_id[] = new int[idP.length];
        int temp_tgl[] = new int[tgl.length];
        int temp_pinalti4[]= new int[idP.length];
        int pinalti4=0;

        for(int i=0; i<idPer.length; i++){
              int pinalti=0;
              temp_id[i]=Integer.parseInt(idP[i]);
              temp_tgl[i]=Integer.parseInt(tgl[i]);
              int no=temp_id[i];
               int tanggal=temp_tgl[i];
               int gen[][] = new int[30][12];
               String[] kromosom = a.split(" ");
               int cek[] = new int[kromosom.length];
               for(int j = 0;j < kromosom.length;j++){
                   cek[j] = Integer.parseInt(kromosom[j]);
               }

               int cek1=0;
               for(int j=0;j<30;j++){
                  for(int k=0;k<12;k++){
                      gen[j][k]=cek[cek1];
                       //System.out.print(gen[i][j]);
                       cek1++;
                   }
               }
               //System.out.println("");
               for(int j=0;j<12;j++){
               if(no == gen[tanggal-1][j]) {
                       pinalti = 1;
                   }
                }
              temp_pinalti4[i]=pinalti;
              pinalti4=pinalti4 + temp_pinalti4[i];
          }
        return pinalti4;
    }
```

Source code 5.3 Perhitungan nilai penalti

5.1.4 Perhitungan Nilai *Fitness*

Perhitungan nilai *fitness* diperoleh dengan membagi angka 100 dengan jumlah penalti 1 hingga 4 yang telah dikalikan konstanta sesuai dengan jenis *constraint*. Proses perhitungan *fitness* dapat dilihat pada *source code* 5.4.

```
  public double fitness(int pinalti1, int pinalti2, int pinalti3,
int pinalti4){

        intfitness=
(5*pinalti1+5*pinalti2+20*pinalti3+20*pinalti4)+1;
        double fitnes1 = 100/ (double) fitness;

        int angkaSignifikan = 4;
        double temp = Math.pow(10, angkaSignifikan);
        double y = (double) Math.round(fitnes1*temp)/temp;
        return y;
    }
```

Source code 5.4 Perhitungan nilai *fitness*

5.1.5 Proses *Crossover*

Metode *crossover* yang digunakan adalah *one cut-point crossover*. Pemilihan induk dilakukan secara random. Hasil *crossover* adalah 2 *offspring* jika hasil perhitungan c_r dikalikan popsize menghasilkan nilai genap, dan 1 *offspring* jika hasil perhitungan c_r dikalikan popsize menghasilkan nilai ganjil. Proses *crossover* dapat dilihat pada source code 5.5

```
public String[] crossover(int titik, String p1, String p2) {
        String pisah1[] = p1.split(" ");
        String pisah2[] = p2.split(" ");
        String anak1 = "";
        String anak2 = "";
        String anaks[] = new String[2];

        for (int i = 0; i < titik; i++) {
            anak1 += pisah1[i];
            anak1 += " ";
            anak2 += pisah2[i];
            anak2 += " ";
        }
        for (int i = titik; i < pisah1.length; i++) {
            anak1 += pisah2[i];
            anak1 += " ";
            anak2 += pisah1[i];
            anak2 += " ";
        }

        System.out.println(anak1);
```

```
        System.out.println(anak2);
        anaks[0] = anak1;
        anaks[1] = anak2;
        return anaks;
    }
    int anak3[] = new int[360];
    int temp[] = new int[360];
    String anak = "";
    String pisah[];
    String anaks[];

//crossover
int counter;
        for (counter = 0; counter < iterasi; counter++) {
        Object[] rows = {"Individu", "Parent Asal", "Kromosom",
"Fitness"};
        String parent1, parent2;
        int random1, random2;
        int t;
        String offspring[] = new String[2];
        int jumlah_individu;
        String temp1;
        double temp;
        String parent3;
        int random3;
        offspring = new String[2];
        System.out.println("Hasil Crossover");
        jumlah_individu = populasi;
        Object [] b  = new Object [4];
        for (int ulang = populasi; ulang < offspringcross +
populasi; ulang++) {
            random1 = (int) (Math.random() * populasi);
            random2 = (int) (Math.random() * populasi);
            parent1 = chromosome[counter][random1];
            parent2 = chromosome[counter][random2];
            t = (int) (double) (Math.random() * 360 - 1);
            if (t == 0) {
                t = 359;
            }
            for (int k = 0; k < offspring.length; k++) {
                if (jumlah_individu < (populasi * crossrate) +
populasi) {
                    chromosome[counter][jumlah_individu]         =
offspring[k];
                b[0] = jumlah_individu +1;
                int tmp=random2+1;

fitness[jumlah_individu]=fitness(pinalti1(chromosome
[counter][jumlah_individu]),pinalti2(chromosome[counter][jumlah_in
dividu]),pinalti3(chromosome[counter][jumlah_individu]),pinalti4(c
hromosome[counter][jumlah_individu]));
            }
            }
        }
```

Source Code 5.5 Proses *Crossover*

5.1.6 Proses Mutasi

Metode mutasi yang digunakan adalah *reciprocal exchange mutation*. Proses pertama yang dilakukan yaitu memilih induk yang akan dimutasi secara random. Langkah selanjutnya yaitu memilih dua gen pada kromosom tersebut untuk dilakukan proses penukaran gen. Proses mutasi dapat dilihat pada source code 5.6.

```
    public String mutasi(int titik1, int titik2, String p) {
        anak = "";
        pisah = p.split(" ");
        //System.out.println(p);
        anak3 = new int[360];
        temp = new int[360];
        for (int i = 0; i < pisah.length; i++) {
            if (i < anak3.length) {
                anak3[i] = Integer.parseInt(pisah[i]);
                temp[i] = Integer.parseInt(pisah[i]);
            }
        }
        anak3[titik1] = temp[titik2];
        anak3[titik2] = temp[titik1];
        for (int i = 0; i < pisah.length; i++) {
            if (i < anak3.length) {
                anak += anak3[i];
                anak += " ";
            }
        }
        System.out.println(anak);
        return anak;
    }//mutasi
        int t1, t2;
        int jmlmutasi=jumlah_individu + offspringmut;
        for (int ulang = jumlah_individu; ulang < jmlmutasi;
ulang++) {
            random3 = (int) (Math.random() * populasi);
            parent3 = chromosome[counter][random3];
            t1 = (int) (Math.random() * 360 - 1);
            do {
                t2 = (int) (Math.random() * 360 - 1);
            } while (t2 == t1);
            System.out.println("Titik 1 " + (t1 + 1));
            System.out.println("Titik 2 " + (t2 + 1));
            System.out.println("parent 1 : " + parent3);
            chromosome[counter][ulang] = mutasi(t1, t2, parent3);

fitness[jumlah_individu]=fitness(pinalti1(chromosome[counter][juml
ah_individu]),pinalti2(chromosome[counter][jumlah_individu]),pinal
ti3(chromosome[counter][jumlah_individu]),pinalti4(chromosome[coun
ter][jumlah_individu]));
            jumlah_individu++;
        }
```

Source Code 5.6 Proses Mutasi

5.1.7 Proses Seleksi Elitism

Proses seleksi dilakukan dengan menggunakan metode elitism. Proses seleksi elitism ini dilakukan dengan cara mengurutkan nilai *fitness* dari yang terbesar sampai yang terkecil dari kromosom populasi awal, hasil crossover dan mutasi, kemudian diambil individu terbaik sejumlah populasi awal untuk menjadi induk pada proses selanjutnya. Proses seleksi elitism dapat dilihat pada source code 5.7.

```
//seleksi
            for (int ulang = 0; ulang < jumlah_individu; ulang++)
{
                    temp_chromosome[counter][ulang]             =
chromosome[counter][ulang];
                    temp_fitness[counter][ulang]                =
fitness(pinalti1(chromosome[counter][ulang]),pinalti2(chromosome[c
ounter][ulang]),pinalti3(chromosome[counter][ulang]),pinalti4(chro
mosome[counter][ulang]));

//System.out.println(temp_chromosome[counter][ulang]  +  "  =  "  +
temp_fitness[counter][ulang]);
            }
        //buble sort
             for (int i = 0; i < jumlah_individu; i++) {
                for (int k = i+1; k < jumlah_individu - i; k++) {
                    if       (temp_fitness[counter][k]        >
temp_fitness[counter][i]) {
                        temp = temp_fitness[counter][k];
                        temp1 = temp_chromosome[counter][k];
                        temp_fitness[counter][k]                =
temp_fitness[counter][i];
                        temp_chromosome[counter][k]             =
temp_chromosome[counter][i];
                        temp_fitness[counter][i] = temp;
                        temp_chromosome[counter][i] = temp1;
                    }
                }
            }
            for (int ulang = 0; ulang < populasi; ulang++) {
            chromosome[counter        +        1][ulang]        =
temp_chromosome[counter][ulang];
                c[0] = ulang+1;
                c[1] = temp_chromosome[counter][ulang];
                c[2] = temp_fitness[counter][ulang];
                model.addRow(c);
            }
```

Source Code 5.7 Proses Seleksi *Elitsm*

5.1.8 Proses Pemilihan Individu Terbaik

Individu terbaik diperoleh dari kromosom yang memiliki nilai *fitness* paling besar dari setiap generasi setelah dilakukan keseluruhan proses algoritma genetika. Proses pemilihan individu terbaik dapat dilihat pada source code 5.8.

```
String tes= temp_chromosome[counter][0];
     String[] kromosom = tes.split(" ");
     int tes1[] = new int[kromosom.length];
     for(int i = 0;i < kromosom.length;i++){
              tes1[i] = Integer.parseInt(kromosom[i]);
     }
     int cek1=0;
     int rank[][] = new int[30][12];
     for(int i=0;i<30;i++){
        for(int j=0;j<12;j++){
           rank[i][j]=tes1[cek1];
           cek1++;
        }
     }
```

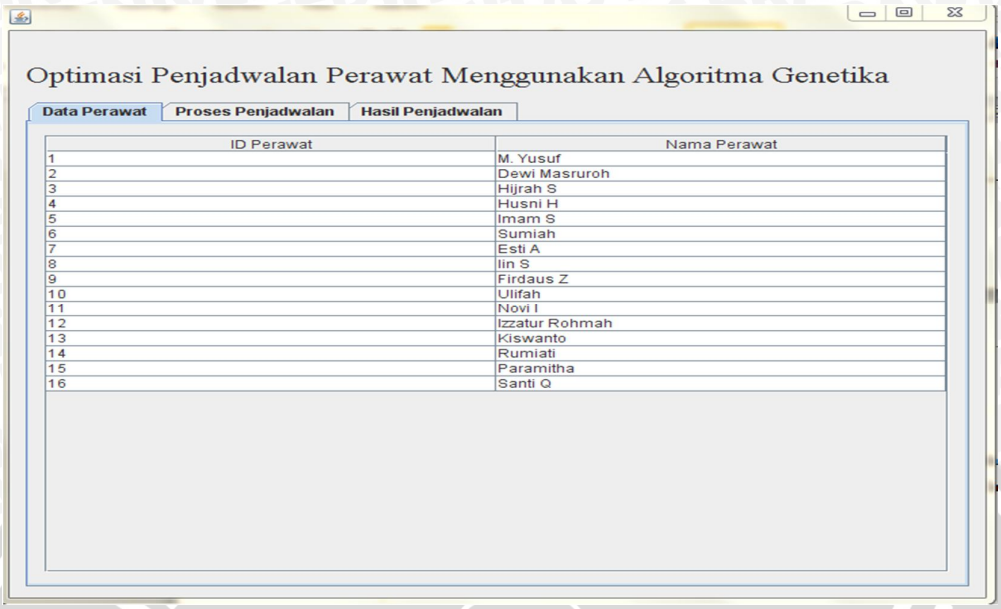
Source Code 5.8 Proses Pemilihan Individu Terbaik

5.2 Implementasi *User Interface*

User interface sistem terdiri dari 3 tab yaitu tab data perawat, proses penjadwalan serta hasil penjadwalan. Tab data perawat berisi data id perawat serta nama perawat. Tab penjadwalan digunakan untuk input parameter algoritma, memproses serta melihat hasil algoritma genetika berupa populasi awal, hasil *crossover* dan mutasi hingga hasil seleksi. Tab hasil penjadwalan berisi individu terbaik dan jadwal jaga perawat. Implementasi *user interface* dapat dilihat pada Gambar 5.1, 5.2 dan 5.3.

5.2.1 Implementasi *User Interface* Halaman Data Perawat

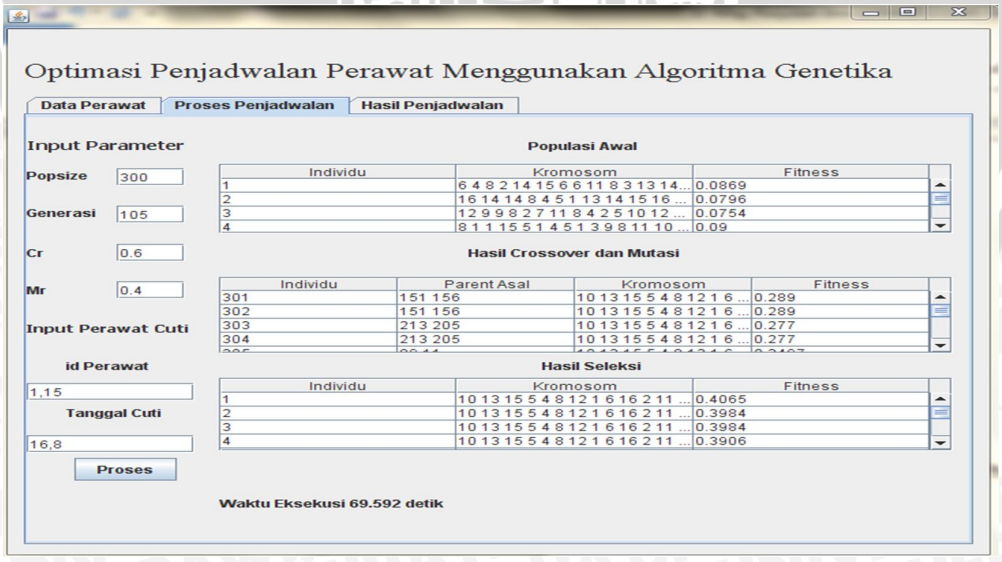
Halaman data perawat akan muncul pertama kali setelah program dijalankan. Halaman ini berisi data id perawat serta nama perawat. Implementasi halaman data perawat dapat dilihat pada Gambar 5.1



Gambar 5. 1 Halaman Data Perawat

5.2.2 Implementasi *User Interface* Proses Penjadwalan

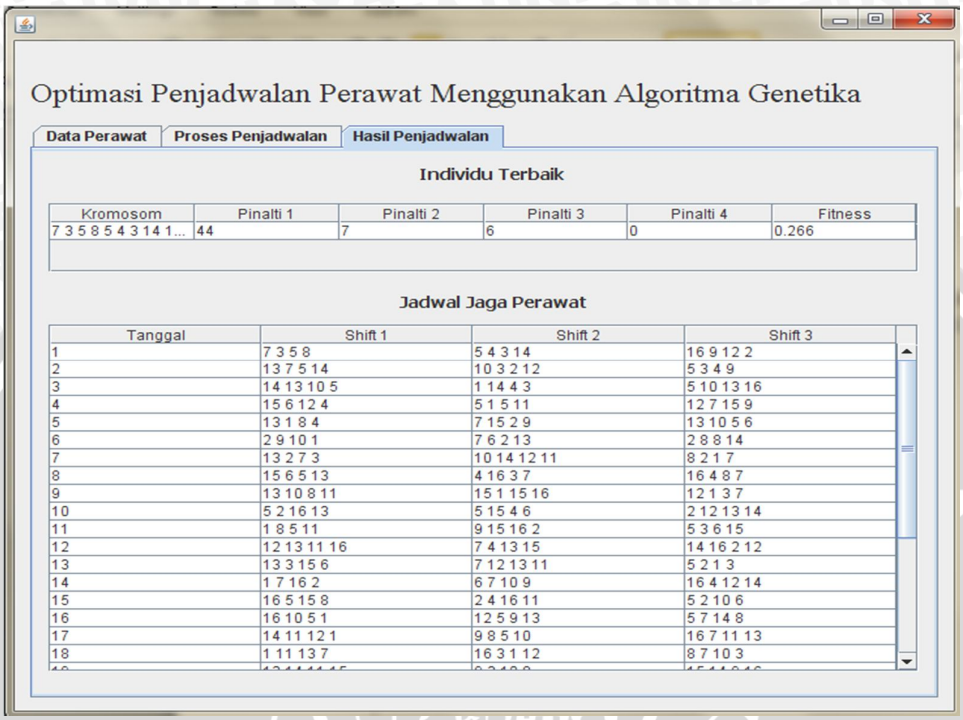
Halaman proses penjadwalan digunakan untuk input parameter algoritma genetika berupa *popsize*, generasi, cr dan mr. pada halaman ini juga terdapat tombol untuk memproses serta melihat hasil algoritma genetika berupa populasi awal, hasil *crossover* dan mutasi hingga hasil seleksi. Implementasi halaman proses penjadwalan dapat dilihat pada Gambar 5.2.



Gambar 5. 2 Halaman proses Penjadwalan

5.2.3 Implementasi *User Interface* Hasil Penjadwalan

Halaman hasil penjadwalan berisi individu terbaik dan jadwal jaga perawat. Implementasi halaman hasil penjadwalan dapat dilihat pada Gambar 5.3.



Gambar 5. 3 Halaman Hasil Penjadwalan

BAB VI
PENGUJIAN DAN ANALISIS

Pada bab ini akan dibahas analisis dan pengujian terhadap sistem yang telah diimplementasikan. Terdapat 3 proses pengujian yaitu pengujian terhadap ukuran populasi, pengujian terhadap banyaknya generasi dan pengujian terhadap *crossover rate* dan *mutation rate.*

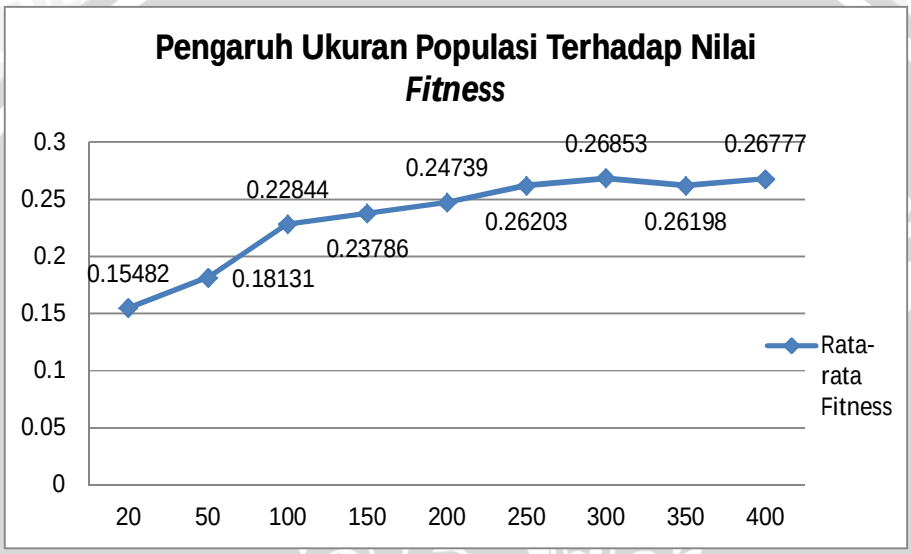
6.1 Pengujian Ukuran Populasi

Pengujian pertama yaitu pengujian ukuran populasi terhadap nilai *fitness*. Pengujian ukuran populasi digunakan untuk menentukan ukuran populasi yang terbaik agar menghasilkan solusi terbaik dalam kasus ini. Jumlah generasi yang digunakan adalah 50 generasi dengan *crossover* rate 0.6 dan *mutation* rate 0.4 (Tjatjo, 2008) ukuran populasi yang diuji adalah 20, 50, 100, 150, 200, 250, 300, 350 serta 400. Pengujian ukuran populasi dilakukan sebanyak 10 kali. Hasil pengujian dapat dilihat pada Tabel 6.1 berikut ini :

Tabel 6. 1 Hasil Pengujian Ukuran Populasi

Nilai popsize	Nilai *Fitness*										Rata – rata *fitness*	Rata-rata waktu Eksekusi (detik)
	Pengujian ke -											
	1	2	3	4	5	6	7	8	9	10		
20	0,14 06	0,15 48	0,16 23	0,14 27	0,15 36	0,16 37	0,14 47	0,17 36	0,16 64	0,14 58	0,154 82	2,397 2
50	0,19 01	0,19 19	0,17 36	0,19 96	0,17 06	0,17 06	0,18 66	0,17 83	0,17 51	0,17 67	0,181 31	5,510 1
100	0,21 23	0,19 19	0,22 68	0,22 42	0,21 46	0,24 94	0,23 2	0,24 04	0,22 68	0,26 6	0,228 44	10,72 51
150	0,26 6	0,23 47	0,24 04	0,24 04	0,23 2	0,21 93	0,23 47	0,25 25	0,21 23	0,24 63	0,237 86	15,32 08
200	0,25 58	0,22 94	0,25 25	0,26 25	0,22 17	0,25 58	0,25 91	0,23 47	0,25 91	0,24 33	0,247 39	20,04 13
250	0,25 58	0,28 9	0,23 75	0,23 47	0,23 75	0,28 49	0,28 9	0,26 95	0,28 49	0,23 75	0,262 03	25,78 07
300	0,28 49	0,25 58	0,28 9	0,27 32	0,25 91	0,25 25	0,26 95	0,26 6	0,24 63	0,28 9	0,268 53	29,91 91
350	0,26 6	0,25 25	0,23 2	0,26 6	0,26 95	0,26 95	0,24 33	0,26 6	0,26 6	0,28 9	0,261 98	34,74 42
400	0,26 25	0,26 95	0,26 95	0,26 95	0,27 32	0,26 95	0,26 95	0,26 25	0,26 25	0,26 95	0,267 77	39,82 58

Berdasarkan hasil pengujian pada tabel 6.1, semakin besar ukuran populasi atau *popsize* maka rata – rata *fitness* yang dihasilkan cenderung semakin besar. Pada ukuran populasi 20 individu memiliki nilai rata-rata *fitness* terkecil yaitu 0,15482 sedangkan pada ukuran populasi 300 individu, rata – rata nilai *fitness* mencapai nilai terbaik yaitu 0,26853. Grafik pengaruh jumlah populasi terhadap nilai *fitness* dapat dilihat pada Gambar 6.1.



Gambar 6. 1 Grafik Pengujian Ukuran Populasi

Penambahan ukuran populasi akan memungkinkan algoritma genetika untuk mengeksplorasi area pencarian dan sekaligus mendapatkan solusi yang lebih baik. Tetapi pada batas tertentu ukuran populasi yang terlalu besar akan membebani waktu komputasi dan kenaikan *fitness* yang didapatkan tidak terlalu siginifikan(Mahmudy, Marian & Luong, 2013). Dari hasil uji coba populasi diperoleh ukuran populasi dengan hasil optimal yaitu 300 individu.

6.2 Pengujian Banyaknya Generasi

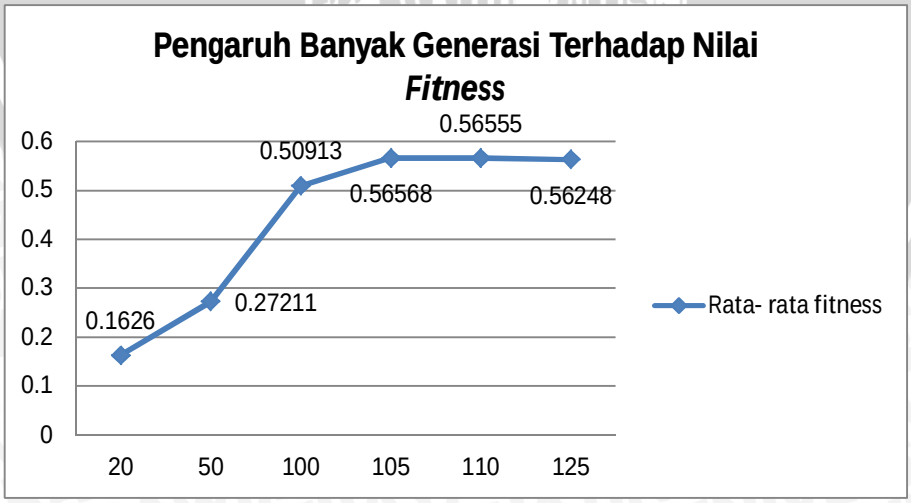
Pengujian kedua yaitu pengujian ukuran generasi terhadap nilai *fitness*. Pengujian ukuran generasi digunakan untuk menentukan banyak generasi yang terbaik agar menghasilkan solusi terbaik dalam kasus ini. Pada uji coba generasi ini digunakan ukuran populasi 300 yang diperoleh dari hasil uji coba ukuran populasi yang telah dilakukan sebelumnya yang dianggap dapat menghasilkan

rata – rata nilai *fitness* terbaik. Ukuran generasi yang diuji adalah 20, 50, 100, 105, 110 dan 125 dengan *crossover* rate 0.6 dan *mutation* rate 0.4. Pengujian ukuran generasi dilakukan sebanyak 10 kali. Hasil pengujian dapat dilihat pada Tabel 6.2 berikut ini :

Tabel 6. 2 Hasil Pengujian Generasi

Nilai generasi	Nilai *Fitness*										Rata – rata *fitness*	Rata-rata waktu Eksekusi
	Pengujian ke -											
	1	2	3	4	5	6	7	8	9	10		
20	0,1678	0,165	0,161	0,1637	0,1799	0,1692	0,1664	0,1572	0,149	0,1468	0,1626	12,6516
50	0,2809	0,277	0,2695	0,2809	0,2849	0,2732	0,266	0,2347	0,277	0,277	0,27211	29,9744
100	0,4854	0,5102	0,5376	0,5682	0,5102	0,5236	0,4854	0,463	0,4975	0,5102	0,50913	58,6617
105	0,5376	0,6024	0,5376	0,5682	0,5848	0,5525	0,5682	0,5682	0,5525	0,5848	0,56568	62,7795
110	0,5525	0,6024	0,5376	0,5236	0,5848	0,641	0,6024	0,5525	0,4739	0,5848	0,56555	104,0354
125	0,5525	0,5525	0,5525	0,6024	0,5525	0,5525	0,5525	0,6024	0,5525	0,5525	0,56248	108,8354

Berdasarkan hasil pengujian pada tabel 6.2, semakin besar ukuran generasi maka rata – rata *fitness* yang dihasilkan cenderung semakin besar. Pada ukuran populasi 20 individu memiliki nilai rata-rata *fitness* terkecil yaitu 0,1626 sedangkan pada ukuran populasi 105 individu, rata – rata nilai *fitness* mencapai nilai terbaik yaitu 0,56568. Grafik pengaruh jumlah populasi terhadap nilai *fitness* dapat dilihat pada Gambar 6.2.



Gambar 6. 2 Grafik Pengujian Banyaknya Generasi

Generasi optimal pada pengujian ini adalah 105. Jika dilanjutkan ke generasi yang lebih besar maka perubahan rata-rata *fitness* tidak signifikan bahkan mengalami penurunan sehingga menghasilkan anak yang hampir sama dengan induknya dan waktu komputasi juga lebih lama. Hasil pengujian ini membuktikan jika jumlah generasi terlalu sedikit maka area pencarian algoritma semakin sempit, sehingga solusinya kurang optimal. Sebaliknya jika semakin banyak generasi maka semakin besar waktu komputasinya dan belum tentu menghasilkan solusi yang lebih optimal (Mahmudy, 2013). Pola kenaikan *fitness* seperti ini juga didapatkan oleh Sundarningsih, Mahmudy dan Sutrisno (2015) yang menerapkan algoritma genetika untuk optimasi *vehicle routing problem with time window* (VRPTW) serta penelitian yang dilakukan oleh Pitaloka, Mahmudy dan Sutrisno (2014) yang menerapkan *hybrid* algoritma genetika untuk permasalahan *vehicle routing problemwith time window*(VRPTW).

6.3 Pengujian *Crossover rate dan Mutation rate*

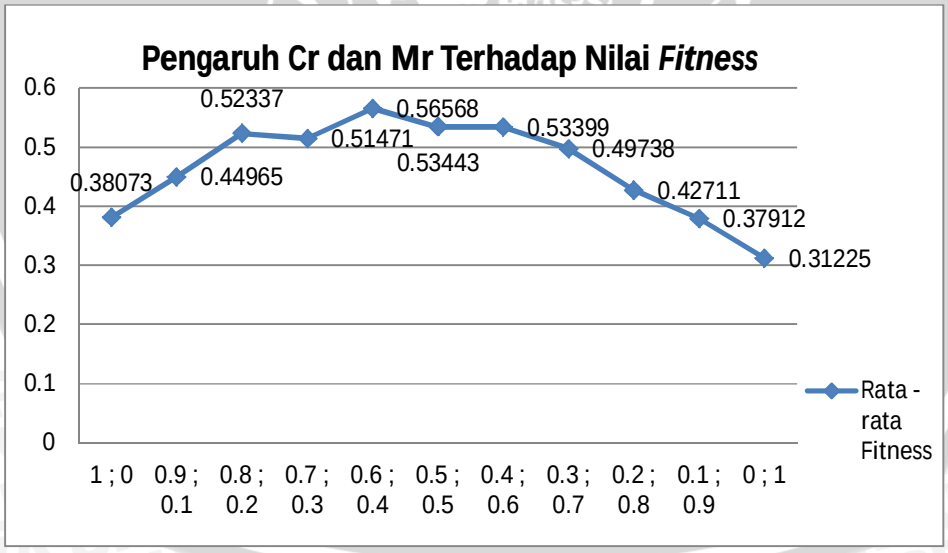
Uji coba berdasarkan *crossover rate* (*cr*) dan *mutation rate* (*mr*) dilakukan untuk mengetahi kombinasi *cr* dan *mr* yang paling baik untuk menghasilkan *fitness* terbaik. Nilai *cr* dan *mr* yang digunakan dalam uji coba ini antara 0 dan 1. Pengujian pada *cr* dan *mr* ini juga menggunakan hasil uji coba yang telah dilakukan sebelumnya yaitu hasil uji coba ukuran populasi dan uji coba banyaknya generasi. Ukuran populasi sebesar 200 dan generasi sebesar 150. Pengujian ukuran populasi ini dilakukan sebanyak 10 kali. Hasil pengujian dapat dilihat pada Tabel 6.3 berikut ini :

Tabel 6. 3 Hasil Pengujian Cr dan Mr

Kombinasi		Nilai *Fitness*										Rata – rata *fitness*	Rata-rata waktu Eksekusi
		Pengujian ke -											
Cr	Mr	1	2	3	4	5	6	7	8	9	10		
1	0	0,4425	0,4237	0,3984	0,3759	0,3268	0,4525	0,3623	0,3115	0,3378	0,3759	0,38073	61,8301
0.9	0.1	0,4975	0,4975	0,4065	0,4975	0,4065	0,4237	0,4854	0,4329	0,4065	0,4425	0,44965	72,1247
0.8	0.2	0,5102	0,5236	0,5236	0,5236	0,4975	0,5102	0,6024	0,4525	0,5376	0,5525	0,52337	68,415
0.7	0.3	0,5236	0,6211	0,4739	0,4854	0,4739	0,4854	0,4975	0,5236	0,5525	0,5102	0,51471	74,0322

0.6	0.4	0,5376	0,6024	0,5376	0,5682	0,5848	0,5525	0,5682	0,5682	0,5525	0,5848	0,56568	62,7795
0.5	0.5	0,463	0,5376	0,463	0,5848	0,5102	0,4975	0,4975	0,5682	0,6849	0,5376	0,53443	74,1887
0.4	0.6	0,6211	0,5525	0,4425	0,5236	0,5102	0,4975	0,5236	0,5376	0,5102	0,6211	0,53399	67,0802
0.3	0.7	0,4425	0,463	0,4854	0,5376	0,463	0,5682	0,4975	0,4854	0,5682	0,463	0,49738	66,6094
0.2	0.8	0,463	0,4329	0,3906	0,4237	0,4329	0,463	0,4739	0,369	0,4237	0,3984	0,42711	69,9607
0.1	0.9	0,3623	0,3831	0,3831	0,3623	0,3497	0,3906	0,369	0,4237	0,369	0,3984	0,37912	73,7791
0	1	0,2849	0,3268	0,277	0,3165	0,2933	0,3268	0,3067	0,3497	0,3559	0,2849	0,31225	62,315

Rata -rata *fitness* yang diperoleh sangat beragam karena tidak ada suatu ketetapan nilai cr dan mr yang digunakan untuk memperoleh solusi optimal. Dalam menentukan kombinasi parameter yang tepat dipengaruhi oleh permasalahan yang akan diselesaikan (Mahmudy, 2013). Pada uji coba ini didapatkan kombinasi *cr* = 0.6 dan *mr* = 0.4 menghasilkan rata – rata *fitness* terbaik yaitu 0,56568 sedangkan pada kombinasi *cr* = 0 dan *mr* = 1 menghasilkan rata – rata *fitness* terendah yaitu 0,31225 Grafik pengaruh cr dan mr terhadap nilai *fitness* dapat dilihat pada Gambar 6.3.



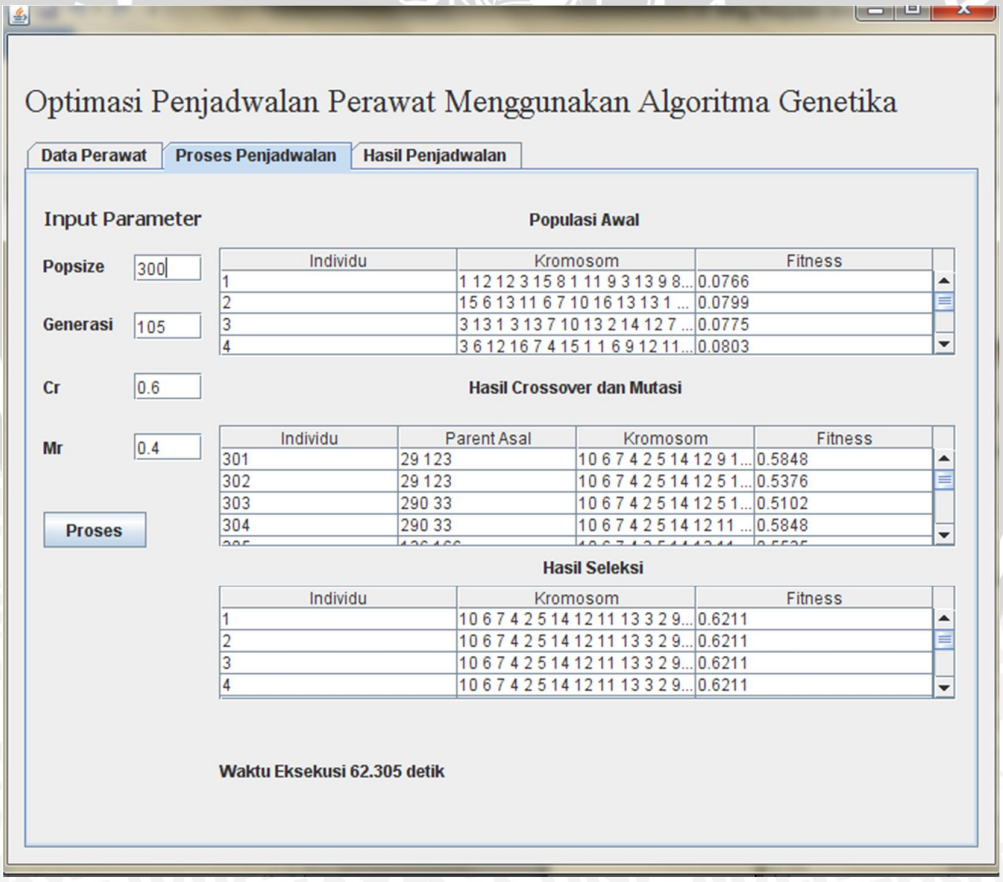
Gambar 6. 3 Grafik Pengujian Ukuran Cr dan Mr

Nilai *cr* yang terlalu rendah dan *mr* terlalu besar mengakibatkan menurunnya kemampuan algoritma genetika untuk belajar ke generasi sebelumnya dan tidak mampu mengeksplorasi daerah *optimum local* (Mahmudy,

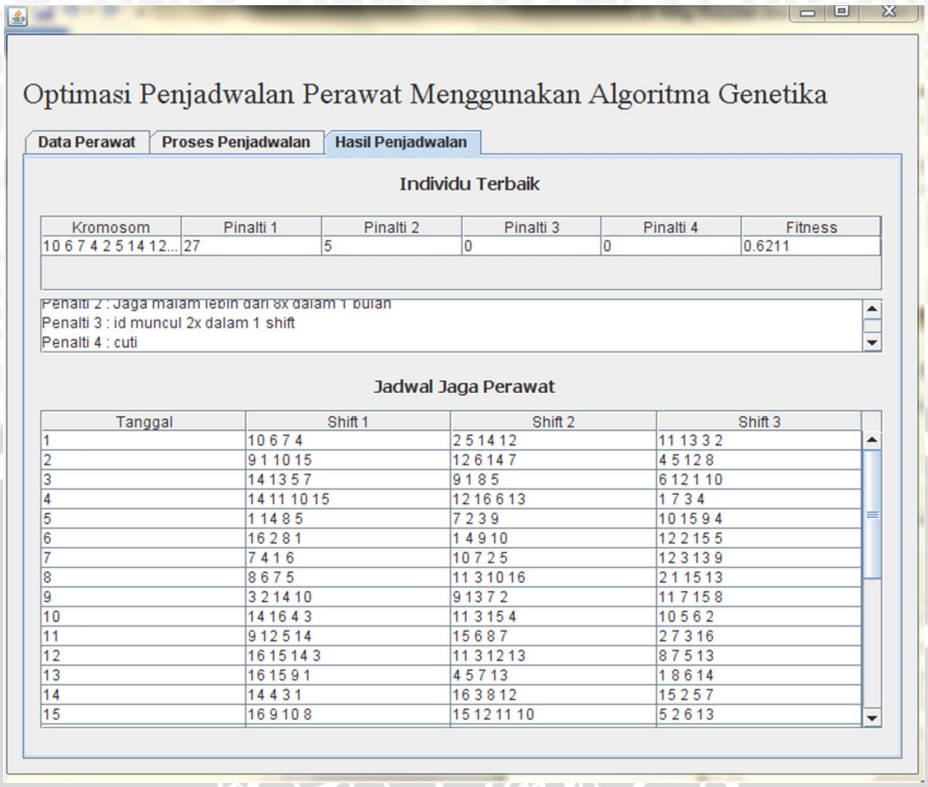
2013). Selain itu nilai *mr* yang terlalu rendah dan *cr* terlalu besar mengakibatkan konvergensi dini yaitu tidak adanya perubahan terlalu besar dalam mendapatkan solusi ketika di uji coba pada beberapa generasi saja (Mahmudy, Marian & Luong, 2014).

6.4 Analisa Hasil

Pada pengujian yang dilakukan diperoleh parameter ukuran populasi yang mendekati optimal yaitu 300 individu dengan 105 generasi serta kombinasi cr = 0.6 dan mr = 0.4. Parameter tersebut digunakan untuk melakukan pengujian kembali sebanyak 10 kali dan didapatkan nilai rata – rata penalti 1 sebanyak 34, penalti 2 sebanyak 3, penalti 3 sebanyak 0 dan penalti 4 sebanyak 0 dengan nilai *fitness* rata- rata yaitu 0,54894. Hasil pengujian penjadwalan perawat dapat dilihat pada Gambar 6.4 dan 6.5 berikut.



Gambar 6. 4 Proses Penjadwalan



Gambar 6. 5 Hasil Penjadwalan

Dari hasil penjadwalan diatas, masih terdapat pelanggaran penalti 1 yaitu perawat berjaga pada shift yang berurutan serta penalti 2 yaitu perawat berjaga malam lebih dari 8 kali. Hal ini menunjukkan bahwa penjadwalan perawat merupakan permasalahan yang cukup rumit dan tidak mudah dicari solusi optimumnya. Proses pengkodean kromosom yang dilakukan secara random serta panjang kromosom yang panjang menyebabkan masih timbulnya penalti dalam proses penjadwalan. Faktor lain yang mempengaruhi munculnya penalti pada hasil penjadwalan adalah kombinasi metode *crossover* serta mutasi yang digunakan. Dalam pengujian, metode *crossover one-cut point* terbukti dapat menghasilkan anak yang memiliki *fitness* lebih baik namun metode mutasi *reciprocal exchange* tidak menghasilkan anak dengan perubahan nilai *fitness* yang signifikan serta cenderung sama dengan *parent.* Oleh karena itu diperlukan modifikasi pada metode *crossover* dan mutasi agar dihasilkan jadwal yang optimal.

BAB VII
PENUTUP

7.1 Kesimpulan

Berdasarkan hasil implementasi dan pengujian dalam penerapan algoritma genetika untuk penjadwalan perawat, maka dapat diambil beberapa kesimpulan sebagai berikut :

1. Pada kasus ini algoritma genetika direpresentasikan dengan representasi permutasi berbasis kode perawat. Bentuk representasi kromosom yang digunakan memiliki panjang kromosom pada interval [1…360] yang pembangkitan kromosomnya *random*.
2. Metode *crossover* yang digunakan yaitu *one-cut point* dan untuk proses mutasinya menggunakan metode *reciprocal exchange mutation*. Sedangkan metode seleksi yang digunakan yaitu *elitism* yang menghasilkan nilai *fitness* paling optimal dari generasi terakhir. Namun nilai *fitness* yang dihasilkan masih belum maksimum karena dengan panjang kromosom hingga 360 membutuhkan waktu komputasi yang lama dan hasil *random* dari proses algoritma genetika yang lebih bervariasi sehingga sulit untuk mendapatkan nilai *fitness* yang optimal.
3. Untuk menentukan kualitas dari solusi penjadwalan yang dihasilkan diukur dari nilai *fitness*nya. Perhitungan *fitness* didapat dari jumlah penalti yang telah dikalikan dengan konstanta dan selanjutnya dihitung menggunakan rumus *fitness*. Semakin besar nilai *fitness* yang dihasilkan maka solusi yang dihasilkan semakin baik sedangkan jika semakin kecil nilai *fitness*nya maka semakin buruk solusi yang dihasilkan.
4. Dalam menentukan parameter algoritma genetika yang digunakan pada penjadwalan perawat maka dilakukan pengujian parameter yang terdiri dari ukuran generasi, ukuran populasi (*popsize*) dan kombinasi *crossover rate* (*cr*) dan *mutation rate* (*mr*). Hasil dari pengujian merupakan parameter dengan nilai rata-rata *fitness* tertinggi dari percobaan yang dilakukan sebanyak 10 kali. Hasilnya didapatkan jumlah generasi 105,

jumlah populasi 300 dan kombinasi *crossover rate* dan *mutation rate* 0.6:0.4 yang memiliki nilai rata-rata *fitness* tertinggi.

5. Berdasarkan hasil pengujian maka didapatkan kesimpulan bahwa nilai parameter algoritma genetika berupa ukuran populasi, generasi dan kombinasi cr dan mr berpengaruh terhadap hasil optimasi. Ukuran parameter yang kecil akan menyebabkan area pencarian algoritma genetika semakin sempit sedangkan jika ukuran parameter terlalu besar maka akan membutuhkan waktu komputasi lebih lama dan tidak menjamin akan menghasilkan nilai yang optimal.

7.2 Saran

Pada penelitian tentang penjadwalan selanjutnya, agar dihasilkan solusi jadwal yang lebih optimal pengujian parameter dapat dilakukan lebih banyak lagi. Pemilihan parameter yang tepat diharapkan dapat menghasilkan solusi jadwal yang lebih optimal. Kombinasi metode *crossover*, mutasi atau seleksi lainnya dapat digunakan dalam menerapkan algoritma genetika untuk mendapatkan solusi yang lebih bervariasi dan lebih optimal dalam kasus penjadwalan perawat.

DAFTAR PUSTAKA

Atmasari. 2010. *Penjadwalan Perawat Unit Gawat Darurat Dengan Menggunakan Goal Programming*. Jurusan Matematika. Fakultas Matematika dan Ilmu Pengetahuan Alam. Institut Teknologi Sepuluh Nopember. Surabaya.

Faraji, Rasoul. 2014. *An Efficient Crossover Architecture For Hardware Parallel.* Department of Electrical and Computer Engineering, Graduate University of Advanced Technology, Haftbagh BLV, Kerman, Iran. Neurocomputing Journal 128 (2014) 316–327.

Faris, Muhammad Zaini. 2013. *Optimasi Penjadwalan Mata Pelajaran Menggunakan Genetic Algorithm.* Program Teknologi Informasi dan Ilmu Komputer. Universitas Brawijaya. Malang.

Kaya, Mustafa. 2008. *The Effects Of Two New Crossover Operators On Genetic Algorithm Performance.* Aksaray University, Faculty of Engineering, Adana Street, Aksaray, Turkey.

Mahmudy, Wayan Firdaus. 2013. *Algoritma Evolusi*. Program Teknologi Informasi dan Ilmu Komputer. Universitas Brawijaya. Malang.

Mahmudy, WF, Marian, RM & Luong, LHS 2013. *Modeling And Optimization Of Part Type Selection And Loading Problems In Flexible Manufacturing System Using Real Coded Genetic Algorithms.* International Journal of Electrical, Electronic Science and Engineering, vol. 7, no. 4, pp. 181-190.

Mahmudy, WF, Marian, RM & Luong, LHS. 2014. *Hybrid Genetic Algorithms For Part Type Selection And Machine Loading Problems With Alternative Production Plans In Flexible Manufacturing System*. ECTI Transactions on Computer and Information Technology (ECTI-CIT), vol. 8, no. 1, pp. 80-93.

Pitaloka, DA, Mahmudy, WF & Sutrisno. 2014. *Penerapan hybrid algoritma genetika untuk permasalahan vehicle routing problem with time windows (VRPTW)*. DORO: Repository Jurnal Mahasiswa PTIIK Universitas Brawijaya, vol. 4, no. 11.

Sabarulin, dkk. 2013. *Faktor Yang Mempengaruhi Kinerja Perawat Dalam Mendokumentasikan Asuhan Keperawatan Di Rumah Sakit Woodward Palu*. Jurnal AKK, Vol 2 No 3, September 2013, hal 29-34.

Sbeity, Ilhab, dkk. 2014. *Combining The Analytical HierarchyProcess And The Genetic Algorithm To Solve The Timetable Problem.* International Journal of Software Engineering & Applications (IJSEA), Vol.5, No.4, July 2014.

Sharma, Anshul, dkk. 2013. *Review Paper of Various Selection Methods in Genetic Algorithm.* International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 7, July 2013. ISSN: 2277 128X.

Sundarningsih, D, Mahmudy, WF & Sutrisno. 2015. *Penerapan Algoritma Genetika Untuk Optimasi Vehicle Routing Problem With Time Window (VRPTW) : Studi Kasus Air Minum Kemasan*. DORO: Repository Jurnal Mahasiswa PTIIK Universitas Brawijaya, vol. 5, no. 9.

Tjatjo, R A. 2008. *Optimasi Penjadwalan Daftar Jaga Perawat Menggunakan Algoritma Genetika Studi Kasus Pada Rumah Sakit Undata Palu*. Program Studi Ilmu Komputer. Jurusan Matematika. Fakultas Matematika dan Ilmu Pengetahuan Alam. Universitas Brawijaya. Malang.

Ulya, Anisa. 2010. *"Penggunaan Algoritma Genetik Dengan Pemodelan Dua Tingkat Dalam Permasalahan Penjadwalan Perawat Pada Unit Gawat Darurat Rumah Sakit Umum Xyz Surabaya"*. Jurusan Sistem Informasi. Fakultas Teknologi Informasi. Institut Teknologi Sepuluh November. Surabaya.

Wong, T.C. 2014. *A Two Stage Heuristic Approach For Nurse Scheduling Problem: A Case Study In An Emergency Department*. International Journal of Computers & Operations Research 51 (2014) 99–110.