PENERAPAN ALGORITMA GENETIKA UNTUK OPTIMASI PENJADWALAN JAGA SATPAM DI UNIVERSITAS BRAWIJAYA BERBASIS *SMS GATEWAY*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Zata Ismah
NIM: 125150200111108

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2016

PENGESAHAN

PENERAPAN ALGORITMA GENETIKA UNTUK OPTIMASI PENJADWALAN JAGA SATPAM DI UNIVERSITAS BRAWIJAYA BERBASIS SMS GATEWAY

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Zata Ismah
NIM: 125150200111108

Skripsi ini telah diuji dan dinyatakan lulus pada
8 Agustus 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Dian Eka Ratnawati, S.Si., M.Kom	Issa Arwani, S.Kom, M.Sc
NIP: 19730619 200212 2 001	NIP: 19830922 201212 1 003

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D

NIP: 19710518 200312 1 001

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 dalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang 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, 30 Juli 2016

Zata Ismah

NIM: 125150200111108

KATA PENGANTAR

Dengan menyebut nama Allah SWT Yang Maha Pengasih dan Maha Penyayang. Puji syukur kehadirat Allah SWT karena limpahan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan skripsi dengan judul "PENERAPAN ALGORITMA GENETIKA UNTUK OPTIMASI PENJADWALAN JAGA SATPAM DI UNIVERSITAS BRAWIJAYA BERBASIS SMS GATEWAY". Shalawat serta salam senantiasa tercurahkan kepada junjungan Nabi besar kita Nabi Muhammad SAW beserta keluarga dan para sahabatnya. Skripsi ini disusun untuk memenuhi salah satu syarat memperoleh gelar Sarjana Komputer di Fakultas Ilmu Komputer di Universitas Brawijaya Malang (Filkom UB).

Pada kesempatan ini penulis juga ingin menyampaikan terima kasih yang sebesar-besarnya kepada semua pihak yang telah memberikan bantuan dan dukungan dalam menyelesaikan skripsi ini, antara lain :

1. Ibu Dian Eka Ratnawati, S.Si., M.Kom, selaku Dosen Pembimbing skripsi satu yang telah meluangkan waktu, membimbing, mengarahkan dan memberikan saran kepada penulis sehingga dapat menyelesaikan skripsi ini.
2. Bapak Issa Arwani, S.Kom, M.Sc selaku Dosen Pembimbing skripsi dua yang telah meluangkan waktu, membimbing, mengarahkan dan memberikan saran kepada penulis sehingga dapat menyelesaikan skripsi ini.
3. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D selaku Ketua Jurusan Terknik Informatika Universitas Brawijaya
4. Bapak Agus Wahyu Widodo, S.T, M.Cs selaku Ketua Program Studi Teknik Informatika Universitas Brawijaya
5. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D, selaku ketua Fakultas Ilmu Komputer Universitas Brawijaya.
6. Seluruh Dosen FILKOM UB yang telah membagikan ilmunya kepada penulis selama masa perkuliahan.
7. Seluruh staff dan karyawan FILKOM UB yang telah banyak membantu dalam hal administrasi penulis dalam pelaksanaan penyusunan skripsi ini.
8. Kedua Orang Tua Penulis yaitu Syamsul Bahri, Yeniwati dan Kedua Saudara Penulis yaitu Ami, Tia, serta om In, om Nal, ante Aci, Ante Upik dan keluarga besar atas segala do'a, nasihat, dukungan baik moril maupun materiil yang begitu besar terhadap kelancaran dalam menyelesaikan skripsi ini.
9. Sahabat serta teman-teman penulis, Vendy, Reza, Julio, Dio, Pandu, Dea, Azza, Melly, Riza, Ainin, Dewi, Ria, Dini, serta seluruh teman-teman angkatan 2012, angkatan 2013, dan angkatan 2014 yang selalu memberikan bantuan, dukungan, motivasi dan berbagi informasi demi kelancaran skripsi.
10. Sahabat serta teman-teman penulis di Malang, Padang dan Samarinda yang selalu mendukung dan memotivasi demi kelancaran skripsi.

Dengan kerendahan hati, penulis menyadari bahwa skripsi ini masih memiliki banyak kekurangan. Oleh karena itu kritik dan saran yang bersifat konstruktif sangat dibutuhkan sebagai pedoman untuk menyempurnakan skripsi ini agar lebih baik. Penulis berharap semoga skripsi ini dapat bermanfaat bagi diri sendiri maupun bagi semua pihak.

Malang, 30 Juli 2016

Penulis

zatamah@gmail.com

ABSTRAK

Penjadwalan jaga satpam di Universitas Brawijaya masih dibuat secara manual sehingga dalam proses pembuatannya masih belum efisien. Selain itu kelompok satpam yang berjaga tidak ada variasi sehingga dalam proses penjagaan dapat membuat jenuh satpam dan dapat mempengaruhi kualitas kinerja. Dalam penelitian ini, diterapkan algoritma genetika untuk menyelesaikan permasalahan penjadwalan jaga satpam. Algoritma genetika sering kali digunakan dalam penyelesaian kasus penjadwalan. Representasi permutasi yang digunakan yaitu permutasi bilangan integer dengan panjang kromosomnya yaitu 1890. Angka-angka pada gennya mempresentasikan kode satpam. Metode crossover yang digunakan yaitu single-point crossover, metode mutasi yaitu reciprocal exchange mutation, dan metode seleksinya yaitu elitism. Dari hasil pengujian yang dilakukan diperoleh parameter optimal yaitu ukuran populasi (*popsize)* sebesar 1000, generasi sebesar 1000, *crossover rate (cr)* sebesar 0.4, dan *mutation rate (mr)* sebesar 0.9. Hasil akhir penelitian ini yaitu jadwal jaga satpam di Universitas Brawijaya selama satu bulan.

Kata kunci : Algoritma Genetika, Satpam, Penjadwalan

ABSTRACT

Security scheduling in University of Brawijaya is still made in manually so that manufacturing process was not efficient. In addition, the group of security stationed there was no variation so the process can make a saturated security and can affect the quality of the performance. In this research, applied genetic algorithms to solve scheduling problems of security. Genetic algorithms are often used in the completion of scheduling cases. Representation permutation permutation used is an integer with a length chromosomes are 1890. The figures of genes present security code. Crossover method used is single-point crossover, mutation method is reciprocal exchange mutation and selection method is elitism. From the test result has optimal parameter of popsize is 1000, generation is 1000, crossover rate is 0.4, and mutation rate is 0.9. The final result from this research is a month of security scedule in University of Brawijaya.

Keyword : *Genetic Algorithm, Security, Schedule*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

Dalam bab ini akan dibahas mengenai latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah dan sistematika penulisan skripsi.

1.1 Latar belakang

Penjadwalan shift kerja telah diatur dalam Undang-Undang Ketenagakerjaan (UUK) Pasal 79 ayat 2 huruf a yaitu Jika jam kerja di lingkungan suatu perusahaan atau badan hukum lainnya (perusahaan) ditentukan 3 (tiga) shift, pembagian setiap shift adalah maksimum 8 (delapan) jam per-hari, termasuk istirahat antar jam kerja . Shift kerja didefinisikan sebagai lamanya waktu yang dikerjakan oleh sekelompok pekerja dimana pekerja tersebut dapat memulai pekerjaannya ketika kelompok yang lainnya telah selesai (Londong, 2012).

Di Universitas Brawijaya, shift kerja jaga satpam terdiri dari shift pagi, siang, dan malam. Komponen yang diperlukan untuk mengatur jam kerjanya yaitu kode satpam, hari dan shift. Permasalahan yang sering dihadapi dalam pembagian jadwal satpam ini yaitu masing-masing kelompok memiliki anggota yang sama setiap harinya sehingga tidak adanya variasi anggota dalam penjagaannya dan dapat membuat jenuh serta pembagian shift kerjanya dilakukan masih secara manual. Untuk dapat menyelesaikan permasalahan penjadwalan satpam, dapat menggunakan Algoritma Genetika. Dalam permasalahan penjadwalan, algoritma genetika telah berhasil diterapkan sehingga algoritma genetika merupakan metode yang cocok dalam pembuatan jadwal jaga satpam.

Beberapa penelitian berhasil mengimplementasikan algoritma genetika untuk optimasi penjadwalan, salah satunya adalah membuat sebuah penjadwalan asisten praktikum dengan menggunakan algoritma genetika dimana algoritma genetika mendapatkan hasil yang mendekati optimum yaitu dengan menggunakan 50 populasi untuk ukuran populasi yang optimal dari hasil uji coba dan menggunakan 30 generasi untuk mendapatkan hasil optimal. Nilai *crossover rate* 0,5 dan nilai *mutation rate* 0,5 itu didapatkan dari nilai fitness terbaik yaitu 964 (Devi, 2015).Selain itu ada juga penelitian yang menggunakan algotima genetika pada permasalahan optimasi untuk penjadwalan perawat dengan menggunakan algoritma genetika dengan memberikan hasil mendekati optimum dengan ukuran populasi sebesar 200 individu dengan nilai rata-rata fitness sebesar 0,80094, dan jumlah generasi sebesar 150 generasi dengan rata-rata fitness sebesar 2,13674 dan nilai *crossover rate* adalah 0.5 dan *mutation rate* adalah 0.5 dengan nilai rata-rata fitness sebesar 3,4266 (Ilmi, 2015).

Menurut Pateghem algoritma genetika merupakan sebuah solusi yang dapat digunakan untuk memecahkan masalah yang kompleks dengan menggunakan teknik evolusi biologi yang didalamnya terdapat seleksi, *crossover* dan mutasi yang nantinya digunakan untuk mendapatkan sebuah nilai dan memberikan solusi yang terbaik (Devi, 2015).

Di samping pembuatan jadwal, terdapat teknologi yang mendukung dalam penyampaian informasi jadwal penjagaan tersebut salah satunya dengan menggunakan *SMS gateway*. *SMS gateway* digunakan untuk mengirimkan informasi jadwal penjagaan melalui sms yang telah diolah dengan menggunakan metode algoritma genetika. Jadwal tersebut dikirim dengan mengambil data di database sistem dan mengirimnya secara otomatis ke semua nomor telepon yang telah tersimpan. Tujuan digunakannya *SMS gateway* ini agar satpam tersebut dapat mengetahui jadwal jaganya selama satu bulan.

Penelitian kali ini mencoba mengimplementasikan algoritma genetika pada permasalahan penentuan jadwal jaga satpam berbasis *SMS gateway*. Penelitian ini diharapkan akan menghasilkan solusi yang terbaik dan sesuai kriteria serta mendapatkan hasil kombinasi kromosom yang optimal sebagai penentu komposisi jadwal satpam. Selain itu, dengan adanya *SMS gateway* diharapkan dapat meningkatkan efektifitas dan efisiensi dari penyampaian informasi jadwal jaga satpam tersebut.

1.2 Rumusan masalah

1. Bagaimana representasi kromosom untuk penjadwalan satpam dengan menggunakan algoritma genetika?
2. Bagaimana mengimplementasikan penjadwalan satpam dengan menggunakan Algoritma Genetika?
3. Bagaimana menentukan parameter algoritma genetika yang tepat?

1.3 Tujuan

Tujuan dari penelitian ini adalah untuk membuat sebuah sistem yang efisien yang mampu membuat penjadwalan satpam secara optimal.

Selain itu, tujuan khusus dari penelitian ini yaitu :

1. Membuat representasi kromosom dari data-data penjadwalan yang akan memberikan solusi dari permasalahan.
2. Mengimplementasikan penjadwalan satpam dengan menggunakan Algoritma Genetika agar mendapatkan hasil yang optimal .
3. Menentukan parameter algoritma genetika yang tepat.

1.4 Manfaat

Manfaat yang diperoleh dalam penulisan skripsi ini adalah mencari solusi terbaik dalam menentukan jadwal jaga satpam agar mendapatkan pembagian jadwal yang adil, serta mengacak anggota regunya untuk menghilangkan kejenuhan dalam penjagaan gerbang karena tidak adanya variasi anggota dalam kelompok.

1.5 Batasan masalah dan ruang lingkup penelitian

Batasan masalah dalam penelitian ini adalah :

1. Periode penjadwalan dibuat dalam satu bulan.
2. Data yang di gunakan merupakan data satpam di Universitas Brawijaya.
3. Jumlah shift kerja satpam terbagi menjadi shift pagi, siang dan malam.
4. Bahasa pemrograman yang digunakan yaitu bahasa java.

Sedangkan ruang lingkup dalam penelitian ini adalah satpam yang ada di Universitas Brawijaya Malang.

1.6 Sistematika pembahasan/laporan

Sistematika penulisan dalam skripsi ini sebegai berikut :

BAB I PENDAHULUAN

Bab 1 memuat latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah dan sistematika penulisan skripsi.

BAB II LANDASAN TEORI

Bab 2 beirisi tentang teori - teori yang mendasari dan mendukung penulisan penerapan algoritma genetika untuk optimasi penjadwalan jaga satpam di universitas brawijaya dengan menggunakan *SMS gateway* .

BAB III METODOLOGI PENELITIAN

Bab 3 berisi algoritma-algoritma yang digunakan dalam pembuatan sistem optimasi penjadwalan perawat menggunakan algoritma genetika.

BAB IV IMPLEMENTASI

Bab 4 membahas mengenai implementasi dari system dan juga *sourcecode* pada bagian-bagian yang penting.

BAB V PENGUJIAN DAN ANALISIS

Bab 5 berisi tentang proses pengujian dan hasis alisis dari pengujian sistem tersebut.

BAB VI PENUTUP

Bab 6 berisi kesimpulan dan saran. Kesimpulan didasarkan atas pengujian dan analisis yang dilakukan didalam proses penelitian.

BAB 2 TINJAUAN PUSTAKA DAN DASAR TEORI

Pada Bab 2 ini berisi teori dasar tentang shift kerja satpam, algoritma genetika dan struktur algoritma genetika dalam pengoptimasian jadwal jaga satpam.

2.1 Kajian Pustaka

Kajian pustaka dilakukan untuk menganalisa dan membandingkan beberapa penelitian sebelumnya yang berhubungan dengan metode algoritma genetika. Karena pada hasil penelitian ini nanti akan berbasis *SMS gateway*, maka penelitian ini juga menganalisa penelitian yang berhubungan dengan *SMS gateway*.

Penelitian pertama dilakukan oleh Ilmi (2015) yang melakukan penelitian dengan judul *Optimasi Penjadwalan Perawat Menggunakan Algoritma Genetika*. Penelitian tersebut menjelaskan tentang penggunaan algoritma genetika dalam penjadwalan perawat. *Input*-an yang digunakan yaitu jumlah populasi (*popsize), crossover rate (cr), mutation rate (mr),* dan jumlah generasi. Algoritma genetika digunakan untuk mencari nilai solusi terbaik dari setiap iterasi yang dilakukan. Hasil yang diperoleh dari penelitian tersebut adalah solusi terbaik dalam penjadwalan perawat yang mana penjadwalan tersebut dituntut untuk mendapatkan jadwal dengan beban kerja seadil mungkin untuk setiap perawat serta memenuhi batasan-batasan penjadwalan yang ada.

Penelitian selanjutnya dilakukan oleh Zulfa (2015) dengan judul *Optimasi Jadwal Mengajar Asisten Laboratorium Menggunakan Algoritma Genetika.* Tujuan dilakukannya penelitian tersebut adalah untuk meningkatkan efektifitas dan efisiensi dalam penyusunan jadwal mengajar asisten praktikum. Sama halnya dengan penelitian yang dilakukan oleh Rifqy Rosyidah Ilmi, *input*-an yang digunakan jumlah populasi (*popsize), crossover rate (cr), mutation rate (mr),* dan jumlah generasi. Hasil akhir dari penelitian tersebut adalah menampilkan hasil dari penjadwalan asisten yang didapatkan dari kromosom terbaik.

Penelitian selanjutnya tentang Optimasi Waktu *Body Repair* Mobil Menggunakan Algoritma Genetika yang dilakukan oleh Wieldan (2013). Tujuan dilakukannya penelitian tersebut karena waktu pengerjaan *body repair* mobil tidak sesuai dengan kesepakatan sehingga menimbulkan komplain dari konsumen dan dikenakan *cashback.* Hasil dari penelitian ini berupa optimasi waktu pengerjaan yang tepat berdasarkan uji cobanya.

Penelitian selanjutnya dilakukan oleh Priyadna dan Riasti (2013) dengan judul *Pembuatan Sistem Informasi Nilai Akademik Berbasis SMS Gateway Pada SMP Negeri 3 Pringkuku Pacitan.* Tujuan penelitiannya ini untuk memberitahukan nilai akademik atau hasil belajar siswa belajar di sekolah kepada orang tuanya dirumah melalui SMS. Terdapat basis data informasi sekolah yang diakses dan kemudian mengirim data ke telepon seluler.

Berdasarkan dari paparan penelitian yang telah dilakukan, maka penulis mengusulkan penelitian yang berjudul *Penerapan Algoritma Genetika Untuk*

Optimasi Penjadwalan Jaga Satpam Di Universitas Brawijaya Dengan Berbasis SMS gateway. Penelitian ini dibuat untuk mencari solusi terbaik dalam menentukan jadwal jaga satpam agar mendapatkan pembagian jadwal yang adil dan mengacak anggota regunya untuk menghilangkan kejenuhan dalam penjagaan gerbang karena tidak adanya variasi anggota regu. Sama halnya dengan penelitian sebelumnya, *input*-an yang digunakan dalam penelitian ini yaitu jumlah populasi (*popsize), crossover rate (cr), mutation rate (mr),* dan jumlah generasi. Ketika sistem telah selesai melakukan iterasi maka sistem akan mengirimkan jadwal jaga satpam tersebut ke semua satpam melalui *SMS gateway.*

Tabel 2.1 Kajian Pustaka

No.	Judul	Permasalahan	Metode	Hasil
1.	*Optimasi Penjadwalan Perawat Menggunakan Algoritma Genetika*	Membuat pembagian penjadwalan jaga perawat secara adil	Algoritma genetika. *Crossover* : *one cut point crossover,* Mutasi : *reciprocal exchange mutation,* Seleksi : *elitism selection.*	Jadwal jaga perawat selama 1 bulan.
2.	*Optimasi Jadwal Mengajar Asisten Laboratorium Menggunakan Algoritma Genetika*	Masih dibuat secara manual dengan permasalahan waktu yang bentrok antara jadwal kuliah dengan jadwal mengajar asisten praktikum, serta membutuhkan waktu yang lama.	Algoritma genetika. *Crossover* : *one cut point crossover* Mutasi : *reciprocal exchange mutation* Seleksi : *elitism selection*	Kromosom terbaik dengan waktu optimal, jumlah pinalti yang rendah dan fitness yang tinggi
3.	Optimasi Waktu *Body Repair* Mobil Menggunakan Algoritma Genetika	Waktu pengerjaan *body repair* mobil tidak sesuai dengan kesepakatan sehingga menimbulkan komplain dari	Algoritma Genetika. Crossover : *Position based Crossover* Mutasi : *reciprocal exchange mutation* Seleksi : *rank selection*	Uji coba dilakukan sebanyak 5 kali. Nilai fitness yaitu semakin kecil peluang *crossover* 10% dan peluang mutasinya 10%

		konsumen dan dikenakan *cashback.*		didapat nilai hasil fitness yang tidak beragam dan tidak maksimal. Hasil fitness Rp 10.178.868 adalah nilai fitness tertinggi yaitu dengan peluang *crossover* dengan titik 90% didapat pengujian dengan nilai terbaik.
4.	*Pembuatan Sistem Informasi Nilai Akademik Berbasis SMS Gateway Pada SMP Negeri 3 Pringkuku Pacitan*	Orang tua tidak mengetahui perkembangan nilai anak disebabkan kesibukannya.	SMS *Gateway*	Sistem dapat mengirimkan sms berupa perkembangan nilai siswanya ke nomor yang tersimpan di database.

2.2 Shift Kerja Satpam

Satuan Pengamanan (satpam) bertugas untuk menyelenggarakan keamanan swakarsa lingkungan kerjanya yang dibentuk oleh instansi/proyek/badan usaha. Pengaturan umum mengenai jadwal kerja (shift), jam kerja dan perhitungan upah kerja lembur bagi anggota Satuan Pengamanan (Satpam) pada prinsipnya merujuk pada pasal 77 dan pasal 78 UU No. 13 Tahun 2003 tentang Ketenagakerjaan (UU 13/2003). Dalam pasal 77 ayat (2) UU No. 13/2003 berisi tentang waktu kerja tenaga kerja tidak boleh lebih dari 40 jam dalam satu minggu dan tidak berlaku bagi sektor usaha atau pekerjaan tertentu. Jika pemimpin mempekerjakan tenaga kerja lebih dari 40 jam maka waktu tersebut dihitung lembur dan lembur paling banyak tiga jam dalam satu hari (pasal 78 ayat (2) UU No. 13/2003).

2.3 Algoritma Genetika

Salah satu tipe dari algoritma evolusi yang populer adalah algoritma genetika. Algoritma genetika pada dasarnya terinspirasi dari prinsip genetika dan seleksi

alam (teori evolusi Darwin) yang ditemukan oleh John Holland (1975) di Universitas Michigan, Amerika Serikat. Penelitian itu dipopulerkan oleh salah satu muridnya, David Goldberg. Berdasarkan teori Darwin, yaitu *survival of the fittest,* di alam terjadi persaingan antara individu-individuu untuk bertahan hidup sehingga makhluk yang paling kuat mendominasi makhluk yang paling lemah (Sutojo et al., 2011).

Dalam cikal bakal penggunaan algoritma genetika untuk pencarian dalam sistem buatan, beberapa ahli biologi menggunakan komputer digital untuk mengerjakan simulasi dari sistem genetika. Beberapa ahli tersebut diantaranya (Kuswadi, 2007) :

1. Baricelli, N.A. (1957), melakukan penelitian tentang proses evolusi simbiogenetik yang direalisasikan dengan metode artifisial.
2. Baricelli, N.A. (1962), mengajukan teori evolusi dan uji numeriknya.
3. Franser, A.S. (1960), menyimulasikan sistem genetika dengan komputer yang meliputi aspek-aspek S-linkage, dominasi dan epistasis.

Algoritma genetika digunakan untuk menyelesaikan sebuah masalah kompleks pada berbagai bidang, misalnya permasalahan yang terdapat pada bidang fisika, biologi, ekonomi, sosiologi dan lainnya (Mahmudy, 2013). Menurut Mahmudy, Marian & Loung (Mahmudy, 2013), penjadwalan dan perencanaan produksi menggunakan algoritma genetika dalam bidang industri manufaktur, merupakan selah satu contoh penerapan algoritma genetika.

Dalam algoritma genetika tidak selalu memberikan hasil yang terbaik tetapi algoritma genetika ini dapat memecahkan suatu permasalahan yang kompleks denga hasil yang mendekati optimal. Untuk bisa mendapatkan nilai baru dan memberikan solusi yang optimal, dalam memecahkan masalah menggunakan algoritma genetika menggunakan seleksi, *crossover* dan mutation.

Menurut Haupt & Haupt (Mahmudy, 2013), pada algoritma genetika terdapat beberapa kelebihan antara lain algoritma genetika dapat menyelesaikan masalah yang kompleks dengan banyak variabel serta algoritma genetika bisa melakukan pencarian tanpa memperoleh informasi yang spesifik dari masalah yang ada gengan menggunakan chomosome untuk melakukan pengkodean.

Goldberg (1989) mengemukakan bahwa algoritma genetika mempunyai karakteristik-karakteristik yang perlu diketahui sehingga dapat terbedakan dari prosedur pencarian atau optimasi yang lain yaitu: (Bridga, 2014)

1. Algoritma genetika bekerja dengan pengkodean dari himpunan solusi permasalahan yang berdasarkan pada parameter yang telah ditetapkan dan bukan dari parameter itu sendiri. Sebagai contoh, untuk mendapatkan nilai minimum dari fungsi $f(x)=y=x^4+2x^3+5$, algoritma genetika terlebih dahulu merepresentasikan x dalam bentuk string biner dan bukan secara langsung mencari nilai x atau y-nya.

2. Algoritma genetika melakukan pencarian bukan hanya pada sebuah individu melainkan pada sebuah populasi dari beberapa individu yang merupakan solusi dari permasalahan.
3. Algoritma genetika merupakan fungsi objektif *(fitness),* yaitu individu yang memiliki solusi terbaik akan diseleksi dan bukan turunan dari suatu fungsi.
4. Algoritma genetika bukan menggunakan aturan-aturan deterministrik melainkan aturan-aturan transisi peluang.

Ciri-ciri untuk menyelesaikan permasalahan yang membutuhkan algoritma genetika yaitu : (Sutojo et al., 2011).

1. Ruang pencariannya sangat besar, kompleks, dan kurang dipahami
2. Untuk menyederhanakan ruang pencarian yang sangat besar menjadi ruang percarian yang lebih sempit tidak memiliki pengetahuan yang memadai.
3. Tidak memiliki analisis matematis yang dapat menangani ketika metode konvensional tidak berhasil dalam menyelesaikan masalah yang dihadapi
4. Hasil dari solusinya tidak harus optimal, jika sudah memenuhi kriteria sudah bisa diterima
5. Memiliki kemungkinan yang jumlah solusinya tak hingga
6. Membutuhkan solusi *"real-time"*, yaitu solusi yang bisa didapatkan dengan cepat sehingga bisa diimplementasikan pada permasalahan yang memiliki perubahan yang cepat

Dalam penerapan algoritma genetika, melibatkan beberapa parameter yang mana parameter tersebut akan menentukan kesuksesan suatu optimasi. Adapun parameter yang terdapat dalam algoritma genetika diantaranya (Desiani & Arhami, 2006) :

1. Fungsi *fitness* (fungsi tujuan) yang dimiliki oleh setiap individu dapat menentukan tingkat kesesuaian individu tersebut berdasarkan kriteria yang ingin dicapai.
2. Populasi jumlah individu yang dilibatkan dalam setiap generasi
3. Probabilitas terjadinya persilangan (*crossover*) pada suatu generasi
4. Probabilitas terjadinya mutasi pada masing-masing individu.
5. Jumlah generasi yang akan dibentuk dapat menentukan lama dari penerapan algoritma genetika

2.3.1 Pengertian Dasar Algoritma Genetika

Dibawah ini adalah beberapa pengertian dasar yang perlu diketahui tentang algoritma genetika, diantaranya (Sutojo et al., 2011) :

1. Gen (Genotype) merupakan variabel dasar yang bernilai binner, float, integer, maupun karakter yang membentuk suatu kromosom.
2. Allele merupakan nilai dari suatu gen yang berupa binner, float, integer, maupun karakter.
3. Kromosm merupakan kumpulan atau gabungan dari bebrapa gen yang membentuk arti tertentu. Kromosom terdiri dari beberapa jenis, diantaranya :
   a. Kromosom biner adalah kromosom yang disusun dari beberapa gen yang bernilai binner yang memiliki tingkat keberhasilan yang tinggi.
   b. Kromosom float adalah kromosom yang disusun dari beberapa gen yang bernilai pecahan, termasuk gen yang bernilai bulat serta tingkat keberhasilan kromosom ini rendah dalam kecepatan (jumlah generasi).
   c. Kromosom string adalah kromosom yang disusun dari beberapa gen yang bernilai string.
   d. Kromosom kombinatorial adalah kromosom yang disusun dari beberapa gen yang nilai berdasarkan urutannya.
4. Individu merupakan kumpulan gen (hampir sama dengan kromosom). Individu menyatakan salah satu kemungkinan solusi dari suatu permasalahan.
5. Populasi merupakan sekumpulan individu yang akan diproses secara bersamaan dalam satu siklus proses evolusi.
6. Generasi merupakan satu satuan siklus proses evolusi.
7. Nilai Fitness merupakan nilai yang menjadi acuan untuk mencapai nilai yang optimal. Nilai tersebut dilihat dari seberapa baik nilai dari suatu individu atau solusi yang didapatkan.

2.3.2 Kekurangan dan Kelebihan Algoritma Genetika

Algoritma genetika memiliki beberapa kekurangan dalam penggunaannya, diantaranya (Berlianty & Arifin, 2010) :

1. Algoritma genetika menggunakan bilangan yang random atau acak, sehingga memungkinkan kromosom yang baik tidak ikut dalam proses tersebut.
2. Algoritma genetika menggunakan bilangan random satau atau acak, hal itu menyebabkan kemungkinan bahwa kromosom yang baik tidak ikut terproses.
3. Algoritma genetika menggunakan bilangan random atau acak pada setiap pemilihan kromosom dalam proses mutasi dan *crossover.*

4. Algoritma genetika yang menggunakan bilangan random atau acak belum tentu menghasilkan solusi yang optimal karena tergantung bilangan yang digunakan.

Selain memiliki kekurangan, algoritma genetika juga memiliki beberapa kelebihan dalam penggunaannya, diantaranya (Berlianty & Arifin, 2010) :

1. Algoritma genetika menggunakan manipulasi kode-kode set parameter algoritma.
2. Algoritma genetika bebas untuk menggunakan kode masalah dengan berbagai cara.
3. Algoritma genetika menggunakan populasi titik.
4. Algoritma menggunakan informasi fungsi tujuan.
5. Algoritma genetika menggunakan aturan probabilistik.
6. Algoritma genetika relatif banyak melakukan iterasi secara berulang-ulang, sehingga untuk menyelesaikan masalah tersebut perlu suatu sistem program aplikasi komputer.

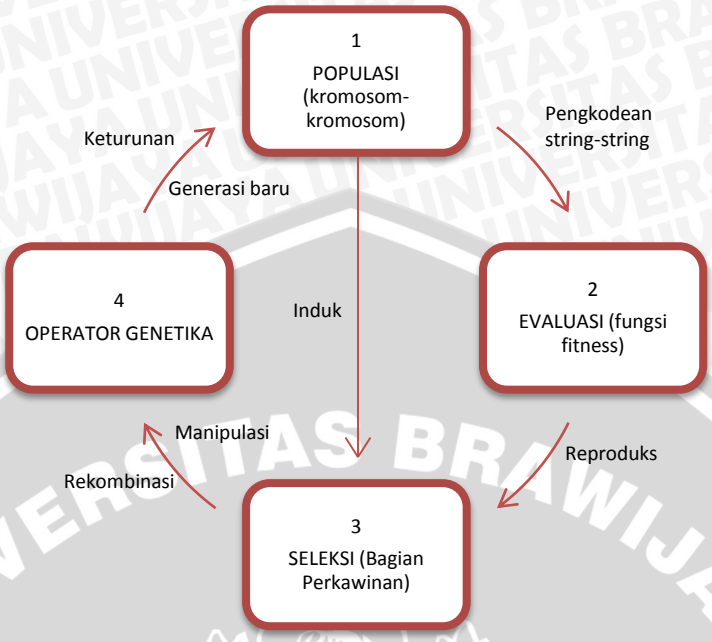
2.4 Struktur Algoritma Genetika

Dalam algoritma genetika terdapat populasi yang merupakan teknik pencarian atas beberapa solusi. Solusi dari suatu masalah direpresentasikan menjadi string chromosome. String chromosome tersusun dari beberapa gen yang menggambarkan variabel-variabel keputusan yang digunakan dalam sebuah solusi.

Di dalam algoritma genetika diawali dengan proses inisiaisasi yang berarti menciptakan individu-individu secara acak yang memiliki susunan gen kromosom tertentu. Dalam menentukan populasi, pada populasi awal dibuat secara acak dan populasi selanjutnya merupakan generasi yang didapatkan dari hasil evolusi kromosome menggunakan iterasi. Kemudian menggunakan nilai fitness untuk mengevaluasi kromosome yang ada pada setiap generasi. Nilai fitness yang diperoleh menunjukkan kualitas kromosom dalam populasi tesebut. Semakin besar nilai fitness yang diperoleh maka semakin baik chromosome tersebut untuk dijadikan calon solusi. Kemudian melakukan proses reproduksi untuk menghasilkan keturunan (offsring) yang terbentuk dari 2 kromosom generasi yang berperan sebagai parent dengan menggunakan penyilangan (*crossover*). Selain menggunakan *crossover* untuk menghasilkan keturunan dapat juga menggunakan mutasi. Seleksi dilakukan untuk memilih mana individu yang dipertahankan pada generasi selanjutnya dari populasi dan offspring. Fungsi probabilitis mempunyai fungsi untuk memilih individu yang dipertahankan hidup. Individu yang baik dengan nilai fitness yang besar mempunyai peluang untuk terpih (Mahmudy, 2013).

Setelah melewati iterasi maka diperoleh individu terbaik. menurut Michalewicz (Mahmudy, 2013) bisa diambil kesimpulan bahwa algoritma genetika dapat menghasilkan solusi yang optimum dengan melakukan pencarian di sejumlah alternatif titik optimum berdasarkan fungsi probanbilistik.

Dibawah ini merupakan gambaran siklus algoritma genetika (Kuswadi, 2007).



Gambar 2.1 Siklus Algoritma Genetika

Keterangan Gambar 2.1 :

1. Membangun sebuah "*populasi*" yang terdiri dari beberapa string.
2. Evaluasi masing-masing string (*fitness value).*
3. Proses seleksi agar didapat string yang terbaik.
4. Manipulasi genetika untuk menciptakan populasi baru dari string.

2.4.1 Representasi Kromosom

Representasi merupakan hasil akhir dari suatu permasalahan yang ingin diselesaikan. Representasi kromosom atau encoding, biasanya direpresentasikan dengan bilangan real, decimal atau biner (Panharesi, 2015). Untuk menghasilkan solusi yang sesuai, diperlukan representasi kromosom dalam perhitungannya. Secara permutasi, representasi kromosom juga dapat menggunakan (Panharesi, 2015):

- Bilangan real (*real-number encoding)*, nilai gen berada dalam interval [0,R]. Biasanya nilai R = 1 dan R merupakan bilangan real positif.
- Bilangan decimal *(discrete decimal encoding),* nilai gen berada pada interval [0,9] dan nilai gen biasanya berupa deretan bilangan bulat.
- Bilangan biner *(binary encoding),* nilai gen berupa angka 1 dan 0.
- Representasi permutasi *(permutation encoding),* bisa digunakan untuk menyatakan solusi.

2.4.2 Inisialisasi

Inisialisasi biasanya digunakan untuk membangkitkan himpunan solusi baru secara acak dan terdiri dari sejumlah string kromosom serta ditempatkan pada penampungan yang disebut populasi (Mahmudy, 2014). Dalam tahap ini harus ditentukan terlebih dahulu ukuran populasinya *(popsize)* yang berfungsi untuk membatasi perhitungan agar cakupannya tidak terlalu besar. Ukuran populasi merupakan banyaknya kromosom (individu) dalam satu populasi. Jika dalam suatu populasi jumlah kromosomnya teralu sedikit maka algoritma genetika memiliki kemungkinan yang sedikit untuk melakukan proses *crossover* dan hanya sebagian kecil dari ruang pencarian yang akan dieksplorasi (dijelajahi). Sebaliknya, jika jumlah kromosom terlalu banyak maka algoritma genetika cenderung menjadi lambat dalam menemukan solusi (Desiani & Arhami, 2006).

2.4.3 Reproduksi

Reproduksi merupakan suatu proses untuk menghasilkan keturunan dari individu-individu yang ada di populasi (Mahmudy, 2014). Individu-individu tersebut akan dipilih sebagai parent yang mana nanti akan menghasilkan keturunan yang disebut *offspring.* Ada dua operator genetika yang biasa digunakan dalam proses reproduksi yaitu *crossover* atau tukar silang dan *mutation.*

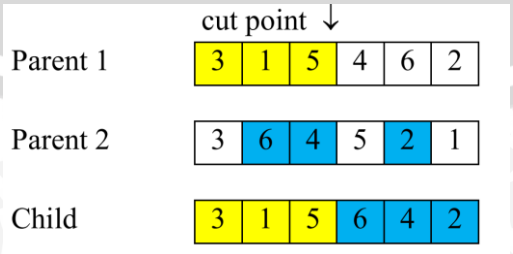
2.4.3.1 Crossover

Crossover atau tukar silang merupakan suatu operator genetika yang digunakan dalam proses reproduksi dengan perkawinan silang antara satu atau lebih kromosom induk yang terpilih sehingga menghasilkan satu atau lebih *offspring* (Panharesi, 2015). Nilai *offspring* diperoleh dari perkalian antara *crossover rate* dengan *popsize* atau dapat dirumuskan pada persamaan 2-1.

$$offspring = crossover\ rate * popsize \quad \text{..... 2-1}$$

Misalnya diketahui cr = 0.2 dan *popsize* = 10, maka *offspring* yang dihasilkan adalah 0.2 x 10 = 2 *offspring.*

Salah satu jenis *crossover* yang digunakan dalam penelitian ini yaitu modifikasi *one-cut-point crossover.* Modifikasi *one-cut-point crossover* biasanya digunakan pada representasi biner. Contoh metode modifikasi *one-cut-point crossover* dapat dilihat pada Gambar 2.2.



Gambar 2.2 Ilustrasi Modifikasi *One-Cut-Point Crossover*

Pada Gambar 2.2, segmen kiri pada kromosom child didapatkan dari parent 1 dan segmen kanan didapatkan dari urutan gen tersisa dari parent 2 (Mahmudy, 2014).

2.4.3.2 Mutation

Mutation atau mutasi merupakan suatu operator genetika yang digunakan dalam proses reproduksi dengan merubah secara acak nilai gen pada kromosom (Panharesi, 2015). Sama halnya dengan *crossover,* untuk memperoleh nilai *offspring* maka lakukan perkalian antara *mutation rate* dengan *popsize* atau dapat dilihat pada persamaan 2-2*.* Umumnya pada mutasi akan dipilih satu individu sebagai induk untuk menghasilkan satu *offspring.*

$$offspring = mutation\ rate * popsize \quad \text{..... 2-2}$$

Metode mutasi yang paling sederhana yaitu *reciprocal exchange mutation.* Cara kerja metode ini dengan memilih dua posisi (*exchange point / XP)* secara random kemudian menukarkan nilai pada posisi tersebut (Mahmudy, 2013). Contoh perhitungan mutasi dengan metode *exchange point* terdapat dalam penelitian sebelumnya, yaitu *Penerapan Algoritma untuk Penjadwalan Asisten Praktikum* (Devi, 2015). Proses mutasi yang dilakukan adalah sebagai berikut.

Kromosom sebelum mutasi :

Gen ke

1	2	3	4	5	6	7	8	9	10
1	1	2	2	3	3	4	4	5	5

Kromosom setelah mutasi :

Gen ke

1	2	3	4	5	6	7	8	9	10
1	1	2	2	3	3	4	4	5	5

Gambar 2.3 Ilustrasi Mutasi

Pada Gambar 2.3 merupakan ilustrasi mutasi, nilai yang ditukar nilai dari gen ke 3 dan gen ke 9.

2.4.4 Evaluasi

Evaluasi merupakan proses perhitungan nilai *fitness* tiap kromosomnya (Panharesi, 2015). Dalam dunia nyata, evolusi pada individu yang memiliki nilai *fitness* tertinggi akan bertahan hidup, sedangkan individu yang *fitness*nya bernilai rendah akan mati. Untuk masalah optimasi, jika solusi yang dicari adalah memaksimalkan sebuah fungsi *h*, maka nilai *fitness* yang digunakan adalah nilai dari fungsi *h* tersebut, yakni *f = h*. Namun jika masalahnya meminimalkan fungsi *h*, maka fungsi *h* tidak dapat digunakan secara langsung. Nilai *fitness* untuk masalah ini yaitu *f = 1/h,* artinya semakin kecil nilai *h* semakin besar nilai *f.* Jika *h* bernilai

0, maka fungsi ini akan bermasalah yang mengakibatkan f bernilai tak terhingga. Oleh karena itu, *h* perlu ditambah dengan sebuah bilangan yang sangat kecil. Sehingga rumus *fitness* tersebut ditunjukkan pada persamaan (2-3) (Suyanto, 2011).

$$f = \frac{1}{(h+a)} \quad \text{2-3}$$

Dimana :

- *a* adalah bilangan yang dianggap sangat kecil dan bervariasi dengan masalah yang akan diselesaikan.
- *h* adalah fungsi yang akan diminimalkan.
- *f* adalah fungsi *fitness.*

Nilai *fitness* adalah nilai yang dapat menyatakan solusi yang ada apakah sudah baik atau tidak. Nilai fitness ini diperoleh dari permasalahan yang ada. Semakin besar *fitness* maka semakin baik kromosom tersebut untuk dijadikan calon solusi (Mahmudy, 2013a). Dalam penelitian ini, untuk mendapatkan nilai *fitness* dilakukan perhitungan penalti. Penalti merupakan pelanggaran yang tidak sesuai dengan aturan. Jenis penalti dibagi menjadi dua yaitu *soft constraint* dan *hard constraint. Soft constraint* yaitu batasan tambahan yang biasanya sebuah permintaan sedangkan *hard constraint* yaitu batasan yang tidak boleh dilanggar atau harus dipenuhi dalam menyelesaikan suatu masalah. Setiap pelanggaran yang dilakukan dihitung satu. Nilai pelanggaran dihitung berdasarkan kemunculannya pada tiap kromosom.

2.4.5 Seleksi

Seleksi merupakan suatu proses untuk memilih individu dari himpunan populasi dan *offspring* yang dipertahankan hidup pada generasi berikutnya (Mahmudy, 2014). Seleksi akan dilakukan dengan cara mengambil kromosom sejumlah populasi awal. Pada umumnya, individu yang terpilih yaitu individu yang memiliki nilai *fitness* terbesar pada kromosomnya. Fungsinya agar generasi selanjutnya lebih baik dari generasi sebelumnya. Macam-macam metode pada proses seleksi yaitu *roulette wheel, binary tournament,* dan *elitism*.

Seleksi *elitism* yaitu peng-kopi-an individu terbaik untuk dimasukkan sebagai anggota populasi pada generasi berikutnya (Suyanto, 2011). Individu terbaik diperoleh dari nilai *fitness* tertinggi. Individu yang bertahan hidup pada generasi berikutnya dikumpulkan sebanyak *popSize*nya. Pada seleksi elitism menjamin individu yang terbaik akan selalu lolos pada generasi berikutnya.

2.5 SMS *Gateway*

SMS gateway adalah *software* yang digunakan sebagai gerbang penghubung antara sistem komputer dengan SMS Center dari operator seluler (Astuwasito, 2012). *SMS gateway* ini dapat mengirimkan pesan ke ratusan nomor yang ada di dalam database secara otomatis. Keunggulan dalam layanan *SMS* diantaranya :

1. Biaya terjangkau dan pengiriman pesan terjamin sampai ke nomor tujuan jika nomor tersebut masih aktif digunakan.
2. Layanan SMS sangat mudah digunakan.
3. Dapat mengirimkan pesan dimanapun dan kapanpun.

Dalam pembangunan sistem berbasis *SMS gateway* banyak aplikasi yang menyediakannya. Salah satu aplikasi yang sering digunakan dalam pembuatan SMS *gateway* ini yaitu Gammu.

2.5.1 Gammu

Gammu berfungsi untuk mengembangkan applikasi *SMS gateway*. Gammu merupakan pustaka *SMS gateway server* yang diciptakan oleh seorang programmer phyton berkebangsaan Jerman yang bernama Micar Cihar. Aplikasi ini sangat mudah diimplementasikan dan juga gratis. Selain itu, *gammu* memiliki beberapa kelebihan dibandingkan dengan aplikasi *SMS gateway* lainnya (Muhadkly, 2007):

1. *Gammu* bisa dijalankan di *Windows* maupun *Linux*
2. Banyak *device* yang kompatibel oleh *gammu*
3. *Gammu* menggunakan *database MySQL*
4. Baik kabel data USB maupun SERIAL, semuanya kompatibel di *Gammu*

Untuk menggunakan aplikasi *Gammu,* peralatan dan aplikasi yang harus dipersiapkan diantaranya :

1. *Gammu* untuk W*indows*
2. *Xampp*
3. Hp atau modem GSM
4. Driver HP atau modem

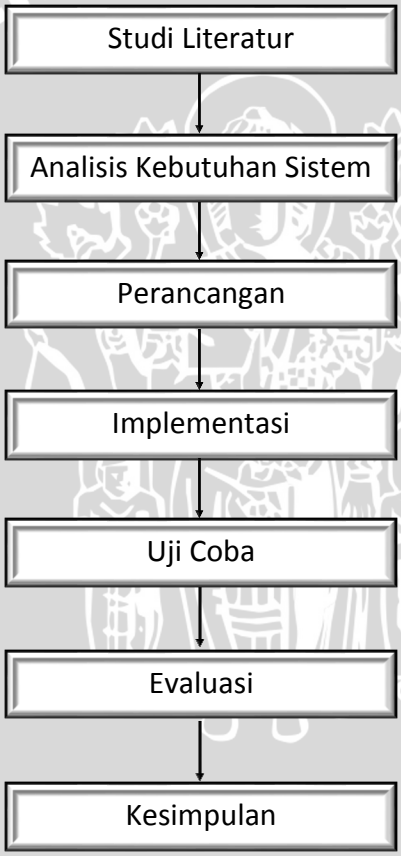
Untuk menggunakannya, *Gammu* dan *Xampp* harus dipasang terlebih dahulu di *Windows.* Hal pertama yang perlu dilakukan dalam proses pemasangan *Gammu* yaitu dengan mengekstrak folder *Gammu*. Di dalam *Gammu* terdapat berkas gammurc dan smsdrc. Pada port yang ada pada berkas gammurc (**PORT=COM**) harus diubah sesuai dengan *port* yang digunakan oleh modem atau hp. pada berkas smsdrc, baris *user, password, pc,* dan *database* harus diubah sesuai dengan *database* yang digunakan. Tuliskan perintah "gammu –identify" pada *command prompt* untuk memeriksa apakah modem dan HP sudah bisa digunakan oleh *Gammu.* SMS *service* dapat diaktifkan dengan menuliskan perintah "gammu --smsd MYSQL smsdrc" pada *command prompt*. Jika *sms service* sudah jalan, mulai mengirim SMS ke no. lain dengan cara insert data ke tabel outbox, jika sudah terkirim, sms akan pindah dari tabel outbox ke tabel sent items (Muhadkly, 2007)

BAB 3 METODOLOGI PENELITIAN

Bab ini membahas tentang metodologi yang digunakan dalam penelitian penjadwalan jaga satpam dan menjelaskan rancangan sistem yang dikembangkan. Tujuannya agar penelitian ini memiliki kegiatan penelitian yang terstruktur dan tepat serta perancangan yang baik.

3.1 Tahapan Penelitian

Metodologi penelitian yang dilakukan dalam penelitian ini yaitu studi literatur, metode pengambilan data, analisis kebutuhan, perancangan, implementasi, uji coba dan evaluasi. Tahapan-tahapan dalam penelitian tersebut dapat diilustrasikan dengan diagram blok metodologi penelitian seperti pada **Gambar 3.1** berikut ini.



Gambar 3.1 Diagram Blog Metodologi Penelitian

3.2 Studi Literatur

Studi literatur berisikan tentang sumber-sumber yang dapat dijadikan sebagai landasan ilmu dan dasar teori yang terkait dengan penelitian ini yaitu *Algoritma Genetika Untuk Optimasi Penjadwalan Jaga Satpam Di Universitas Brawijaya*

Dengan Berbasis SMS gateway. Literatur yang digunakan diambil dari buku, internet dan jurnal yang terkait dengan :

1. Algoritma Genetika
2. *SMS gateway*

3.3 Analisis Kebutuhan Sistem

Dalam pembuatan aplikasi penerapan algoritma genetika untuk optimasi penjadwalan jaga satpam berbasis *SMS gateway* memerlukan kebutuhan, baik kebutuhan perangkat keras maupun kebutuhan perangkat lunak serta kebutuhan data yang diperlukan dalam penelitian ini. Kebutuhan tersebut meliputi :

1. Kebutuhan Hardware
   - Komputer PC
   - Modem
2. Kebutuhan Software
   - Microsoft office 2013 sebagai perhitungan manualnya
   - Netbeans sebagai text editor pembuata source code program
   - XAMPP sebagai penghubung antara netbeans dan MySQL
   - MySQL sebagai tempat penyimpanan database
   - Gammu untuk menjembatani / mengomunikasikan antara database *SMS gateway* dengan sms devices
3. Data yang dibutuhkan

   Data yang dibutuhkan dalam sistem ini yaitu jumlah satpam, jumlah kelompok dan jumlah shift dari data satpam yang ada di Universitas Brawijaya

3.4 Perancangan

Perancangan bertujuan untuk mengetahui komponen yang diperlukan dalam mengimplementasikan algoritma genetika dalam penjadwalan jaga satpam. Proses perancangan terdiri dari *crossover,* mutasi, perhitungan pinalti dan *fitness*, serta seleksi. Tujuan dari pembuatan proses perancangan ini agar mendapatkan nilai yang optimal.

3.5 Implementasi

Implementasi penjadwalan jaga satpam ini dilakukan sesuai dengan perancangan yang telah dibuat.

3.6 Uji Coba

Pengujian dilakukan setelah proses implementasi selesai dilakukan. Langkah ini bertujuan untuk menguji hasil kerja sistem sudah sesuai atau belum dan juga

untuk mengetahui seberapa baik metode yang digunakan dalam menyelesaikan permasalahan ini.

3.7 Evaluasi

Langkah evaluasi ini bertujuan untuk mengetahui nilai parameter metode yang terbaik dalam penjadwalan jaga satpam.

3.8 Kesimpulan

Kesimpulan digunakan sebagai laporan akhir bagaimana kinerja sistem tersebut.

BAB 4 ANALISIS DAN PERANCANGAN

Pada bab ini akan dibahas proses perancangan algoritma untuk memperoleh keluaran yang terbaik.

4.1 Formulasi Permasalahan

Permasalahan dalam penelitian ini adalah untuk mengoptimasi penjadwalan jaga satpam dengan menggunakan algoritma genetika. Penyebab masalah ini yaitu karena penjadwalan yang masih dibuat secara manual dan kelompok penjaganya tidak pernah diganti sehingga menimbulkan kejenuhan dalam penjagaan karena tidak ada variasi kelompoknya. Oleh karena itu, untuk mengatasi permasalahan tersebut diperlukan sebuah sistem optimasi penjadwalan jaga satpam menggunakan algoritma genetika.

Data satpam diperoleh dari pos satpam yang ada di Universitas Brawijaya Malang. Data satpam yang akan diselesaikan pada penelitian ini ada sebanyak 82 orang. Berikut ini daftar nama satpamnya.

Tabel 4.1 Data Satpam

Kode Satpam	Nama Satpam
1	Ach Darussalam
2	Ach Lutfi Wal Aman
3	Ach Nur Agus Junaidi
4	Ach Sulthoni
5	Adi Nugroho
6	Adi Sulistiono
7	Agung Santoro
8	Agung Susianto
9	Agus Hary Utomo
10	Agus Minardi
11	Agus Santoso
12	Agus Widiarto
13	Anang Faisol
14	Anang Wahono
15	Andy Handoko
16	Antok Mardi Cahyanto
17	Ari Setiawan
18	Arif Margono
19	Aris Rumpoko
20	Ariyanto
21	Bambang Tri Winarso
22	Bogy Evry Sandy
23	Choirul Anwar
24	Daniel Topic Hambudi
25	Darmanto
26	Denis Prasetyo
27	Devry Mada Sandy
28	Didik Siswanto
29	Edi Santoso
30	Endik Eko Harianto
31	Fadjar Adi Candra
32	Feri Irawan
33	Ferry Indra Hardiyan B
34	Gathot Suyono
35	Gatot Aryo Subroto
36	Gelombang Taufan F
37	Gianto
38	Gunawan Budi H
39	Hadi Setyajaya
40	Hartoni
41	Hendra Agus T
42	Hendro Lasmintarto
43	Herman Lukito
44	Heru Priambodo
45	Ifan Fauzi
46	Ilham Bachtiar
47	Imam Fauzi
48	Imam Safi'i
49	Iwan Setiawan

50	Khoirul Anwar
51	M Erfan
52	M Riyan Finarko
53	M. Hamdani Wicaksono
54	Machmud Zazuli
55	Mardi Utomo
56	Marjoko
57	Mashudi
58	Mat Rois
59	Moch Asiyanto
60	Moch Haryono
61	Moch Nirmawan
62	Moch Usman Hadi
63	Mukti Waluyo
64	Mulyadi
65	Ngaderi
66	Nurul Hidayat

67	Piryono
68	Purbuwono
69	Ragil Sukma Perdana
70	Rioh Rully Kurniawan
71	Ronny Septian S
72	Rudi Eko Maryanto
73	Safak Fauzi
74	Sariadi
75	Sayudi
76	Sudarmanto
77	Sumantri
78	Sunar Widodo P
79	Syambodo Rachman
80	Teguh Imam Sujono
81	Yuantono Arif
82	Yusri Anzah

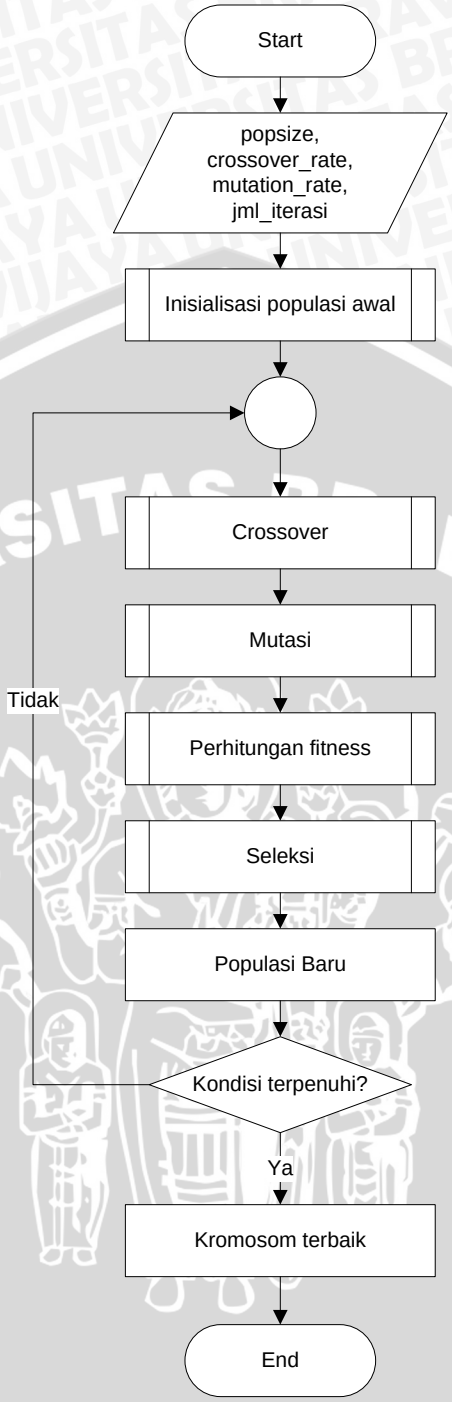
Data diatas nantinya akan dijadwalkan selama 30 hari. Selain itu jumlah shift dan jam penjagaan satpam ditunjukkan pada tabel 4.2 berikut ini :

Tabel 4.2 Jadwal Shift Jaga

No.	Shift	Jam Penjagaan
1	Pagi	06.00 - 14.00
2	Siang	14.00 - 22.00
3	Malam	22.00 - 06.00

4.2 Siklus Penyelesaian Masalah

Proses penyelesaian masalah jadwal penjagaan satpam menggunakan algoritma genetika yang diawali dengan inisialisasi populai awal, reproduksi, evaluasi serta seleksi. *Flowchart* algoritma genetika dapat dilihat pada Gambar 3.2.



Gambar 4.1 *Flowchart* Algoritma Genetika

1. Inisialisasi parameter awal, dimana parameter algoritma genetikanya meliputi p*opsize* (jumlah opulasi), *cr (crossover rate), mr (mutation rate)* dan jumlah generasi.
2. Mengacak nilai dari populasi awal berdasarkan jumlah populasinya.

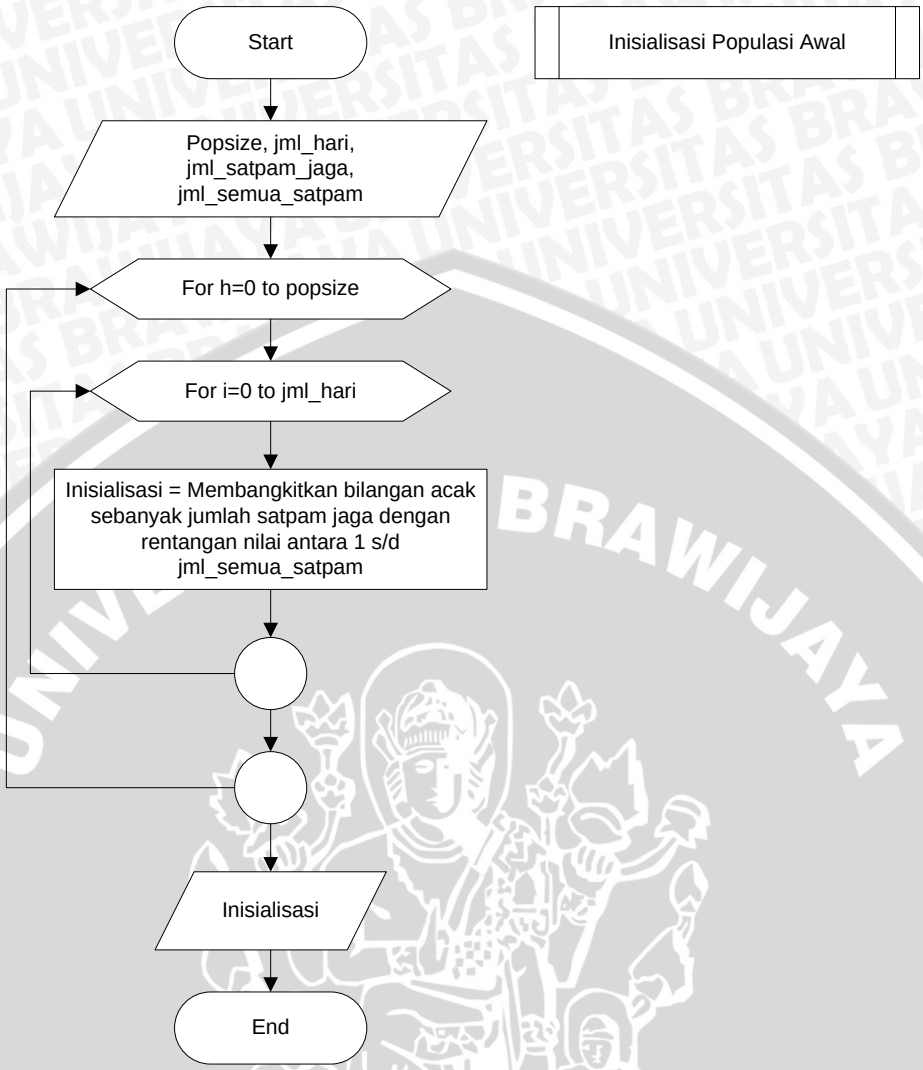
3. Melakukan proses *crossover* menggunakan metode *single point crossover* dengan cara memilih dua induk agar menghasilkan anak (*offspring).*
4. Melakukan proses mutasi menggunakan metode *reciprocal exchange mutation* dengan memilih dua gen secara acak pada induk terpilih dan menukarkan gen tersebut.
5. Menghitung nilai *fitness* dari masing-masing individu.
6. Melakukan seleksi *elitism selection* sebanyak jumlah *popsize* dengan mengambil individunya berdasarkan nilai fitness terbesarnya.
7. Ketika kondisi akhir telah terpenuhi, maka *iterasi* akan berhenti dan menghasilkan solusi terbaik pada generasi tersebut, namun jika kondisi akhir masih belum terpenuhi maka *iterasi* masih akan terus berjalan sampai menemukan solusi terbaiknya.

4.2.2 Representasi Kromosom

Representasi yang digunakan dalam pembuatan jadwal jaga satpam ini adalah representasi permutasi. Representasi permutasinya ini berupa bilangan bulat. Dalam pengkodean representasi kromosom dilakukan dalam bentuk array dimana dalam pengkodeannya berisi kode satpam.

4.2.3 Inisialisasi Populasi Awal

Pada proses ini sistem akan membangkitkan populasi awal secara acak berdasarkan jumlah populasi (*posize*). *Popsize* digunakan untuk menentukan jumlah individu atau kromosom. Sebuah individu atau kromosom merupakan sebuah solusi dari jadwal. Hal ini dilakukan untuk mencari penyelesaian yang optimal. Flowchart dari inisialisasi populasi awal dapat dilihat pada Gambar 4.2.



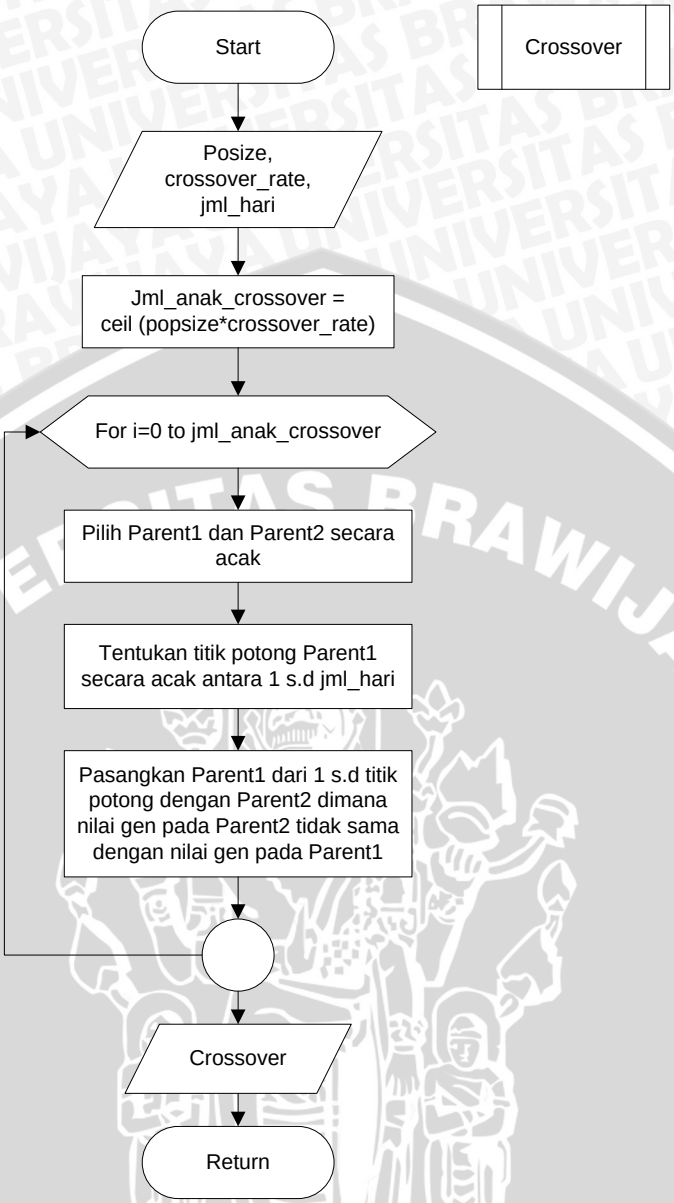
Gambar 4.2 *Flowchart* Inisialisasi Populasi Awal

4.2.4 Reproduksi

Reproduksi dilakukan untuk menghasilkan keturunan (*offspring)* dari individu-individu yang ada pada populasi (Mahmudi, 2013). Individu-individu ini nantinya akan berevolusi. Individu yang mampu beradaptasi dengan lingkungannya akan bertahan hidup pada generasi selanjutnya. Pada proses reproduksi terdapat *crossover* dan mutasi.

4.2.4.1 Crossover

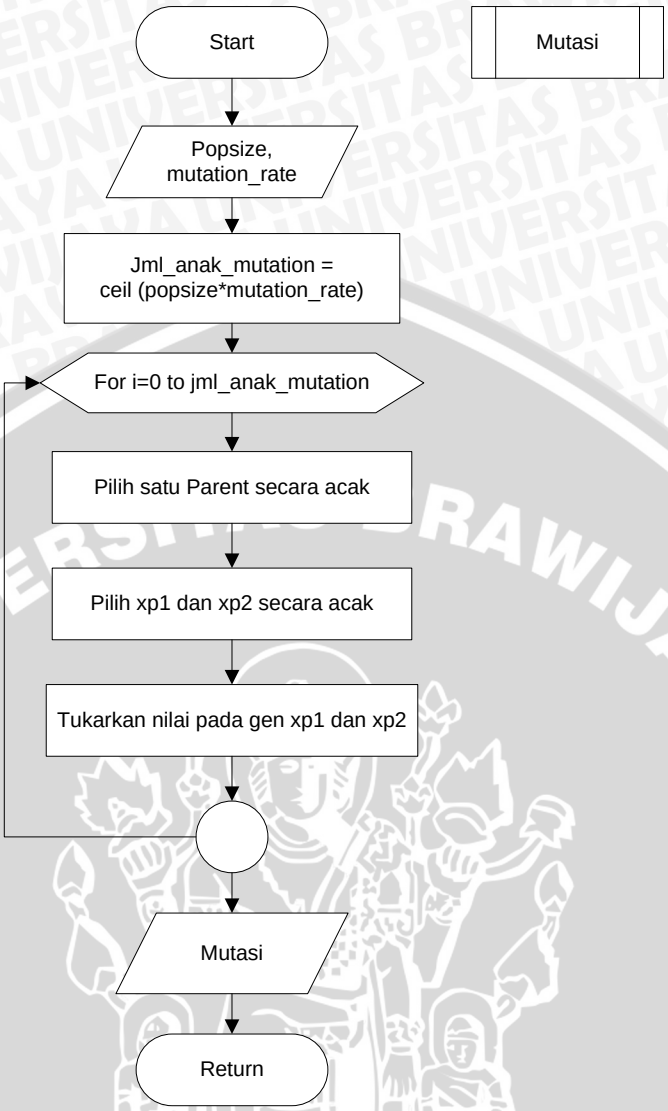
Crosover merupakan persilangan dari dua buah induk atau *parent* yang dipilih titiknya secara acak. Pada proses crossover, metode yang digunakan adalah *single-point crossover*. Langkah pertama yang dilakukan untuk mendapatkan jumlah anak atau *offspring*-nya yaitu memilih dua buah induk secara acak. *Offspring* diperoleh dari perkalian antara *crossover rate* dan *popsize.* Setelah itu, tentukan titik keberapa yang akan dipotong. Titik yang telah dipotong pada induk pertama disilang atau ditukarkan dengan titik yang telah terpotong pada induk kedua. Gambar 4.3 merupakan *flowchart* dari metode *single-point crossover*.



Gambar 4.3 *Flowchart* Modifikasi *One-Cut-Point Crossover*

4.2.4.2 Mutasi

Mutasi merupakan operator pada algoritma genetika yang mengubah atau memindahkan satu atau lebih gen pada suatu individu. Metode mutasi yang digunakan adalah *reciprocal exchange mutation.* Metode ini hanya memindahkan dua gen dalam satu induk. *Flowchart* metode *reciprocal exchange mutation* dapat dilihat pada Gambar 4.4.



Gambar 4.4 *Flowchart Reciprocal Exchange Mutation*

4.2.5 Evaluasi

Pada proses evaluasi dilakukan perhitungan penalti dan *fitness.* Penalti merupakan sebuah nilai pelanggaran yang tidak sesuai aturan. Nilai pelanggaran dihitung berdasarkan berapa banyak kemunculan pelanggaran pada setiap kromosom. Setiap ada pelanggaran maka dihitung 1 pelanggaran. Jenis pelanggaran juga terbagi dua yaitu *soft constraint* dan *hard constraint.* Berikut dapat dilihat pada Tabel 4.12 jenis pelanggaran beserta nilai pelanggaran:

Tabel 4.3 Jenis Pelanggaran

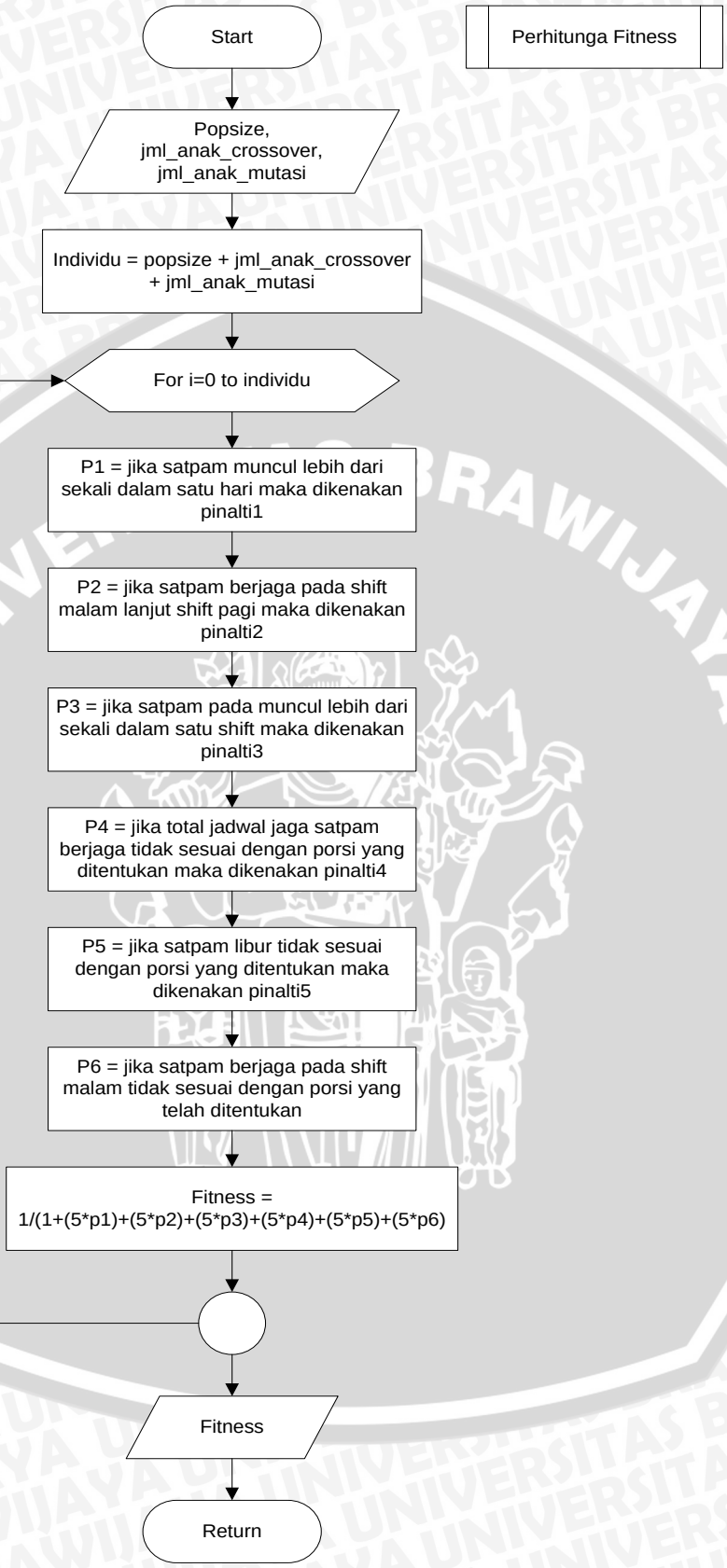
No. Pelanggaran	Keterangan	Jenis Pelanggaran	Konstanta Pelanggaran	Nilai Pelanggaran
P1	Satpam tidak boleh muncul lebih dari sekali dalam satu hari	Hard constraint	5	1
P2	Satpam tidak boleh berjaga pada shift malam dan lanjut shift pagi	Hard constraint	5	1
P3	Satpam tidak boleh muncul lebih dari sekali dalam satu shift	Hard constrain	5	1
P4	Total jadwal jaga satpam dalam tidak sesuai dengan porsi yang telah ditentukan	Hard constraint	5	1
P5	Satpam memiliki jatah libur yang tidak sesuai dengan porsi yang telah ditentukan	Hard constraint	5	1
P6	Satpam berjaga pada shift malam tidak sesuai dengan porsi yang telah ditentukan	Hard constraint	5	1

Nilai dari setiap individu dapat diketahui dengan melakukan perhitungan nilai *fitness*. Nilai *fitness* yang diperoleh menunjukkan kualitas kromosom dalam populasi tersebut. Hasil perhitungan fitness digunakan selanjutnya pada proses seleksi dalam mencari individu terbaik untuk menjadi solusi dalam suatu permasalahan. Semakin besar nilai fitness yang diperoleh maka semakin baik chromosome tersebut untuk dijadikan solusi. Rumus yang digunakan untuk melakukan perhitungan fitnessnya yaitu :

$$Fitness = \frac{1}{1+Penalti}$$

$$= \frac{1}{1+(5\sum P1 + 5\sum P2 + 5\sum P3+5\sum P4+5\sum P5+5\sum P6)}$$

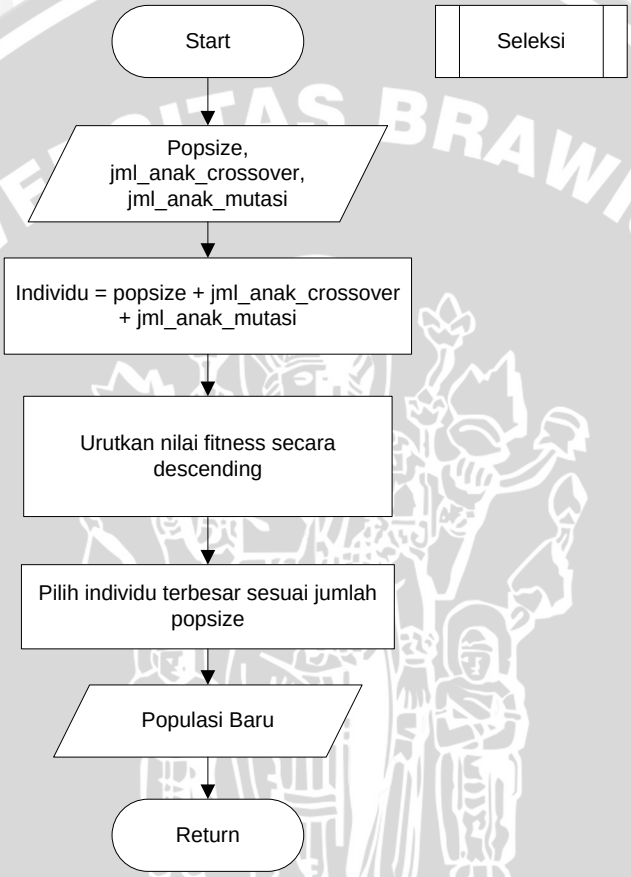
Gambar 4.5 merupakan *flowchart* dari perhitungan *fitness* optimasi penjadwalan jaga satpam.



Gambar 4.5 *Flowchart* Perhitungan *Fitness*

4.2.6 Seleksi

Tujuan dilakukannya seleksi adalah untuk memilih individu dari himpunan populasi dan *offspring* yang dipertahankan hidup pada generasi berikutnya. Beberapa metode seleksi yang sering digunakan yaitu roulette wheel, binary tournament, dan elitism (Mahmudy, 2013). Dalam penelitian ini, metode yang digunakan yaitu metode seleksi *elitism.* Metode seleksi *elitism* ini mengambil nilai fitness terbesar dari individu. Individu tersebut berasal dari dari penampungan populasi dan *offspring.* Gambar 4.6 merupakan *flowchart* dari metode seleksi *elitism*.



Gambar 4.6 *Flowchart* Seleksi *Elitism*

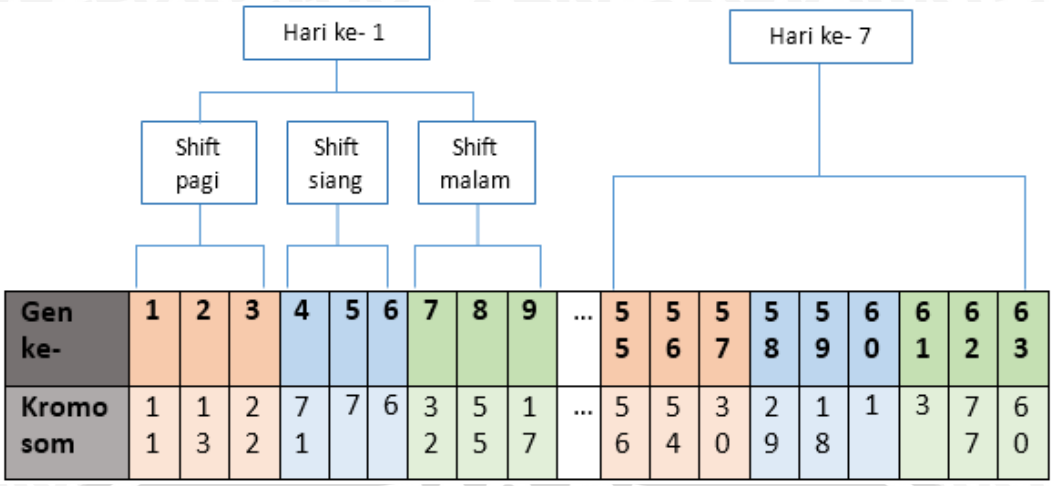
4.3 Perhitungan Manual

4.3.1 Skenario Manualisasi

Diketahui jumlah hari = 7, jumlah semua satpam = 11, jumlah satpam jaga = 9, *popsize* = 3, *crossover rate* = 0.3, *mutation rate* = 0.3.

4.3.2 Representasi Kromosom

Contoh representasi kromosom dapat dilihat pada Gambar 4.7.



Gen ke-	1	2	3	4	5	6	7	8	9	...	55	56	57	58	59	60	61	62	63
Kromosom	11	13	22	71	7	6	32	55	17	...	56	54	30	29	18	1	3	77	60

Gambar 4.7 Representasi Kromosom

Gambar 4.7 merupakan contoh dari pengkodean kromosom yang mana kode satpam tersebut dapat dilihat pada Tabel 4.1. Jumlah gen didapatkan dari perkalian antara jumlah hari dan jumlah satpam jaga. Dibawah ini merupakan perhitungan dari jumlah gen.

$$Jumlah\ Gen = jumlah\ hari * jumlah\ satpam\ jaga$$
$$= 7 * 9$$
$$= 63\ gen$$

Pada skenario manualisasi jumlah satpam jaga pada satu hari yaitu 9, karena setiap shift terbagi 3, maka setiap shift jumlah satpam yang berjaga yaitu 3 orang per hari. Pada gen ke-1 berisikan angka 11 yang berarti kode dari satpam nomor 11, gen ke-2 berisikan angka 13 yang berarti kode dari satpam nomor 13, begitu seterusnya hingga gen paling terakhir. Kode satpam tersebut dibangkitkan secara acak. Dalam satu kromosom tersebut dapat dikatakan sebagai sebuah solusi yang terbentuk dari suatu permasalahan.

4.3.3 Inisialisasi Populasi Awal

Inisialisasi populasi awal dilakukan sebanyak 3 kali karena jumlah *posize* yang diketahui yaitu 3. Contoh tabel inisialisasi populasi awal dapat dilihat pada Tabel 4.4, Tabel 4.5 dan Tabel 4.6.

Tabel 4.4 Parent ke-1 (P1)

Hari	SHIFT 1			SHIFT 2			SHIFT 3		
	1	2	3	1	2	3	1	2	3
1	5	1	4	2	11	7	3	10	8
2	4	10	9	5	11	7	9	7	1
3	9	1	2	4	3	8	8	10	11
4	7	5	3	3	4	10	8	11	9
5	3	4	2	5	5	1	10	8	6
6	5	2	10	8	9	4	6	7	1
7	5	2	10	9	11	1	3	7	6

Tabel 4.5 Parent ke-2 (P2)

Hari	SHIFT 1			SHIFT 2			SHIFT 3		
	1	2	3	1	2	3	1	2	3
1	11	2	1	7	3	10	7	8	4
2	3	10	5	2	4	8	9	1	11
3	5	11	4	9	10	1	8	6	7
4	3	8	1	7	4	5	9	6	2
5	2	10	1	9	7	11	4	8	3
6	9	6	5	8	8	10	3	3	1
7	5	10	7	10	3	1	2	6	4

Tabel 4.6 Parent ke-3 (P3)

Hari	SHIFT 1			SHIFT 2			SHIFT 3		
	1	2	3	1	2	3	1	2	3
1	3	10	5	4	11	6	8	9	2
2	9	7	3	7	2	8	10	4	6
3	2	8	8	1	7	9	3	10	6
4	2	7	4	10	1	11	8	5	3
5	7	1	9	11	2	5	10	4	6
6	8	5	10	1	2	7	9	3	4
7	1	2	4	9	5	3	7	8	6

Gen-gen pada setiap populasi terbentuk secara acak (*random*) dari kode satpam yang ada pada Tabel 4.1.

4.3.4 Reproduksi

4.3.4.1 Crossover

Untuk menetukan berapa banyak crossover yang akan dilakukan maka terlebih dahulu menghitung jumlah *offspring* dengan mengalikan *popsize* dan *crossover rate.* Di bawah ini merupakan contoh perhitungan *offspring.*

$$Offspring = popsize * crossover\ rate$$
$$= 3 * 0.3$$
$$= 0.9 \approx 1$$

Jika *parent* yang terpilih adalah P1 (Tabel 4.6) dan P2 (Tabel 4.7), serta titik *crossover*-nya adalah hari ke-4 atau pada gen ke-36. Maka *crossover* dapat dilakukan dengan mengambil gen P1 dari gen pertama sampai dengan gen sebelum titik potong. Gen tersebut terletak pada hari ke-1 s/d hari ke-4. Setelah itu pada P2, diambil nilai gen yang mana nilai gen tersebut tidak sama dengan nilai gen pada P1. Untuk lebih jelasnya dapat dilihat pada Tabel 4.7, Tabel 4.8, Tabel 4.9 dan Tabel 4.10.

Tabel 4.7 Parent Pertama Yang Disilang (P1)

Hari	SHIFT 1			SHIFT 2			SHIFT 3		
	1	2	3	1	2	3	1	2	3
1	5	1	4	2	11	7	3	10	8
2	4	10	9	5	11	7	9	7	1
3	9	1	2	4	3	8	8	10	11
4	7	5	3	3	4	10	8	11	9
5	3	4	2	5	5	1	10	8	6
6	5	2	10	8	9	4	6	7	1
7	5	2	10	9	11	1	3	7	6

Tabel 4.8 Parent Kedua Yang Disilang (P2)

Hari	SHIFT 1			SHIFT 2			SHIFT 3		
	1	2	3	1	2	3	1	2	3
1	11	2	1	7	3	10	7	8	4
2	3	10	5	2	4	8	9	1	11
3	5	11	4	9	10	1	8	6	7
4	3	8	1	7	4	5	9	6	2
5	2	10	1	9	7	11	4	8	3
6	9	6	5	8	8	10	3	3	1
7	5	10	7	10	3	1	2	6	4

Tabel 4.9 Gen P1 Yang Sama Pada P2

Hari	SHIFT 1			SHIFT 2			SHIFT 3		
	1	2	3	1	2	3	1	2	3
1	11	2	1	7	3	10	7	8	4
2	3	10	5	2	4	8	9	1	11
3	5	11	4	9	10	1	8	6	7
4	3	8	1	7	4	5	9	6	2
5	2	10	1	9	7	11	4	8	3
6	9	6	5	8	8	10	3	3	1
7	5	10	7	10	3	1	2	6	4

Table 4.10 Hasil Crossover (C1)

Hari	SHIFT 1			SHIFT 2			SHIFT 3		
	1	2	3	1	2	3	1	2	3
1	5	1	4	2	11	7	3	10	8
2	4	10	9	5	11	7	9	7	1
3	9	1	2	4	3	8	8	10	11
4	7	5	3	3	4	10	8	11	9
5	6	1	6	2	2	1	7	4	8
6	9	6	5	8	8	10	3	3	1
7	5	10	7	10	3	1	2	6	4

Pada Tabel 4.9, gen-gen yang ditandai dengan warna kuning merupakan gen-gen yang sama dengan gen pada P1. Oleh karena itu, gen yang tidak berwarna kuning akan dipasangkan dengan P1 yang mana P1 tersebut diambil dari gen pertama sampai dengan gen sebelum titik potong. Hasil dari crossover tersebut dapat dilihat pada Tabel 4.10.

4.3.4.2 Mutasi

Jika pada *crossover* jumlah *offspring* tergantung dari *popsize* dan *crossover rate,* maka pada mutasi jumlah *offspring* didapat dari *popsize* dan *mutation rate.* Contoh jumlah *offspring* dapat dilihat pada perhitungan di bawah ini.

$$Offspring = popsize * mutation\ rate$$
$$= 3 * 0.3$$
$$= 0.9 \approx 1$$

Jumlah gen pada setiap barisnya adalah 9. Jika *parent* yang terpilih adalah P3 dan titik gen yang terpilih adalah 21 dan 40 (Tabel 4.11), maka pada anaknya kedua nilai dari gen tersebut saling ditukar. Pada gen ke-21 nilainya menjadi 11 sedangkan pada gen ke-40 nilainya menjadi 8 (Tabel 4.12). Untuk lebih jelasnya, proses mutasi dapat dilihat pada Tabel 4.11 dan Tabel 4.12.

Tabel 4.11 Parent Yang Dimutasi (P3)

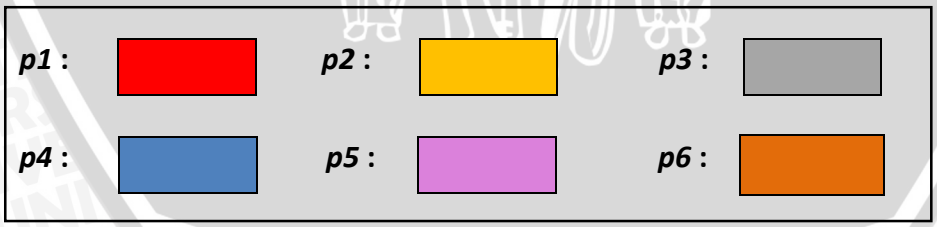
Hari	SHIFT 1			SHIFT 2			SHIFT 3		
	1	2	3	1	2	3	1	2	3
1	3	10	5	4	11	6	8	9	2
2	9	7	3	7	2	8	10	4	6
3	2	8	8	1	7	9	3	10	6
4	2	7	4	10	1	11	8	5	3
5	7	1	9	11	2	5	10	4	6
6	8	5	10	1	2	7	9	3	4
7	1	2	4	9	5	3	7	8	6

Tabel 4.12 Hasil Mutasi (M1)

Hari	SHIFT 1			SHIFT 2			SHIFT 3		
	1	2	3	1	2	3	1	2	3
1	3	10	5	4	11	6	8	9	2
2	9	7	3	7	2	8	10	4	6
3	2	8	11	1	7	9	3	10	6
4	2	7	4	10	1	11	8	5	3
5	7	1	9	8	2	5	10	4	6
6	8	5	10	1	2	7	9	3	4
7	1	2	4	9	5	3	7	8	6

4.3.5 Evaluasi

Sebelum melakukan perhitungan *fitness* maka dilakukan perhitungan masing-masing pinalti. Untuk lebih jelasnya, proses perhitungan setiap pinalti dapat dilihat pada Tabel 4.13, Tabel 4.14, Gambar 4.9, dan Tabel 4.15. Pada Gambar 4.8 merupakan penjelasan warna yang digunakan pada setiap pinalti.



Gambar 4.8 Penjelasan Warna Pada Pinalti

Tabel 4.13 Pelanggaran p1, p2 dan p3 Pada Parent 1

Hari	SHIFT 1			SHIFT 2			SHIFT 3		
	1	2	3	1	2	3	1	2	3
1	5	1	4	2	11	7	3	10	8
2	4	10	9	5	11	7	9	7	1
3	9	1	2	4	3	8	8	10	11
4	7	5	3	3	4	10	8	11	9
5	3	4	2	5	5	1	10	8	6
6	5	2	10	8	9	4	6	7	1
7	5	2	10	9	11	1	3	7	6

Pada Tabel 4.13 didapatkan jumlah warna merah *(p1)*, warna oren *(p2)* dan warna abu-abu *(p3)* adalah 8, 8, dan 2. Lalu setiap angka pada *p1, p2, dan p3* tersebut dibagi dua dan memperoleh hasil 4, 4, dan 1. Nilai ini didapat berdasarkan ketentuan pelanggaran yang telah dibuat pada Tabel 4.13.

Tabel 4.14 Pelanggaran p4 dan p5 Pada Parent 1

Hari	KODE SATPAM										
	1	2	3	4	5	6	7	8	9	10	11
1	1	1	1	1	1	0	1	1	0	1	1
2	1	0	0	1	1	0	2	0	2	1	1
3	1	1	1	1	0	0	0	2	1	1	1
4	0	0	2	1	1	0	1	1	1	1	1
5	1	1	1	1	2	1	0	1	0	1	0
6	1	1	0	1	1	1	1	1	1	1	0
7	1	1	1	0	1	1	1	0	1	1	1
Jumlah nilai != 0	6	5	6	6	7	3	6	6	6	7	5
Jumlah nilai = 0	1	2	2	1	1	4	2	2	2	0	2

Tabel 4.14 merupakan jumlah jaga masing-masing satpam setiap hari. Nilai 0 pada tabel mewakili jadwal satpam libur, sedangkan nilai yang tidak sama dengan 0 merupakan jadwal satpam masuk perharinya. Untuk menentukan berapa kali satpam diperbolehkan libur dan masuk, dilakukan perhitungan di bawah ini.

Jumlah satpam = 11

Jumlah hari = 7

Jumlah satpam jaga perhari = 9

Jumlah gen = *Jumlah satpam jaga perhari * hari*

= 9 * 7 = 63

Minimal jaga = *Jumlah gen / Jumlah satpam*

= 63 / 11 = 5.72 ≈ 5 (dibulatkan kebawah)

Maksimal jaga = *Minimal jaga + 1*

= 5 + 1 = 6

Minimal libur = *Jumlah hari – Maksimal jaga*

= 7 – 6 = 1

Maksimal libur = *Jumlah hari – Minimal jaga*

= 7 – 5 = 2

Dari perthitungan di atas diperoleh hasil jika setiap satpam mendapatkan jadwal jaga sebanyak 5 atau 6 kali selama seminggu, sedangkan untuk jadwal libur masing-masing satpam yaitu sebanyak 1 atau 2 kali dalam seminggu. Pada tabel 4.14, didapatkan jumlah pelanggaran *p4* dan *p5*. *p4* didapatkan dengan menghitung berapa jumlah nilai selain 0 pada setiap satpam yang tidak sama dengan minimal jaga (5) dan maksimal jaga (6), sedangkan *p5* didapatkan dengan menghitung jumlah nilai 0 pada setiap satpam yang tidak sama dengan minimal libur (1) dan maksimal libur (2). Sehingga diperoleh nilai pelanggaran *p4* dan *p5* yaitu 3 dan 2.

Tabel 4.15 Pelanggaran p6 Pada Parent 1

Kode Satpam	Total Jaga
1	2
2	0
3	2
4	0
5	0
6	3
7	3
8	4
9	2
10	3
11	2

Tabel 4.15 merupakan tabel pelanggaran *p6.* Untuk mendapatkan nilai pelanggaran *p6,* maka dilakukan perhitungan seperti berikut.

Maksimal jaga = 6

Minimal jaga = 5

Jumlah jaga shift 1 dan shift 2 = *Maksimal jaga / jumlah shift*

= 6 / 3 = 2

Maksimal Jaga Shift 3 = *Maksimal jaga – (Jumlah jaga shift 1 + Jumlah jaga shift 2)*

= 6 – (2+2) = 2

Minimal Jaga Shift 3 = *Minimal jaga – (Jumlah jaga shift 1 + Jumlah jaga shift 2)*

= 5 – (2 + 2) = 1

Dari perhitungan di atas didapatkan hasil jika pada tabel 4.15 jumlah jaga satpam tidak sama dengan maksimal jaga shift 3 (2) dan minimal jaga shift 3 (1). Sehingga nilai pelanggaran *p6* adalah 7.

Berdasarkan nilai-nilai pinalti yang telah diperoleh, maka perhitungan *fitness* dapat dilakukan dengan cara dibawah ini.

Diketahui : *p1* = 4, *p2* = 4, *p3* = 1, *p4* = 3 , *p5* = 2 dan *p6* = 7

$$Fitness = \frac{1}{1+Penalti}$$

$$= \frac{1}{1+(5\sum P1 + 5\sum P2 + 5\sum P3 + 5\sum P4 + 5\sum P5 + 5\sum P6)}$$

$$= \frac{1}{1+((5*4)+(5*4)+(5*1)+(5*3)+(5*2)+(5*7))}$$

$$= \frac{1}{106}$$

= 0.00943

Maka nilai *fitness* yang diperoleh pada *parent 1* yaitu 0.00943.

4.3.6 Seleksi

Tabel 4.16 merupakan hasil dari perhitungan *fitness* pada setiap individu. Nilai *fitness* tersebut diurutkan dari yang terbesar sampai yang terkeci (Tabel 4.17). Setelah nilai *fitness* tersebut diurutkan diambil individu yang memiliki nilai *fitness* terbesar sebanyak jumlah *popsize*-nya (Tabel 4.18).

Tabel 4.16 Nilai *Fitness* Tiap Individu

Induk	Fitness
P1	0.009433962
P2	0.009009009
P3	0.010989011
C1	0.007936508
M1	0.010989011

Tabel 4.17 *Descending* Nilai *Fitness*

P(t+1)	asal P(t)	Fitness
P1	P3	0.010989011
P2	M1	0.010989011
P3	P1	0.009433962
P4	P2	0.009009009
P5	C1	0.007936508

Tabel 4.18 Hasil Seleksi Elitism

P(t+1)	asal P(t)	Fitness
P1	P3	0.010989011
P2	M1	0.010989011
P3	P1	0.009433962

Pada Tabel 4.18 terdapat 3 individu yang bertahan hidup untuk generasi selanjutnya. Individu-individu yang bertahan hidup untuk generasi selanjutnya hanya diperbolehkan 3 buah karena jumlah popsizenya adalah 3. Individu-individu tersebut dapat bertahan hidup karena memiliki nilai *fitness* yang lebih besar dibandingkan dengan individu lainnya.

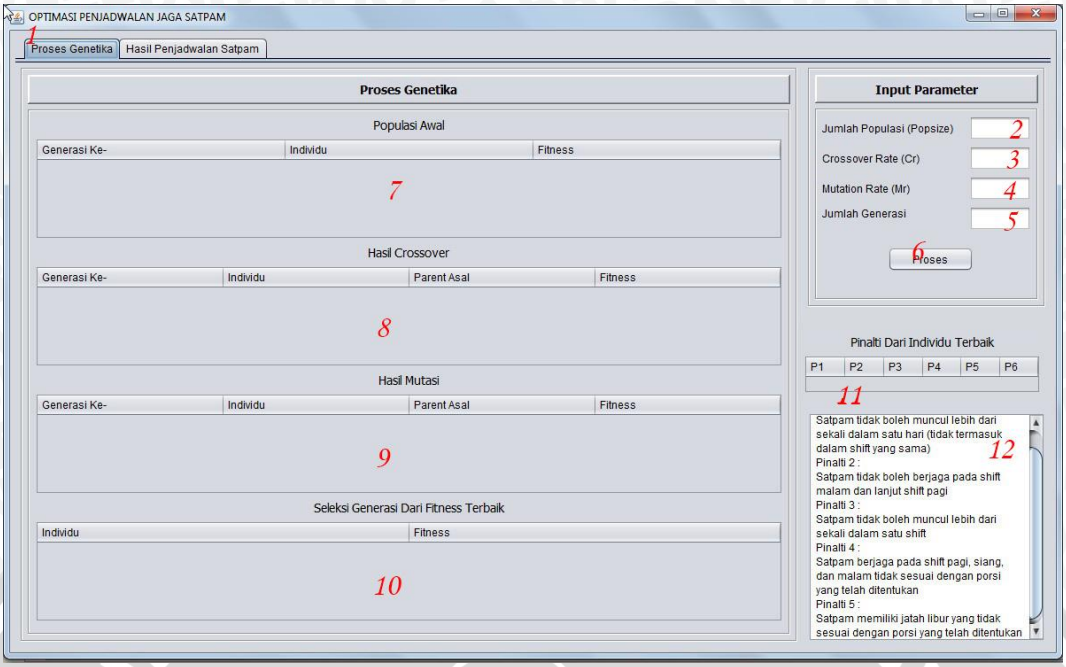
Seleksi pada penelitian ini akan berhenti ketika nilai fitness yang diperoleh pada setiap individu terus sama dan tidak mengalami perubahan lagi.

4.4 Perancangan Antar Muka

Pada sub bab ini akan menjelaskan bagaimana perancangan antar muka (*user interface*) pada sistem penjadwalan jaga satpam ini. Antar muka yang digunakan antara lain, halaman utama (proses genetika), halaman data satpam dan halaman data penjadwalan satpam.

4.4.1 Perancangan Antar Muka Halaman Utama (Data Satpam)

Halaman utama dalam perancangan antar muka ini yaitu proses algoritma genetika pada data satpam. Dimana pada halaman ini terdapat inputan parameter *popsize, crossover rate, mutation rate,* dan jumlah generasi. Untuk lebih jelasnya dapat dilihat pada Gambar 4.9.



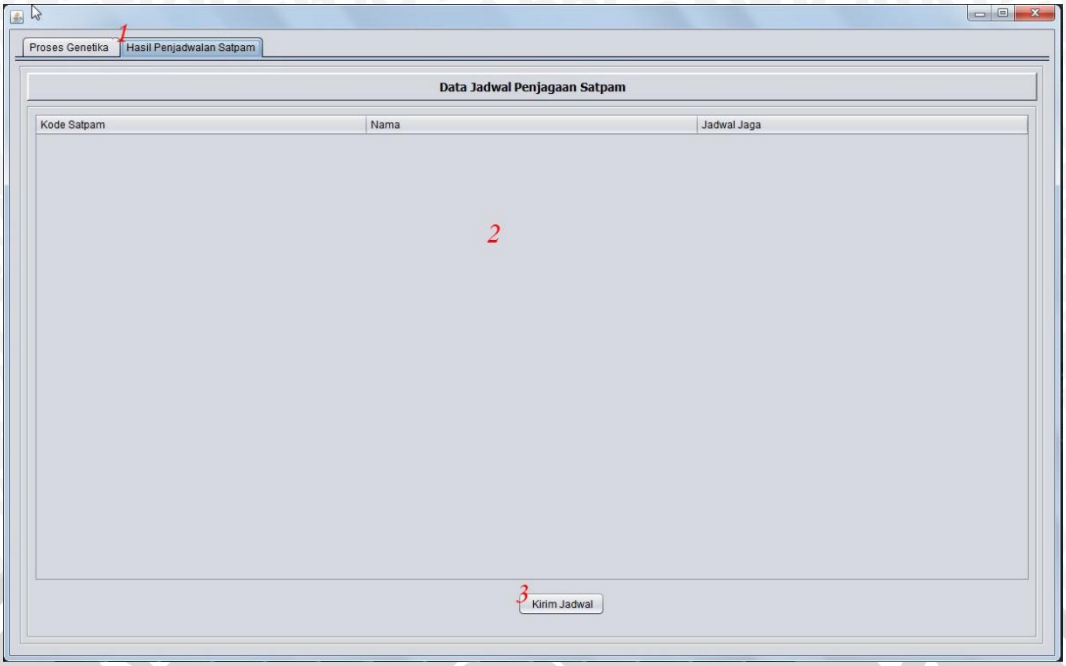
Gambar 4.9 *User Interface* Halaman Proses Genetika

Keterangan Gambar 4.9 :

1. Menu proses genetika, merupakan menu utama dari perancangan antar muka ini.
2. *Input*-an parameter jumlah populasi (*popsize*).
3. *Input*-an parameter jumlah *crossover rate* (*cr*).
4. *Input*-an parameter jumlah *mutation rate* (*mr*).
5. *Input*-an parameter jumlah generasi.
6. Tombol proses untuk memulai proses penjadwalan satpam.
7. Tabel populasi awal.
8. Tabel hasil *crossover*.
9. Tabel hasil mutasi.
10. Tabel hasil seleksi dari *fitness* terbaik.
11. Tabel hasil perhitungan setiap pinalti
12. Keterangan setiap pinalti

4.4.2 Perancangan Antar Muka Data Penjadwalan Satpam

Halaman data penjadwalan data satpam berisi tabel penjadwalan satpam yang telah diproses pada menu proses genetika. Gambarnya dapat dilihat pada Gambar 4.10.



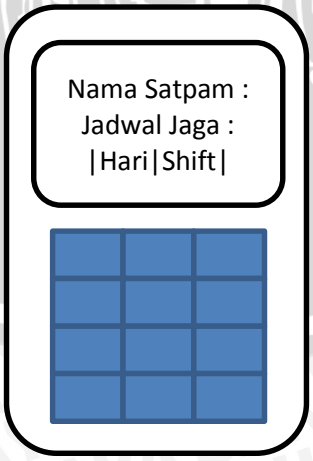
Gambar 4.10 *User Interface* Data Penjadwalan Satpam

Keterangan Gambar 4.10 :

1. Menu data penjadwalan satpam.
2. Tabel data penjadwalan satpam.
3. Tombol simpan dan kirim untuk menyimpan jadwal dan mengirim jadwal.

4.4.3 Perancangan Antar Muka *SMS*

Perancangan antar muka *SMS* merupakan rancangan tampilan pesan yang akan diterima oleh setiap satpam di Universitas Brawijaya. Satpam yang dapat menerima pesan ini yaitu satpam yang nomor *handphone*-nya telah tersimpan dalam *database*. Untuk tampilan SMS-nya dapat dilihat pada Gambar 4.11.



Gambar 4.11 *User Interface* Antar Muka SMS

4.5 Skenario Perancangan

1. Pada halaman utama, pengguna berada pada menu proses genetika.
2. Pengguna diminta memasukkan nilai parameter berupa jumlah populasi (*popsize), crossover rate (cr), mutation rate (mr)* dan jumlah generasi.
3. Jika pengguna telah selesai memasukkan nilai parameter, pengguna dapat menekan tombol proses untuk memproses parameter tersebut.
4. Setelah sistem selesai memproses *input*-an, pengguna dapat melihat hasil prosesnya pada tabel yang ada disebelah kiri *input*-an.
5. Setelah itu, pengguna dapat pergi ke menu data penjadwalan satpam.
6. Pada menu penjadwalan satpam, pengguna dapat melihat kode satpam, nama satpam dan jadwal jaga satpam yang telah diproses.
7. Jika pengguna menyetujui jadwal tersebut, pengguna dapat menekan tombol kirim untuk mengirim data ke nomor *handphone* masing-masing satpam.

4.6 Perancangan Pengujian

Perancangan pengujian dilakukan dalam 3 skenario yaitu ukuran populasi, kombinasi nilai *crossover rate* dan *mutation rate* serta jumlah generasi. Tujuan dilakukan pengujian ini untuk mengetahui kinerja dari tiap-tiap parameternya.

4.6.1 Perancangan Pengujian Berdasarkan Ukuran Populasi

Pengujian ini dilakukan untuk mengetahui ukuran populasi yang optimal dalam pembuatan jadwal jaga satpam menggunakan algoritma genetika. Hipotesa dalam pengujian ini yaitu nilai populasi yang semakin meningkat akan menghasilkan solusi jadwal yang bervariasi dan lebih baik. Hal ini dapat dilihat dari nilai *fitness*-nya. Dalam pengujian ukuran populasi, dilakukan pengujian sebanyak 5 kali dengan rentangan nilai antara 500 sampai dengan 1000. Jumlah iterasinya yaitu 500 serta masing-masing jumlah *crossover rate* dan *mutation rate*-nya adalah 0.4 dan 0.6. Perancangan pengujian terhadap ukuran populasi dapat dilihat pada tabel 4.19.

Tabel 4.19 Perancangan Pengujian Berdasarkan Pengaruh Ukuran Populasi

Percobaan ke-	Banyak Populasi								
	Nilai *Fitness*								
	500	525	600	650	700	800	850	900	1000
1									
2									
3									
4									
5									
Rata-rata Fitness									

4.6.2 Perancangan Pengujian Berdasarkan Jumlah Iterasi/Generasi

Pengujian ini dilakukan untuk mengetahui jumlah generasi yang optimal dalam pembuatan jadwal jaga satpam menggunakan algoritma genetika. Dalam pengujian ini, peneliti melakukan uji coba sebanyak 5 kali. Rentangan nilainya antara 500 sampai dengan 1000. Nilai parameter *crossover rate (cr)* dan *mutation rate (mr)* adalah 0.4 dan 0.6. Sedangkan ukuran populasi yang digunakan berdasarkan nilai pada ukuran populasi yang optimal pada pengujian pertama. Perancangan pengujian terhadap jumlah generasi dapat dilihat pada tabel 4.20.

Tabel 4.20 Perancangan Pengujian Berdasarkan Pengaruh Jumlah Generasi

Percobaan ke-	Jumlah Generasi								
	Nilai *Fitness*								
	500	550	600	625	700	800	850	900	1000
1									
2									
3									
4									
5									
Rata-rata Fitness									

4.6.3 Perancangan Pengujian Berdasarkan Kombinasi *Crossover Rate (*cr*)* dan *Mutation Rate (mr)*

Pengujian ini dilakukan untuk mengetahui kombinasi dari *crossover rate (cr)* dan *mutation rate (mr)* terbaik agar mendapatkan hasil dari solusi yang optimal dalam penjadwalan jaga satpam. Dalam pengujian ini, peneliti menguji coba dengan menggunakan nilai *crossover rate (cr)* yang semakin meningkat dan *mutation rate (mr)* yang semakin menurun. Rentangan nilainya antara 0 sampai dengan 1, serta uji coba dilakukan sebanyak 5 kali percobaan. Ukuran populasinya yaitu sesuai dengan hasil yang optimal pada pengujian pertama, sedangkan untuk nilai iterasinya juga sesuai dengan nilai iterasi yang optimal pada pengujian kedua. Tabel 4.21 merupakan tabel perancangan uji coba pengaruh kombinasi *cr* dan *mr.*

Tabel 4.21 Perancangan Pengujian Pengaruh Kombinasi cr dan mr

Percobaan ke-	***Crossover Rate : Mutation Rate***										
	Nilai *Fitness*										
	0 : 1	0.1: 0.9	0.2: 0.8	0.3: 0.7	0.4: 0.6	0.5: 0.5	0.6: 0.4	0:7: 0.3	0.8: 0.2	0.9: 0.1	1 : 0
1											
2											
3											
4											
5											
Rata-rata Fitness											

4.6.3.1 Perancangan Pengujian Berdasarkan Mutation Rate (mr)

Pengujian ini dilakukan untuk mengetahui seberapa optimalnya nilai *mutation rate* jika dikombinasikan dengan *crossover rate* yang memiliki nilai tetap. Nilai *crossover rate* yang digunakan dalam pengujian ini yaitu nilai *crossover rate* yang memiliki rata-rata terbesar pada pengujian kombinasi *crossover rate* dan *mutation rate* yang telah diuji coba sebelumnya, sedangkan untuk nilai *mutation rate*-nya yaitu 0 s/d 1 dengan masing-masing selisih 0.1. Uji coba ini akan dilakukan sebanyak 5 kali percobaan. Tabel perancangan uji cobanya dapat dilihat pada Tabel 4.22.

Tabel 4.22 Perancangan Pengujian Pengaruh mr

Percobaan ke-	*Mutation Rate*										
	Nilai *Fitness*										
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
1											
2											
3											
4											
5											
Rata-rata Fitness											

4.6.3.2 Perancangan Pengujian Berdasarkan Crossover Rate (cr)

Pengujian ini dilakukan untuk mengetahui seberapa optimalnya nilai *crossover rate* jika dikombinasikan dengan *mutation rate* yang memiliki nilai tetap. Nilai *mutation rate* yang digunakan dalam pengujian ini yaitu nilai *mutation rate* yang memiliki rata-rata terbesar pada pengujian berdasarkan nilai *mutation rate* yang telah diuji sebelumnya, sedangkan nilai *crossover rate*-nya yaitu antara 0 s/d 1 dengan masing-masing selisih 0.1. Uji coba ini akan dilakukan sebanyak 5 kali percobaan. Tabel perancangan uji cobanya dapat dilihat pada Tabel 4.23.

Tabel 4.23 Perancangan Pengujian Pengaruh cr

Percobaan ke-	*Crossover Rate*										
	Nilai *Fitness*										
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
1											
2											
3											
4											
5											
Rata-rata Fitness											

BAB 5 IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi dan bentuk tampilan program.

5.1 Implementasi Program

Dalam implementasi program ini bahasa yang digunakan adalah bahasa JAVA. Implementasi ini dilakukan sesuai dengan metodologi dan perancangan yang telah dibuat sebelumnya.

5.1.1 Proses Pembangkitan Populasi Awal

Pada proses pembangkitan populasi awal dilakukan dengan mengacak kode satpam. Kode tersebut diacak mulai dari angka 1 s/d jumlah semua satpam yang ada dalam *database.* Proses pembangkitan populasi awal dapat dilihat pada source code 5.1.

```
1  int satpam[] = new int[jml_satpam];
2  int random=0;
3  boolean cek;
4  for (int h = 0; h < popsize; h++) {
5     System.out.println("POPULASI "+h);
6     for (int k = 0; k < jml_satpam; k++) {
7        satpam[k]=0;
8     }
9     f=1;
10    for (int i = 0; i < jml_hari; i++) {
11       ArrayList<Integer> list1 = new ArrayList<Integer>();
12       ArrayList<Integer> list2 = new ArrayList<Integer>();
13          for(int x=0;x<jml_satpam_jaga-satpam_jaga_per_shift;
            x++) {
14             list1.add(new Integer(x));
15          }
16          for   (int   x=jml_satpam_jaga-satpam_jaga_per_shift;
            x<jml_satpam_jaga; x++) {
17             list2.add(new Integer(x));
18          }
19          Collections.shuffle(list1);
20          Collections.shuffle(list2);
21          for (int y=0; y<jml_satpam_jaga; y++) {
22             if(y<jml_satpam_jaga-satpam_jaga_per_shift){
23                inisialisasi[h][hr.get(i)][list1.get(y)]=(f);
```

```
24              }
25              if(y>=jml_satpam_jaga-satpam_jaga_per_shift){
26                 int k=y-(jml_satpam_jaga-satpam_jaga_per_shift
                   );
27                 inisialisasi[h][hr.get(i)][list2.get(k)]=(f);
28              }
29              f++;
30              if(f==jml_satpam+1){
31              f=1;
32              }
33           }
34        }
35  }
```

Source code 5.1 Proses Pembangkitan Populasi Awal

Keterangan :

1. Baris 1-3 merupakan inisialisasi variabel.
2. Baris 4-34 merupakan banyaknya populasi yang akan dbangkitkan.
3. Baris 10-23 merupakan proses pengisian nilai kromosom.

5.1.2 Proses Crossover

Metode *crossover* disini menggunakan metode *single-point crossover.* Prosesnya dengan memilih dua buah induk secara acak dan kemudian menyilangkannya. Hasil dari *crossover* berupa 2 buah *offspring,* jika *crossover rate* dikalikan *popsize* menghasilkan nilai genap. Sedangkanakan jika hasil perkalian antara *crossover rate* dan *popsize* berupa nilai ganjil maka akan menghasilkan 1 buah *offspring.* Proses *crossover* dapat dilihat pada source code 5.2.

```
1  for (int h = 0; h < jml_crossover; h++) {
2    int parent1 = randomGenerator.nextInt(popsize);
3    int parent2;
4    do{
5       parent2 = randomGenerator.nextInt(popsize);
6       }while(parent1==parent2);
7    int titik_potong_b = randomGenerator.nextInt(jml_hari-1)+1;
     int c=0;
8    int i=0;
9    int j=0;
10   for (i = 0; i < titik_potong_b; i++) {
11      for (j = 0; j < crossover[0][0].length; j++) {
           crossover[h][i][j]=inisialisasi[parent1][i][j];
```

```
12          crossover_1[h][j+(crossover[0][0].length*i)]=crossov
            er[h][i][j];
13
14          allParent[h+popsize][i][j] = crossover[h][i][j];
            allParent_duplicate[h+popsize][i][j]                =
15          allParent[h+popsize][i][j];
16        }
      }
17    c=j+(crossover[0][0].length*i)-
      (satpam_jaga_per_shift*jml_shift);
18
19    for (int x = 0; x < crossover[0].length; x++) {
        for (int y = 0; y < crossover[0][0].length; y++) {
20      crossover_2[h][y+(crossover[0][0].length*x)]=inisialisa
21      si[parent2][x][y];
22      }
      }
23    for (int x = 0; x < c; x++) {
24      for (int y = 0; y < crossover_2[0].length; y++) {
25        if{
26        (crossover_1[h][x]==crossover_2[h][y]&&crossover_1[h]
          [x]!=0) {
27            crossover_2[h][y]=0;
              break;
28        }
29      }
30    }
31    for (int a = 0; a < crossover_2[0].length; a++) {
32      for (int b = 0; b <crossover_2[0].length-1; b++) {
33        if (crossover_2[h][b] == 0) {
34          int temp = crossover_2[h][b];
35          crossover_2[h][b] = crossover_2[h][b + 1];
36          crossover_2[h][b + 1] = temp;
37        }
38      }
39    }
40    for (int x = titik_potong_b; x < crossover[0].length; x++)
      {
41
42      for (int y = 0; y < crossover[0][0].length; y++) {
          crossover[h][x][y]=crossover_2[h][y+((x-
          titik_potong_b)*crossover[0][0].length)];
43
44        allParent[h+popsize][x][y] = crossover[h][x][y];
```

```
45          allParent_duplicate[h+popsize][x][y] =
            allParent[h+popsize][x][y];
46          }
        }
47  }
48
49
```

Source code 5.2 Proses *Crossover*

Keterangan :

1. Baris 1-49 merupakan banyaknya proses *crossover.*
2. Baris 2-10 merupakan inisialisasi variabel.
3. Baris 11-48 merupakan proses modifikasi *one-cut-point crossover*.

5.1.3 Proses Mutasi

Metode mutasi yang digunakan yaitu *reciprocal exchange mutation.* Prosesnya yaitu memilih sebuah induk secara acak. Setelah itu memilih dua buah gen yang akan ditukarkan. Proses mutasi dapat dilihat pada source code 5.3.

```
1   for (int h = 0; h < mutasi.length; h++) {
2       int parent =
        randomGenerator.nextInt(popsize+jml_crossover);
3       pr[h] = parent+1;
4       int xp1_baris =
        randomGenerator.nextInt(allParent[0].length);
5       int xp1_kolom =
        randomGenerator.nextInt(allParent[0][0].length);
6       int xp2_baris =
        randomGenerator.nextInt(allParent[0].length);
7       int xp2_kolom =
        randomGenerator.nextInt(allParent[0][0].length);
8       for (int i = 0; i < mutasi[0].length; i++) {
9           for (int j = 0; j < mutasi[0][0].length; j++) {
10              mutasi[h][i][j] = allParent[parent][i][j];
11          }
12      }
13      mutasi[h][xp1_baris][xp1_kolom] =
        allParent[parent][xp2_baris][xp2_kolom];
14      mutasi[h][xp2_baris][xp2_kolom] =
        allParent[parent][xp1_baris][xp1_kolom];
15  }
```

Source code 5.2 Proses Mutasi

Keterangan :

1. Baris 1-15 merupakan banyaknya proses *mutasi*.
2. Baris 2-7 merupakan inisialisasi variabel.
3. Baris 8-14 merupakan proses *reciprocal exchange mutation.*

5.1.4 Perhitungan Nilai Pinalti dan *Fitness*

Nilai pinalti diperoleh dari jumlah pelanggaran yang dilakukan pada setiap poin pinalti 1 s/d 5. Perhitungan *fitness* diperoleh dari pembagian angka 100 dengan jumlah penalti 1 s/d 5 yang telah dikalikan dengan konstanta berdasarkan dengan jenis *constraint*-nya. Proses perhitungan pinalti dan *fitness* dapat dilihat pada source code 5.3.

```
1  public void pinaltiFitness(int its){
2    for (int h = 0; h < allParent_duplicate.length; h++) {
3      int a=0;
4      int b=0;
5      int c=0;
6      int e=0, f=0, m=0, n=0, o=0;
7       //pinalti1
8       for (int i = 0; i < allParent_duplicate[0].length;
       i++) {//hari
9         for (int k = 0; k <
10        allParent_duplicate[0][0].length; k++) {//gen
            for (int j = k; j <
11          allParent_duplicate[0][0].length-1; j++) {//gen,
            j=k
12            if(allParent_duplicate[h][i][j+1]==allParent_du
              plicate[h][i][k]){
                if(allParent_duplicate[h][i][k]!=-1){
13                a+=1;
14                allParent_duplicate[h][i][j+1]=-1;
15                break;
16               }
17             }
18          }
19          //pinalti2
20          if(k<allParent_duplicate[0][0].length/jml_shift){/
            /gen
21            if(i<allParent_duplicate[0].length-1){
22              for (int l = 0; l <
23              allParent_per_shift[0][0][0].length; l++)
                {//gen
```

```
24                  if(allParent_per_shift[h][jml_shift-
                    1][i][k]==allParent_per_shift[h][0][i+1][l]
                    ){
25                      b+=1;
                        break;
                    }
26              }
27          }
28      }
29    }
30    //pinalti3
31    for (int s = 0; s < allParent_per_shift[0].length;
      s++) {//shift
32      for (int k = 0; k <
        allParent_per_shift[0][0][0].length; k++) {//gen
33        for (int l = k; l <
          allParent_per_shift[0][0][0].length-1; l++) {
34          if(allParent_per_shift[h][s][i][k]==allParent
            _per_shift[h][s][i][l+1]){
35            c+=1;
            }
36        }
37      }
38    }
39  }
40  //pinalti5
41  for (int l = 0; l < jml_satpam; l++) {
42    e=0;
43    n=0;
44    for (int i = 0; i < allParent_duplicate[0].length;
      i++) {
45      for (int j = 0; j <
        allParent_duplicate[0][0].length; j++) {
46        if((l+1)==allParent_duplicate[h][i][j]){
47          e+=1;
48        }
49      }
50    }
51  }
52  if((jml_hari-e)!=MaxLibur&&(jml_hari-e)!=MinLibur){
53    f+=1;
54  }
55
```

```
56          //pinalti4 satpam jaga sesuai tidak sesuai dengan
            porsi yang ditentukan 23/24
57          if(e!=MinJaga&&e!=MaxJaga){
58             m+=1;
59          }
60          //pinalti6 satpam jaga pada shift malam tidak sesuai
            dengan porsi yang ditentukan
61          for (int j = 0; j < allParent_per_shift[0][0].length;
            j++) {
62             for (int k = 0; k <
63             allParent_per_shift[0][0][0].length; k++) {
64                 if (allParent_per_shift[h][2][j][k]==(1+1)) {
65                    n+=1;
66                 }
67             }
68          }
69          if (n!=y1&&n!=y2) {
70             o+=1;
71          }
72       }
73       P1[h]=a;
74       P2[h]=b;
75       P3[h]=c;
76       P4[h]=m;
77       P5[h]=f;
78       P6[h]=o;
         fitness[h] =
         (float)1/(1+((5*P1[h])+(5*P2[h])+(5*P3[h])+(5*P4[h])+(5*
         P5[h])+(5*P6[h])));
79       }
80    }
```

Source code 5.3 Perhitungan Pinalti dan *Fitness*

Keterangan :

1. Baris 2-70 merupakan proses perhitungan masing-masing pinalti.
2. Baris 71-80 merupakan proses penyimpanan nilai masing-masing pinalti ke dalam array
3. Baris 78 merupakan proses perhitungan *fitness.*

5.1.5 Proses Seleksi

Metode seleksi yang digunakan yaitu elitism. Pada proses ini populasi awal, *crossover,* dan mutasi yang telah memiliki nilai *fitness* akan diurutkan mulai dari *fitness* terbesar sampai yang terkecil sebanyak jumlah *posize*-nya. Proses seleksi elitism dapat dilihat pada source code 5.5.

```
1   public void seleksi(){
2           float [] fit1 = new float[allParent.length];
3           float [] fit2 = new float[allParent.length];
4           for(int b=0; b < allParent.length; b++){
5               fit1[b] = fitness[b];
6               fit2[b] = fitness[b];
7           }
8
9           for (int a = 0; a < fit1.length-1; a++) {
10              for (int b = 0; b <fit1.length-1; b++) {
11                  if (fit1[b] < fit1[b + 1]) {
12                      float temp = fit1[b];
13                      fit1[b] = fit1[b + 1];
14                      fit1[b + 1] = temp;
15                  }
16              }
17          }
18
19          for (int a = 0; a < allParent.length; a++) {
20              for (int b = 0; b < allParent.length; b++) {
21                  if(fit1[a]==fit2[b]){
22                      seleksi[a] = b;
23                      fit2[b]=-1-a;
24                      break;
25                  }
26              }
27          }
28      }
```

Source code 5.5 Proses Seleksi

Keterangan :

1. Baris 2-3 merupakan inisialisasi variabel.
2. Baris 4-7 merupakan proses penyimpanan nilai *fitness* ke dalam fit1 dan fit2.

3. Baris 9-17 merupakan proses mengurutkan nilai *fitness* mulai dari yang terbesar ke yang terkecil
4. Baris 19-27 merupakan proses *parent* berdasarkan nilai *fitness* yang terbesar sampai yang terkecil.

5.1.6 Proses Pembentukan Populasi Baru

Proses ini yaitu mengganti nilai dari populasi awal dengan populasi baru. Populasi baru ini didapatkan dari nilai *fitness* populasi awal, *crossover,* dan mutasi yang telah diseleksi. Proses pembentukan populasi baru dapat dilihat pada source code 5.6.

```
1  public void popBaru(){
2      for (int h = 0; h < inisialisasi.length; h++) {
3          for (int i = 0; i < inisialisasi[0].length; i++) {
4              for (int j = 0; j < inisialisasi[0][0].length; j++)
               {
5                  inisialisasi[h][i][j]=allParent[seleksi[h]][i]
                   [j];
6              }
7          }
8      }
9  }
```

Source code 5.6 Proses Pembentukan Populasi Baru

Keterangan :

1. Baris 2-8 merupakan proses penyimpanan nilai inisialisasi baru berdasarkan nilai *fitness* tertinggi.

5.2 Implementasi *User Interface*

User interface pada sistem ini terdiri dari dua buah tab, yaitu proses genetika dan data penjadwalan. Tab proses genetika berisi inputan parameter algoritma serta hasil dan proses dari populasi awal, *crossover,* mutasi, dan seleksi. Sedangkan pada tab data penjadwalan satpam berisi kode satpam, nama satpam, dan jadwal penjagaan satpam yang jadwalnya dapat dikirim ke masing-masing satpam. Implementasi *user interface* dapat dilihat pada Gambar 5.1 dan 5.2.

5.2.1 Implementasi *User Interface* Halaman Proses Genetika

Halaman proses genetika akan muncul pertama kali setelah program dijalankan. Pada halaman ini berisi inputan parameter berupa *popsize, crossover rate, mutation rate,* dan jumlah generasi. Halaman ini juga terdapat tombol proses yang berfungsi untuk memproses algoritma genetika dari inputan parameter tersebut. Hasilnya berupa populasi awal, hasil *crossover,* hasil mutasi, dan hasil seleksi. Implementasi halaman proses genetika dapat dilihat pada Gambar 5.1.

5.2.2 Implementasi *User Interface* Data Penjadwalan Satpam

Halaman data penjadwalam satpam berisi kode satpam, nama satpam dan hasil penjadwalan satpam. Halaman ini digunakan untuk mengirimkan hasil penjadwalan yang telah diproses kepada setiap satpam yang telah terdaftar di dalam database. Untuk mengirim pesan hasil penjadwalan terdapat tombol kirim pesan di bawah tabel. Implementasi halaman data penjadwalan satpam dapat dilihat pada Gambar 5.2.

BAB 6 PENGUJIAN DAN ANALISIS

Pada bab ini akan dibahas tentang analisis pengujian pada sistem yang telah diimplementasikan. Proses pengujiannya terbagi menjadi 5 buah yaitu pengujian terhadap ukuran populasi, pengujian terhadap jumlah generasi, pengujian terhadap kombinasi *crossover rate* dan *mutation rate,* pengujian terhadap pengaruh *crossover rate,* dan pengujian terhadap pengaruh *mutation rate.*

6.1 Hasil dan Analisis Pengujian Parameter Algoritma Genetika

Pengujian pada parameter algoritma genetika terdiri dari pengujian ukuran populasi, pengujian jumlah generasi, pengujian kombinasi *crossover rate* dan *mutation rate*, pengujian pengaruh *crossover rate,* dan pengujian pengaruh *mutation rate.*

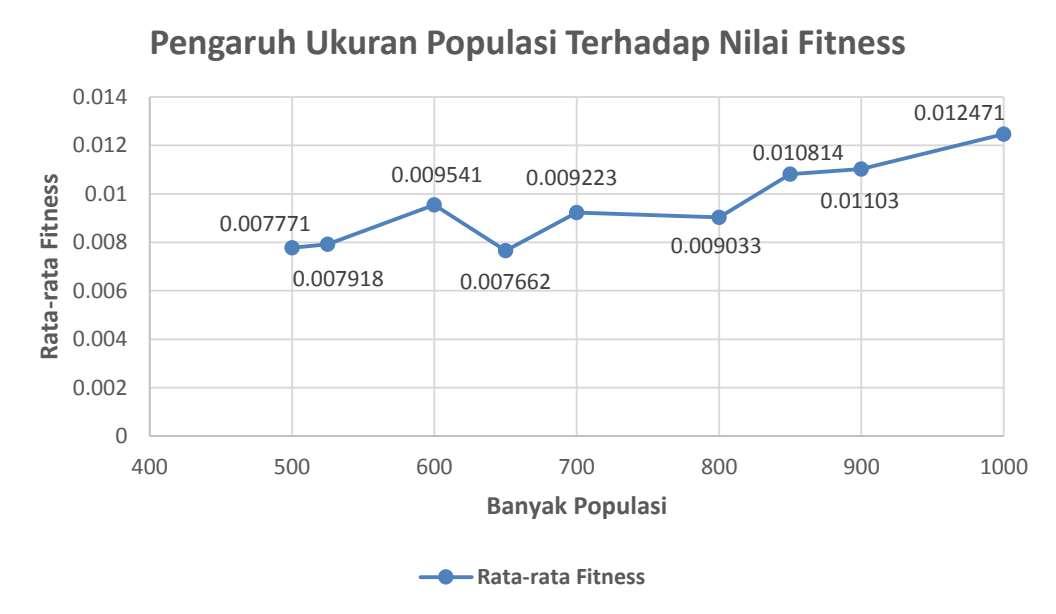
6.1.1 Pengujian Ukuran Populasi

Pengujian pertama ini dilakukan untuk menentukan ukuran populasi yang terbaik agar menghasilkan solusi yang terbaik pada permasalahan serta melihat bagaimana pengaruh ukuran populasi terhadap nilai *fitness*. Jumlah generasi yang digunakan yaitu 500 dengan *crossover rate* dan *mutation rate*-nya 0.4 dan 0.6 Untuk ukuran populasi yang akan diuji adalah 500, 525, 600, 650, 700, 800, 850, 900 dan 1000. Setiap skenario pengujiannya dilakukan sebanyak 5 kali. Hasil pengujian ukuran populasi dapat dilihat pada Tabel 6.1.

Tabel 6.1 Hasil Pengujian Ukuran Populasi

Percobaan ke-	Banyak Populasi								
	Nilai *Fitness*								
	500	525	600	650	700	800	850	900	1000
1	0.01042	0.00621	0.00901	0.00862	0.00943	0.00763	0.00990	0.01515	0.01316
2	0.00735	0.00685	0.00826	0.00662	0.00901	0.01099	0.01042	0.01316	0.01515
3	0.00641	0.00621	0.01316	0.00862	0.01042	0.00763	0.01042	0.00794	0.01316
4	0.00641	0.00990	0.00901	0.00735	0.00990	0.00901	0.01099	0.00901	0.01099
5	0.00826	0.01042	0.00826	0.00709	0.00735	0.00990	0.01235	0.00990	0.00990
Rata-rata *Fitness*	0.00777	0.00792	0.00954	0.00766	0.00922	0.00903	0.01081	0.01103	0.01247

Dari data hasil pengujian, dapat dibuat sebuah grafik untuk melihat pengaruh perubahan ukuran populasi terhadap nilai *fitness* seperti pada gambar 6.1.



Gambar 6.1 Grafik Pengaruh Ukuran Populasi

Berdasarkan hasil pengujian pada Tabel 6.1 dan Gambar 6.1, dapat disimpulkan jika semakin besar jumlah *popsize*-nya maka rata-rata *fitness* yang dihasilkan cenderung meningkat. Pada *popsize* yang memiliki 500 individu, rata-rata nilai *fitness* yang dihasilkan yaitu 0.00777 yang merupakan rata-rata nilai *fitness* terkecil, sedangkan pada *popsize* yang memiliki 1000 individu memiliki rata-rata nilai *fitness* terbaik yaitu 0.01247. Namun pada *popsize* yang memiliki 650 individu mengalami penurunan rata-rata *fitness*. Hal ini disebabkan oleh pembangkitan nilai awal setiap individu yang dilakukan secara acak. Peningkatan ukuran populasi akan memungkinkan solusi yang dihasilkan akan lebih baik, namun tidak menutup kemungkinan jika ukuran populasi yang terlalu besar belum tentu menghasilkan nilai yang lebih optimal daripada ukuran populasi yang lebih kecil.

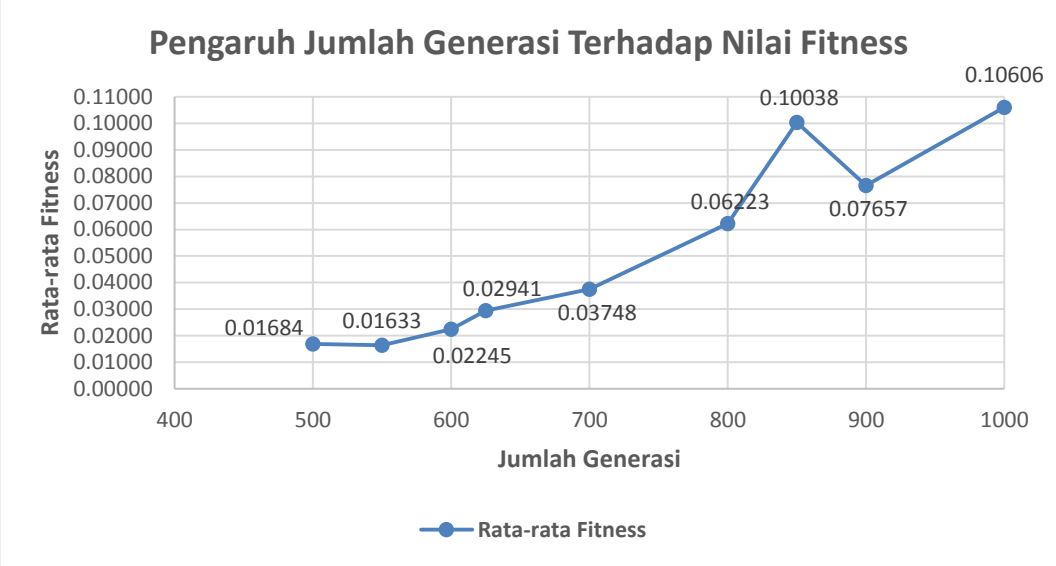
6.1.2 Pengujian Jumlah Generasi

Pengujian kedua yaitu pengujian jumlah generasi terhadap nilai *fitness.* Pengujian jumlah generasi ini digunakan untuk menentukan jumlah generasi terbaik agar menghasilkan solusi terbaik dalam permasalahan ini. Dalam uji coba ini jumlah *popsize* yang digunakan yaitu 1000 individu. *Popsize* tersebut didapatkan dari uji coba ukuran populasi sebelumnya yang menghasilkan nilai optimal. Jumlah generasi yang digunakan adalah 500, 550, 600, 625, 700, 800, 850, 900, dan 1000. Untuk *crossover rate* dan *mutation rate*-nya yaitu 0.4 dan 0.6. Setiap skenario pengujiannya dilakukan sebanyak 5 kali. Hasil pengujian jumlah generasi dapat dilihat pada Tabel 6.2.

Tabel 6.2 Hasil Pengujian Jumlah generasi

Percobaan ke-	Jumlah Generasi								
	Nilai *Fitness*								
	500	550	600	625	700	800	850	900	1000
1	0.00901	0.01163	0.01515	0.02174	0.03846	0.04762	0.16667	0.04762	0.09091
2	0.02778	0.01786	0.02174	0.04762	0.03846	0.06250	0.09091	0.09091	0.09091
3	0.01786	0.02439	0.03846	0.01316	0.04762	0.04762	0.06250	0.06250	0.09091
4	0.01639	0.00990	0.01515	0.03226	0.02439	0.09091	0.09091	0.09091	0.09091
5	0.01316	0.01786	0.02174	0.03226	0.03846	0.06250	0.09091	0.09091	0.16667
Rata-rata Fitness	0.01684	0.01633	0.02245	0.02941	0.03748	0.06223	0.10038	0.07657	0.10606

Dari data hasil pengujian, dapat dibuat sebuah grafik untuk melihat pengaruh perubahan jumlah generasi terhadap nilai *fitness* seperti pada gambar 6.2.



Gambar 6.2 Grafik Pengaruh Jumlah Generasi

Berdasarkan hasil pengujian pada Tabel 6.2 dan Gambar 6.2, dapat disimpulkan jika semakin besar jumlah generasinya maka rata-rata nilai *fitness* yang dihasilkan cenderung meningkat. Pada jumlah generasi 500, rata-rata nilai *fitness* yang dihasilkan yaitu 0.01684 yang merupakan rata-rata nilai *fitness* terkecil, sedangkan pada jumlah generasi 1000 memiliki rata-rata nilai *fitness* terbaik yaitu 0.10606. Percobaan pertama pada jumlah generasi 1000 menghasilkan nilai yang konvergen pada generasi ke 1000, percobaan kedua

menghasilkan nilai yang konvergen pada generasi ke 824, percobaan ketiga menghasilkan nilai yang konvergen pada generasi ke 848, percobaan keempat menghasilkan nilai yang konvergen pada generasi ke 815, dan percobaan kelima menghasilkan nilai yang konvergen pada generasi ke 935.

Dari hasil uji coba tersebut diperoleh jumlah generasi dengan hasil optimal yaitu 1000. Namun pada jumlah generasi 900 rata-rata *fitness* mengalami penurunan. Hal ini dikarenakan pembangkitan nilai awal setiap individu dilakukan secara acak. Hasil pengujian ini membuktikan jika generasi yang digunakan semakin besar maka belum tentu memberikan rata-rata nilai *fitness* terbaik serta jika jumlah generasinya sedikit maka area penjelajahan pada algoritma genetika semakin sempit dan menghasilkan solusi yang kurang optimal.

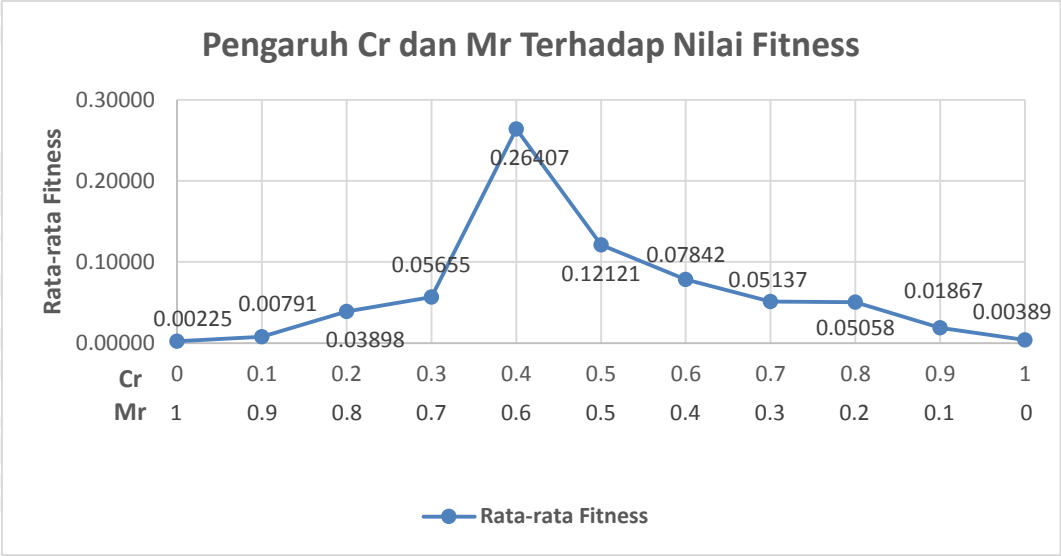
6.1.3 Pengujian Kombinasi Crossover rate dan Mutation rate

Pengujian ketiga dilakukan untuk mengetahui kombinasi *crossover rate (cr)* dan *mutation rate (mr)* yang baik agar menghasilkan nilai *fitness* terbaik. Nilai *cr* dan *mr* yang digunakan yaitu rentangan nilai antara 0 dan 1. Ukuran populasi yang digunakan yaitu 1000 dan jumlah generasi yang digunakan yaitu 1000 yang diperoleh dari ukuran populasi dan jumlah generasi terbaik pada pengujian sebelumnya. Hasil pengujian pengaruh *cr* dan *mr* dapat dilihat pada Tabel 6.3.

Tabel 6.3 Hasil Pengujian Kombinasi Crossover rate dan Mutation rate

Percobaan ke-	***Crossover Rate : Mutation Rate***										
	Nilai *Fitness*										
	0.0 : 1.0	0.1: 0.9	0.2: 0.8	0.3: 0.7	0.4: 0.6	0.5: 0.5	0.6: 0.4	0.7:0 .3	0.8: 0.2	0.9: 0.1	1.0 : 0.0
1	0.00 222	0.00 662	0.01 408	0.06 250	0.04 762	0.16 667	0.04 762	0.04 762	0.03 226	0.01 639	0.00 338
2	0.00 194	0.00 498	0.03 226	0.06 250	0.09 091	0.16 667	0.09 091	0.03 226	0.01 961	0.02 778	0.00 463
3	0.00 262	0.00 901	0.06 250	0.06 250	1.00 000	0.09 091	0.16 667	0.04 762	0.09 091	0.01 639	0.00 415
4	0.00 232	0.01 235	0.04 762	0.04 762	0.09 091	0.09 091	0.02 439	0.03 846	0.04 762	0.01 639	0.00 369
5	0.00 217	0.00 662	0.03 846	0.04 762	0.09 091	0.09 091	0.06 250	0.09 091	0.06 250	0.01 639	0.00 362
Rata-rata Fitness	0.00 225	0.00 791	0.03 898	0.05 655	0.26 407	0.12 121	0.07 842	0.05 137	0.05 058	0.01 867	0.00 389

Dari data hasil pengujian, dapat dibuat sebuah grafik untuk melihat pengaruh kombinasi *crossover rate* dan *mutation rate* terhadap nilai *fitness* seperti pada gambar 6.3.



Gambar 6.3 Grafik Pengaruh *Cr* dan *Mr*

Dari hasil pengujian Tabel 6.3 dan Gambar 6.3 diperoleh nilai rata-rata *fitness* yang beragam karena dalam algoritma genetika tidak ada ketetapan nilai *cr* dan *mr* untuk memperoleh solusi yang optimal. Rata-rata *fitness* terbaik pada Tabel 6.3 dan Gambar 6.3 yaitu 0.26407. Hal itu didapatkan dari uji coba kombinasi *cr* = 0.4 dan *mr*=0.6. Sedangkan pada kombinasi *cr* = 0 dan *mr* = 1, diperoleh nilai rata-rata fitness terendah yaitu 0.00225.

Berdasarkan bentuk grafik pada Gambar 6.3 diperoleh kesimpulan jika cr yang terlalu besar dan mr yang terlalu rendah mengakibatkan konvergensi dini dimana solusi yang dihasilkan tidak terlalu baik ketika diuji coba dilakukan pada beberapa generasi saja. Selain itu nilai cr yang terlalu rendah dan mr yang terlalu besar dapat mengakibatkan menurunnya nilai rata-rata *fitness* yang membuat kemampuan algoritma genetika menurun, hal ini menyebabkan algoritma genetika tidak mampu menjelajahi *optimal local.*

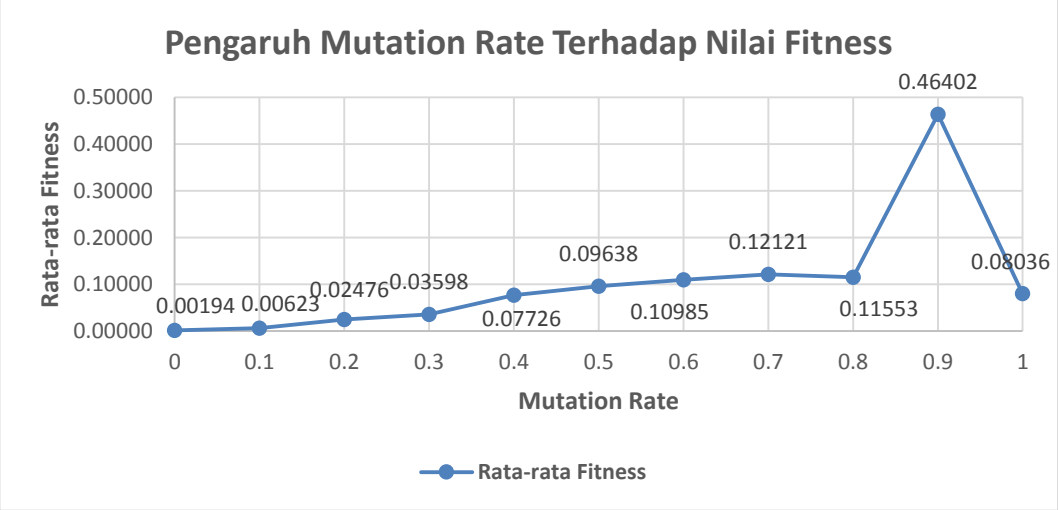
6.1.4 Pengujian Berdasarkan *Mutation Rate (mr)*

Pengujian ini dilakukan untuk mengetahui kombinasi *crossover rate (cr)* terbaik dengan *mutation rate (mr)* agar menghasilkan nilai *fitness* terbaik. Nilai *cr* yang digunakan yaitu 0.4 yang didapatkan dari nilai *fitness* terbaik pada hasil uji coba kombinasi *crossover rate* dan *mutation rate*, sedangkan *mr* yang digunakan yaitu rentangan nilai antara 0 dan 1 dengan percobaan sebanyak 5 kali. Ukuran populasi yang digunakan yaitu 1000 dan jumlah generasi yang digunakan yaitu 1000 yang diperoleh dari ukuran populasi dan jumlah generasi terbaik pada pengujian sebelumnya. Hasil pengujian berdasarkan *mutatin rate* dapat dilihat pada tabel 6.4.

Tabel 6.4 Hasil Pengujian Berdasarkan *Mutation Rate (mr)*

Percobaan ke-	*Mutation Rate*										
	Nilai *Fitness*										
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
1	0.00196	0.00198	0.027778	0.03846	0.03846	0.03846	0.16667	0.16667	0.06625	1.00000	0.16667
2	0.00187	0.01042	0.027778	0.03846	0.16667	0.16667	0.16667	0.09091	0.16667	0.09091	0.0625
3	0.00215	0.00177	0.03226	0.03226	0.09091	0.06250	0.09091	0.09091	0.09091	1.00000	0.04762
4	0.00181	0.00709	0.01961	0.03226	0.027778	0.04762	0.06250	0.16667	0.09091	0.0625	0.0625
5	0.00190	0.00990	0.01639	0.03846	0.06250	0.16667	0.06250	0.09091	0.16667	0.16667	0.0625
Rata-rata *Fitness*	0.00194	0.00623	0.02476	0.03598	0.07726	0.09638	0.10985	0.12121	0.11553	0.46402	0.08036

Dari data hasil pengujian, dapat dibuat sebuah grafik untuk melihat pengaruh *mutation rate* terhadap nilai *fitness* seperti pada gambar 6.4.



Gambar 6.4 Grafik Pengaruh *Mr*

Dari hasil uji coba Tabel 6.4 dan Gambar 6.4 didapatkan nilai rata-rata *fitness* yang bervariasi dan cenderung meningkat. Hal ini dikarenakan proses pertukaran gen pada metode *reciprocal exchange mutation* dilakukan secara acak. Pada nilai *mr* = 0.9 didapatkan nilai rata-rata *fitness* terbaik yaitu 0.46902, sedangkan pada nilai *mr* = 0 menghasilkan nilai rata-rata *fitness* terkecil yaitu 0.00194. Pada nilai *mr* = 1, rata-rata *fitness* yang dihasilkan mengalami penurunan yaitu 0.08036.

Sehingga dapat disimpulkan jika *mr* terlalu besar maka algoritma genetika tidak mampu menjelajahi daerah pencarian secara efektif namun jika *mr* terlalu kecil maka algoritma genetika tidak mampu memperluas daerah pencarian.

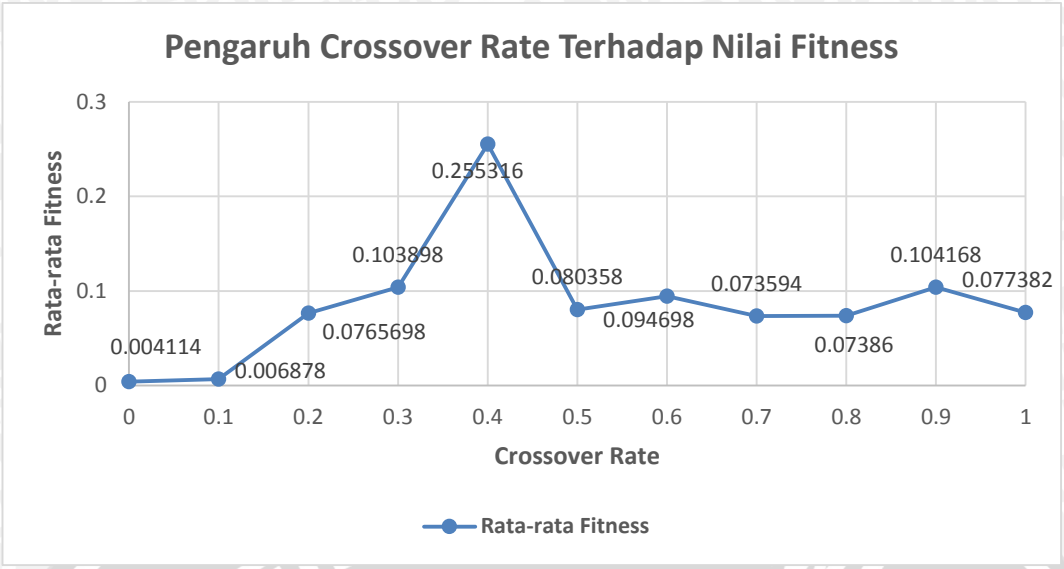
6.1.5 Perancangan Pengujian Berdasarkan *Crossover Rate (cr)*

Pengujian ini dilakukan untuk mengetahui kombinasi *crossover rate (cr)* dengan *mutation rate (mr)* terbaik agar menghasilkan nilai *fitness* terbaik. Nilai *cr* yang digunakan yaitu rentangan nilai antara 0 dan 1, sedangkan *mr* yang digunakan yaitu 0.9 yang didapatkan dari nilai *fitness* terbaik pada hasil uji coba pengujian berdasarkan nilai *mutation rate*. Ukuran populasi yang digunakan yaitu 1000 dan jumlah generasi yang digunakan yaitu 1000 yang diperoleh dari ukuran populasi dan jumlah generasi terbaik pada pengujian sebelumnya. Hasil pengujian berdasarkan *crossover rate* dapat dilihat pada tabel 6.5.

Tabel 6.5 Hasil Pengujian Berdasarkan *Crossover Rate* (cr)

Percobaan ke-	*Crossover Rate*										
	Nilai *Fitness*										
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
1	0.00362	0.00641	0.06250	0.04762	0.09091	0.06250	0.06250	0.09091	0.09091	0.06250	0.06250
2	0.00442	0.00641	0.04762	0.04762	0.06250	0.06250	0.09091	0.04762	0.06250	0.16667	0.06250
3	0.00442	0.00474	0.09091	0.16667	0.09091	0.16667	0.16667	0.09091	0.06250	0.16667	0.04762
4	0.00442	0.01042	0.09091	0.16667	1.00000	0.04762	0.06250	0.09091	0.06250	0.06250	0.04762
5	0.00369	0.00641	0.09091	0.09091	0.03226	0.06250	0.09091	0.04762	0.09091	0.06250	0.16667
Rata-rata *Fitness*	0.00411	0.00688	0.07657	0.10390	0.25532	0.08036	0.09470	0.07359	0.07386	0.10417	0.07738

Dari data hasil pengujian, dapat dibuat sebuah grafik untuk melihat pengaruh *crossover rate* terhadap nilai *fitness* seperti pada gambar 6.5.



Gambar 6.5 Grafik Pengaruh *Cr*

Dari hasil uji coba Tabel 6.5 dan Gambar 6.5 didapatkan nilai rata-rata *fitness* yang bervariasi. Hal ini dikarenakan pada proses pemilihan induknya dilakukan secara acak dan juga pada pemilihan titik potongnya juga dilakukan secara acak. Pada nilai *cr* = 0.4 didapatkan nilai rata-rata *fitness* terbaik yaitu 0.25532, sedangkan pada nilai *cr* = 0 menghasilkan nilai rata-rata *fitness* terkecil yaitu 0.00411. Sehingga dapat disimpulkan jika *cr* terlalu kecil maka algoritma genetika tidak mampu menjelajahi daerah pencarian secara efektif namun jika *cr* terlalu besar maka algoritma genetika tidak mampu memperluas daerah pencarian.

6.2 Hasil Analisa dan Pengujian Sistem

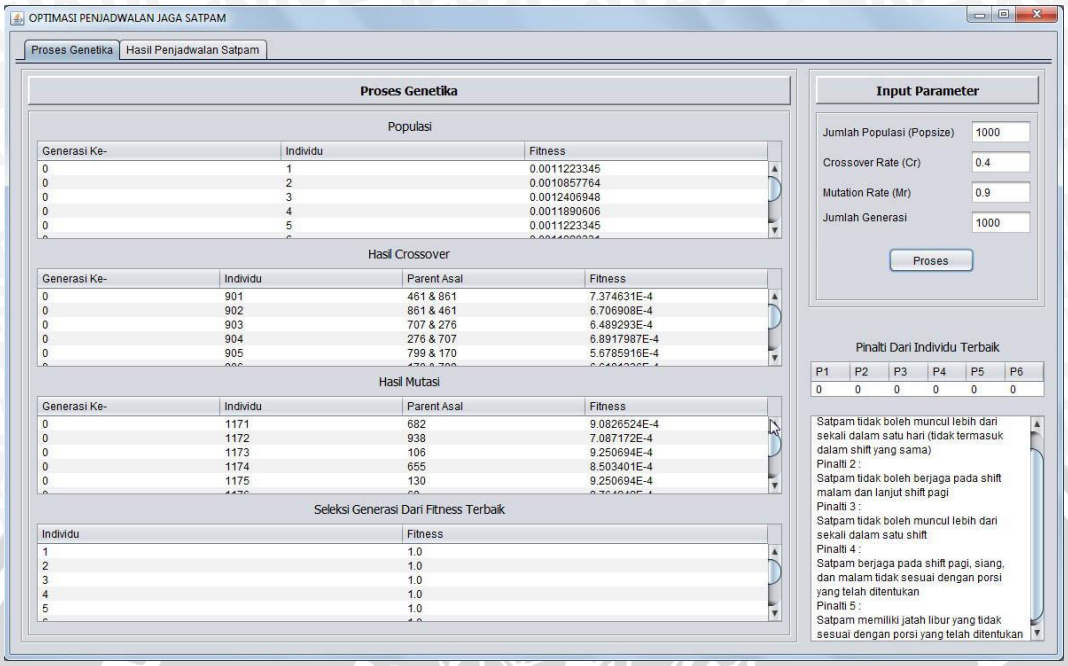
Pengujian sistem dilakukan untuk melihat seberapa baiknya solusi yang dihasilkan jika parameter yang digunakan adalah paremeter yang memiliki nilai rata-rata *fitness* terbaik pada uji coba sebelumya. Solusi yang dihasilkan nanti tergantung seberapa banyak pelanggaran yang dilakukan oleh setiap individu. Selain itu, pengujian ini juga untuk memeriksa apakah sistem SMS *gateway* sudah dapat berfungsi dengan benar atau tidak dengan cara mengirim pesan jadwal penjagaan kepada setiap satpam.

6.2.1 Hasil Pengujian Sistem Proses Algoritma Genetika

Pada pengujian sistem proses algoritma genetika, nilai inputan parameter yang digunakan yaitu parameter yang memiliki nilai rata-rata *fitness* terbaik dan optimal pada pengujian sebelumnya. Berikut parameter algoritma genetika yang digunakan untuk pengujian ini :

Ukuran populasi (*popsize)*	: 1000
Banyak Generasi	: 1000
Crossover Rate	: 0.4
Mutation Rate	: 0.9

Dari inputan parameter yang digunakan untuk pengujian ini, didapatkan hasil pengujian sistem proses algoritma genetika seperti pada Gambar 6.6.



Gambar 6.6 Proses Genetika

Pinalti Dari Individu Terbaik

P1	P2	P3	P4	P5	P6
0	0	0	0	0	0

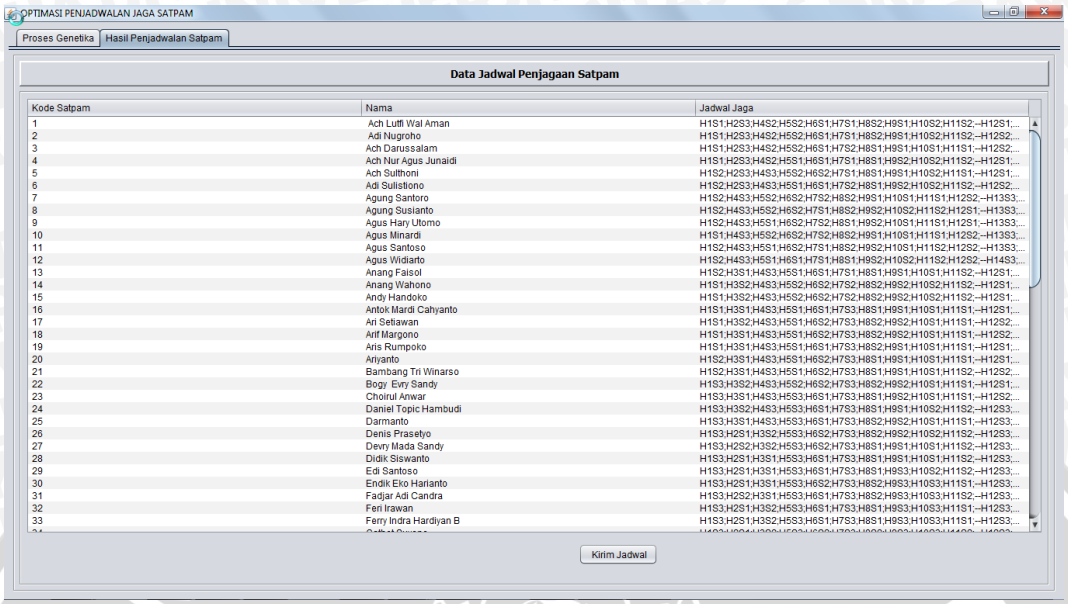
Gambar 6.7 Pinalti Dari Individu Terbaik

Berdasarkan Gambar 6.7, dapat dilihat tidak terdapat pelanggaran pinalti. Hal ini menunjukkan bahwa nilai rata-rata *fitness* yang dihasilkan mencapai nilai optimal dan sempurna karena menghasilkan nilai rata-rata *fitness* yaitu 1.0. Namun tidak menutup kemungkinan jika percobaan dilakukan kembali terdapat pelanggaran pinalti sehingga nilai *fitness* yang dihasilkan menjadi tidak sempurna atau dibawah 1.0. Hal ini disebabkan oleh pembangkitan nilai awal populasi setiap individu dilakukan secara acak serta jumlah gen yang panjang yaitu 1890. Kombinasi *crossover rate* dan *mutation rate* yang digunakan juga menjadi faktor yang mempengaruhi munculnya pinalti pada proses penjadwalan karena metode yang digunakan pada *crossover* dan mutasi melakukan proses pertukaran induk dan gen secara acak.

6.2.2 Hasil Pengujian Sistem SMS Gateway

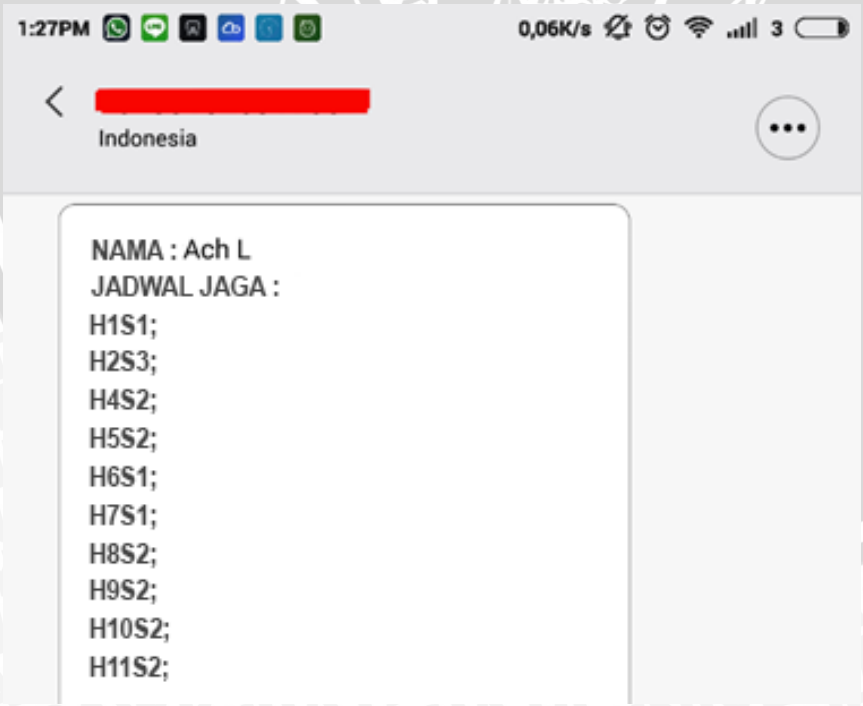
Pada pengujian sistem *SMS gateway*, aplikasi yang digunakan untuk mengirimkan pesan yaitu *Gammu*. Pesan yang akan dikirimkan berupa hasil proses perhitungan algoritma genetika yang telah diuji coba pada pengujian sistem

proses algoritma genetika. Hasil proses perhitungan algoritma genetika yang berupa jadwal penjagaan satpam dapat dilihat pada Gambar 6.8.



Gambar 6.8 Hasil Penjadwalan

Untuk jadwal penjagaan satpam dapat dilihat pada Gambar 6.8 yang berupa jadwal jaga satpam selama 30 hari. Jadwal pada Gambar 6.8 ini yang akan dikirimkan ke setiap satpam melalui *SMS*. Untuk hasil pengiriman jadwalnya dapat dilihat pada gambar 6.9.



Gambar 6.9 SMS Jadwal Jaga Satpam

Gambar 6.9 merupakan pesan yang berupa jadwal penjagaan satpam yang telah dikirimkan ke salah satu nomor *handphone* yang tersimpan dalam *database*. Pesan ini nantinya akan mengirimkan jadwal ke semua satpam yang nomor *handphone*-nya telah disimpan dalam *database*, sehingga masing-masing satpam akan mendapatkan jadwal jaganya sendiri selama 30 hari. Pesan ini nantinya akan dikirim dalam dua bagian karena pesan yang dikirim melebihi 160 karakter. Dalam mengirimkan pesan ini harus dipastikan terlebih dahulu sinyal dan pulsa masih ada. Karena pada *SMS gateway* proses mengirim pesannya sama dengan *SMS* pada umumnya yang membutuhkan sinyal dan pulsa. Dalam proses pengirimannya, pesan akan disimpan terlebih dahulu didalam database dan menunggu antrian untuk dikirim secara otomatis ke nomor *handphone* tujuan.

Pada Gambar 6.9, terdapat pesan "NAMA : Ach L" yang merupakan nama dari salah satu satpam yang menerima jadwal penjagaannya. Pada bagian "JADWAL JAGA :" terdapat tulisan H1S1 maksud dari pesan tersebut yaitu satpam akan berjaga pada hari ke-1 dan shift ke-1 atau shift pagi. Selanjutnya, terdapat pesan H2S3 maka satpam akan berjaga pada hari ke-2 dan shift ke-3 atau malam begitupun pesan dibawahnya yang tertulis H4S2, maka satpam akan berjaga pada hari ke-4 dan shift ke-2 atau siang dan seterusnya. Setiap satpam ini akan mendapat jadwal jaga sebanyak 23 atau 24 kali dalam 30 hari.

BAB 7 KESIMPULAN

7.1 Kesimpulan

Berdasarkan implementasi dan hasil pengujian yang telah dilakukan dalam penerapan algoritma genetika untuk optimasi penjadwalan jaga satpam di universitas brawijaya berbasis sms gateway, maka diperoleh beberapa kesimpulan yaitu :

1. Algoritma genetika yang digunakan pada kasus ini yaitu representasi permutasi yang mana pengkodean kromosomnya menggunakan kode satpam. Representasi kromosomnya memiliki panjang kromosom pada interval [1…1890] yang mana pengkodean kromosom tersebut dibangkitkan secara acak.
2. Metode *crossover* yang digunakan yaitu *single point crossover* dan untuk metode mutasi yaitu *reciprocal exchange mutation.* Sedangkan untuk metode seleksinya yaitu *elitism.* Nilai *fitness* yang dihasilkan dalam pengujian ada yang mencapai nilai optimal, namun banyak juga yang tidak mencapai nilai optimal karena panjangnya gen dan pada pembangkitan nilai setiap individu pada populasi awal dilakukan secara sehingga menyebabkan nilai rata-rata *fitness* yang dihasilkan bervariasi.
3. Untuk menetukan nilai parameter maka dilakukan pengujian parameter yang terdiri dari *popsize,* jumlah generasi serta kombinasi antara *crossover rate* dan *mutation rate.* Nilai parameter yang digunakan yaitu parameter yang memiliki nilai rata-rata *fitness* tertinggi dari uji coba sebanyak 5 kali.. Hasil yang didapatkan pada uji coba tersebut berdasarkan nilai rata-rata *fitness* terbaik yaitu *popsize* 1000 individu, *crossover rate* 0.4, *mutation rate* 0.9, dan jumlah generasi 1000. Pada jumlah generasi terbaik yaitu 1000, mengalami konvergen pada rentangan nilai 815 s/d 1000.

7.2 Saran

Untuk mendapatkan solusi yang lebih optimal maka diperlukan pengujian parameter dan waktu percobaan yang lebih banyak lagi. Pemilihan parameter yang tepat dalam memberikan solusi jadwal yang optimal.

DAFTAR PUSTAKA

Astuwasito, Lathief Noor. 2012. *Pembuatan Aplikasi Berbasis SMS Gateway Untuk Pemesanan Tiket Pesawat Menggunakan NetBeans IDE 6.8 Pada Gardoe Tiket*. Sekolah Tinggi Manajemen Informatika Dan Komputer. Yogyakarta.

Berlianty, Intan & Arifin, Miftahol. 2010. *Teknik-teknik Optimasi Heuristik*. Yogyakarta : Graha Ilmu.

Brigida, 2014. *Struktur Umum Algoritma Genetika [Online]*. Tersedia: http://informatika.web.id/struktur-umum-algoritma-genetika.htm [diakses tanggal 21 April 2016].

Desiani, Anita dan Arhami, Muhammad. 2006. *Konsep Kecerdasan Buatan*. Yogyakarta : ANDI Yogyakarta.

Devi, Okky Cintia. 2015. *Penerapan Algoritma Genetika untuk Penjadwalan Asisten Praktikum*. Fakultas Ilmu Komputer. Universitas Brawijaya. Malang.

Ilmi, Rifqy Rosyidah. 2015. *Optimasi Penjadwalan Perawat Menggunakan Algoritma Genetika.* Fakultas Ilmu Komputer. Universitas Brawijaya. Malang.

Kusumadewi, Sri. 2003. *Artifical Intelligence (Teknik dan Aplikasinya).* Graha Ilmu. Yogyakarta.

Kuswadi, Son. 2007. *Kendali Cerdas dan Aplikasi Praktisnya*. Yogyakarta : ANDI Yogyakarta.

Londong, Dedy. 2012. *Penjadwalan Shift Kerja [Online]*. Tersedia: http://dedylondong.blogspot.co.id/2012/03/penjadwalan-shift-kerja.html [diakses tanggal 2 Maret 2016].

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

Mahmudy, Wayan Firdaus. 2015. *Dasar-Dasar 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. 2014. *Optimasi Penjadwalan Two-Stage Assembly Flowshop Menggunakan Algoritma Genetika Yang Dimodifikasi.* Konferensi Nasional Sistem Informasi (KNSI), STMIK Dipanegara, Makassar, 27 Februari-1 Maret, pp. 478-483.

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 Transaction on Computer and Information Technology (ECTI-CIT), vol. 8, no. 1, pp. 80-93.

Muhadkly, 2007. SMS Gateway Menggunakan Gammu. [Online]. Tersedia: http://ilmukomputer.org/2007/09/27/sms-gateway-menggunakan-gammu/ [diakses tanggal 21 April 2016].

Panharesi, Yasmin Ghassani. 2015. *Optimasi Distribusi Barang Dengan Algoritma Genetika.* Program Teknologi Informasi dan Ilmu Komputer. Universitas Brawijaya. Malang.

Priyadna, Anjar dan Riasti, Berliana Kusuma. 2013. *Pembuatan istem Informasi Nilai Akademik Berbasis SMS Gateway Pada SMP Negeri 3 Pringkuku Pacitan*. Indonesian Journal on Networking and Security, Vol. 2, No. 1.

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

Suyanto. 2011. *Artifical Intelligence Searching-Reasoning-Planning-Learning*. Informatika. Bandung.

Wieldan. 2013. *Optimasi Waktu Body Repair Mobil Menggunakan Algoritma Genetika*. Fakultas Ilmu Komputer. Universitas Brawijaya.