OPTIMASI PENEMPATAN RUANG SIDANG SKRIPSI MENGGUNAKAN ALGORITME GENETIKA

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Nelli Nur Rahma
NIM: 155150201111003

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2019

PENGESAHAN

OPTIMASI PENEMPATAN RUANG SIDANG SKRIPSI
MENGGUNAKAN ALGORITME GENETIKA

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Nelli Nur Rahma
NIM: 155150201111003

Skripsi ini telah diuji dan dinyatakan lulus pada
3 Mei 2019
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Budi Darma Setiawan, S.Kom, M.Cs	Agus Wahyu Widodo, S.T, M.Cs
NIP: 19841015 201404 1 002	NIP: 19740805 200112 1 001

Mengetahui
Ketua Jurusan **Teknik Informatika**

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

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

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

Malang, 19 Februari 2019

Nelli Nur Rahma

NIM: 155150201111003

PRAKATA

Puji syukur atas kehadirat Allah SWT atas segala rahmat, taufik, dan hidayah-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul "*Optimasi penempatan ruang sidang skripsi menggunakan algoritme genetika*".

Penulis menyadari bahwa skripsi ini juga tak lepas dari bantuan semua pihak baik berupa bimbingan, motivasi, saran, kritik, maupun doa. Oleh karena itu, penulis ingin meyampaikan rasa hormat dan terima kasih kepada:

1. Bapak Budi Darma Setiawan, S.Kom, M.Cs selaku dosen pembimbing I yang telah membagikan ilmu dan waktu yang telah diluangkan dalam memberikan bimbingan, kritik, dan saran dalam pengerjaan skripsi ini.
2. Bapak Agus Wahyu Widodo, S.T, M.Cs selaku dosen pembimbing II yang telah membagikan ilmu dan waktu yang telah diluangkan dalam memberikan bimbingan, kritik, dan saran dalam pengerjaan skripsi ini.
3. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D selaku Dekan Fakultas Ilmu Komputer Universitas Brawijaya dan selaku dosen mata kuliah Algoritme Genetika penulis yang telah banyak memberikan ilmu pada mata kuliah ini.
4. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D selaku Ketua Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya.
5. Kedua orang tua penulis, Bapak Marimun dan Ibu Samuti, serta kakak yang penulis sayangi Hendrik Setiawan, mbak Yunik, serta dek Amel yang telah memberikan dukungan finansial maupun moril, memberikan motivasi, kasih sayang, doa yang tidak pernah putus, dan selalu sabar dalam mendidik dan membesarkan penulis.
6. Sahabat-sahabat terbaik penulis, Silfia Ajeng Indrawati, Rizka Hidni Syarfina, Paxia Faharuni Sahara yang selalu memberi motivasi, dukungan, dan menjadi tempat terbaik dalam berbagi. Mas Dimas Setiawan yang selalu mengajari dan membagikan ilmunya selama pengerjaan skripsi, dan dek Dwi Retnoningrum yang selalu menjadi pengingat dan memberikan semangat dalam mengerjakan skripsi ini.
7. Teman-teman TIF A khususnya, Tri Rahayuni yang selalu memberi ilmu dan mengajari banyak hal selama di bangku kuliah ini, Dedin, Fatimah, Wahyu, Hanif, Rosita, Arjun, dan Romadlon yang selalu memberikan motivasi dan semangat dalam berbagai hal, dan teman-teman lainnya yang tidak bisa disebutkan satu persatu.
8. Keluarga besar Kelompok Riset Mahasiswa Fakultas Ilmu Komputer periode 2016/2017, 2017/2018, 2018/2019 yang memberikan banyak pengalaman dan pelajaran berharga selama menjadi anggota dan pengurus.
9. Keluarga besar Pers Display Fakultas Ilmu Komputer periode 2016/2017, 2017/2018, 2018/2019 yang juga memberikan banyak pengalaman dan pelajaran berharga selama menjadi anggota dan pengurus.
10. Teman seperjuangan PKL di Pasuruan Mayang Panca Rini, dan teman-teman induksi riset kelas N.

11. Seluruh bapak ibu dosen dan staff akademik Fakultas Ilmu Komputer yang telah memberikan banyak ilmu, bantuan, dan dukungan selama menempuh studi di Fakultas Ilmu Komputer Universitas Brawijaya.
12. Semua pihak yang tidak dapat penulis sebutkan satu persatu dalam membantu menyelesaikan skripsi ini.

Penulis menyadari bahwa skripsi ini tidak akan berhasil tanpa bantuan dan dukungan dari beberapa pihak dan penulis menyadari bahwa skripsi ini masih jauh dari sempurna dan masih banyak terdapat kekurangan. Oleh karena itu, penulis mengharapkan kritik dan saran yang membangun untuk penelitian ini. Akhir kata penulis berharap skripsi ini dapat memberikan manfaat dan berkah bagi semua pihak.

Malang, 19 Februari 2019

Nelli Nur Rahma

Email : nellinurrahma@gmail.com

ABSTRAK

Nelli Nur Rahma, Optimasi Penempatan Ruang Sidang Skripsi Menggunakan Algoritme Genetika.
Pembimbing: Budi Darma Setiawan, S.Kom, M.Cs dan Agus Wahyu Widodo, S.T, M.Cs

Kesulitan dalam penyusunan jadwal sidang skripsi akan berbanding lurus dengan jumlah mahasiswa yang mendaftar. Semakin banyak mahasiswa yang akan sidang skripsi, maka proses penyusunan jadwal akan semakin kompleks. Permasalahan utama yang sering terjadi dalam penempatan ruang sidang skripsi di Fakultas Ilmu Komputer yaitu jika dosen menguji lebih dari satu sidang secara bersambung dengan ruang berbeda yang cukup jauh. Dosen akan membutuhkan waktu lebih untuk berpindah dari satu ruangan ke ruangan lain. Tak jarang pula dalam penempatan ruangnya, dosen ditempatkan pada gedung yang berbeda dengan jarak cukup jauh dalam sesi yang berlanjut sehingga menimbulkan proses sidang skripsi menjadi terlambat dari penjadwalan awal karena proses *moving* tersebut. Seiring dengan berkembangnya Ilmu Pengetahuan dan Teknologi, proses penjadwalan dapat dilakukan dengan lebih baik. Salah satu algoritme yang dapat diterapkan dalam membuat rekomendasi jadwal sidang skipsi yang optimal yaitu Algoritme Genetika. Algoritme Genetika dapat digunakan dalam penyelesaian masalah yang kompleks dengan banyak variabel dan menghasilkan himpunan solusi optimal. Dalam pembentukan kromosom menggunakan representasi permutasi, proses *crossover* menggunakan metode *one cut point crossover*, proses mutasi menggunakan metode *random mutation*, proses evaluasi dengan mencari nilai *fitness* pada masing-masing individu, metode seleksi yang digunakan yaitu *elitism*. Pada hasil pengujian penempatan ruang sidang skripsi untuk satu hari didapatkan nilai rata-rata *fitness* tertinggi sebesar 1,000 pada kombinasi nilai *cr* 0,5 dan *mr* 0,5, ukuran populasi sebesar 90, dan ukuran generasi sebesar 90. Solusi penempatan ruang menggunakan sistem mampu memberikan jadwal yang optimal dengan tidak melanggar *constraint* sama sekali.

Kata kunci**:** algoritme genetika, penempatan ruang, sidang skripsi, optimasi.

ABSTRACT

Nelli Nur Rahma, ***Optimization Room Placement The Final Presentation of Thesis Using Genetic Algorithms.***
***Advisor*: Budi Darma Setiawan, S.Kom, M.Cs dan Agus Wahyu Widodo, S.T, M.Cs**

Difficulty in preparation the final presentation for minor thesis scheduling will comparable with quantity of students who register. The more students will attend to final presentation for minor thesis, then process to preparation scheduling will more complex. The main problem that often happen in room placement final presentation for thesis in Computer Science Faculty is where lecturer examine more than one sessions continuously with different room which far away. Lecturer will need more time for move from one room to another room. Does not rarely in room placement, lecturer will placement in different building with far away in continuous sessions so raises final presentation for minor thesis process will be late from initial scheduling because moving process. With the development of science and technology management, scheduling process should be done better. One of algoritm can use for recommendation of optimal final presentation for minor thesis is Genetic Algoritm. Genetic Algorithm can use for finished complex problems with many variable and result set of optimal solutions. In the formation of chromosome that used is permutation representation, the crossover process that used is one cut point crossover method, the mutation process that used is one random mutation method, the evaluation process with get fitness value each individual, the selection method that used is elitism. In the test result of room placement final presentation for minor thesis in one day obtained highest average fitness value is 1,000 in combination of cr 0,5 and mr 0,5, population size is 90, and generation size is 90. Room placement solution that use system can offer optimal scheduling with does not break the constraint at all.

Keywords: genetic algorithm, room placement, the final presentation of thesis, optimization.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR PERSAMAAN

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar belakang

Sidang skripsi merupakan tahap akhir mahasiswa strata satu yang merupakan penentu kelulusan dari perguruan tinggi yang sedang dijalani. Penjadwalan skripsi biasanya dilakukan oleh sekretaris jurusan dari masing-masing fakultas. Penentuan jadwal skripsi sendiri perlu memperhatikan beberapa hal diantaranya ketersediaan slot waktu dan ruang kosong yang digunakan untuk sidang, kehadiran dua dosen pembimbing dan dua dosen penguji, dan keahlian dari dosen penguji terkait topik skripsi yang diambil oleh mahasiswa (Pramudita, et al., 2016).

Kesulitan dalam penyusunan jadwal sidang skripsi akan berbanding lurus dengan jumlah mahasiswa yang mendaftar (Ardhianto, et al., 2014). Semakin banyak mahasiswa yang akan sidang, maka proses penyusunan jadwal akan semakin kompleks. Permasalahan utama yang sering terjadi dalam penempatan ruang sidang skripsi di Fakultas Ilmu Komputer yaitu jika dosen mendapatkan jadwal lebih dari satu sidang secara bersambung dengan ruang berbeda yang cukup jauh. Dosen akan membutuhkan waktu lebih untuk berpindah dari satu ruangan ke ruangan lain. Tak jarang pula dalam penempatan ruangnya, dosen ditempatkan pada gedung yang berbeda dengan jarak cukup jauh dalam sesi yang berlanjut sehingga menimbulkan proses sidang menjadi terlambat dari penjadwalan awal karena proses *moving* tersebut. Hal ini dikarenakan terbatasnya ruang yang digunakan dalam proses sidang skripsi.

Terkait permasalahan di atas, telah banyak permasalahan mengenai penjadwalan yang dapat diselesaikan menggunakan algoritma genetika. Pada penelitian Ardhianto, et al. (2014) yaitu melakukan implementasi Algoritma Genetika pada penjadwalan sidang tugas akhir Teknik Informatika. Hasil dari penelitian tersebut didapatkan rekomendasi jadwal sidang tugas akhir yang optimal pada periode sidang antara Mei sampai dengan Oktober 2012. Pada penelitian Luber, et al. (2017) dengan menggunakan algoritma genetika dapat menyelesaikan masalah penjadwalan pendamping mahasiswa difabel. Hasil dari penelitian tersebut menunjukkan bahwa algoritma genetika menghasilkan nilai *fitness* sebesar 0.966 dengan 100 generasi. Penelitian Husada, et al. (2018) melakukan optimasi penjadwalan kuliah pengganti menggunakan algoritma genetika. Hasil yang didapatkan dari penelitian tersebut yaitu algoritma genetika menghasilkan nilai *fitness* sebesar 0.667 dengan jumlah generasi 30, ukuran populasi 50, nilai *Cr* 0.7 dan *Mr* 0.3.

Seiring dengan berkembangnya Ilmu Pengetahuan dan Teknologi, proses penjadwalan dapat dilakukan dengan lebih baik. Penerapan pada studi kasus penjadwalan di bidang teknologi yaitu membuat rekomendasi jadwal sidang skripsi dengan penempatan ruang yang optimal. Salah satu algoritme yang dapat

diterapkan dalam membuat rekomendasi jadwal sidang yang optimal yaitu Algoritme Genetika. Algoritme ini berbasiskan populasi untuk optimasi dengan ruang lingkup masalah yang kompleks dan sangat luas. Algoritme Genetika dapat digunakan untuk menyelesaikan masalah yang kompleks dengan banyak variabel. Algoritme Genetika juga dapat menyelesaikan permasalahan dengan banyak obyektif dan menghasilkan himpunan solusi optimal (Mahmudy, 2015).

Dari uraian tersebut diusulkan optimasi penempatan ruang sidang skripsi menggunakan algoritme genetika dengan data yang digunakan adalah data mahasiswa yang mendaftar sidang skripsi, dosen pembimbing dan dosen penguji, ruang kelas dan waktu yang tersedia untuk sidang skripsi. Sistem penjadwalan ini ditujukan untuk mengetahui kualitas solusi jadwal yang diperoleh dari penerapan algoritme genetika sehingga didapatkan hasil penjadwalan yang optimal.

1.2 Rumusan masalah

1. Bagaimana pengaruh dari nilai parameter genetika untuk optimasi penempatan ruang sidang skripsi?
2. Bagaimana hasil pengujian dan kualitas solusi yang didapatkan dari penerapan Algoritme Genetika untuk menyelesaikan masalah optimasi penempatan ruang sidang skripsi?

1.3 Tujuan

Tujuan dari penelitian ini adalah sebagai berikut:

1. Mengetahui pengaruh dari nilai parameter genetika dalam optimasi penempatan ruang sidang skripsi
2. Menguji dan mengetahui kualitas solusi yang didapatkan dari penerapan Algoritme Genetika untuk optimasi penempatan ruang sidang skripsi.

1.4 Manfaat

Manfaat dari penelitian ini adalah sebagai berikut:

1. Bagi Fakultas Ilmu Komputer

   Memberikan solusi alternatif untuk penempatan ruang sidang skripsi.

2. Bagi Penulis

   Menjadi salah satu jalur untuk mengimplementasikan metode Algoritme Genetika untuk dapat menyelesaikan permasalahan penempatan ruang sidang skripsi.

1.5 Batasan masalah

Agar pembahasan tidak menyimpang dari tujuan awal penelitian, maka batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Metode yang digunakan untuk proses seleksi yaitu *Elitism*

2. Studi kasus dalam penelitian ini dikhususkan pada Program Studi Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya, Malang.
3. Hasil akhir dari sistem yaitu berupa rekomendasi jadwal sidang skripsi yang optimal dengan penempatan ruang yang sama untuk dosen dengan jadwal sidang pada sesi yang berlanjut.
4. Pengujian dalam penelitian ini hanya berfokuskan di pengujian metode, tidak melakukan pengujian kinerja *hardware*.
5. Optimasi penempatan ruang dilakukan saat tidak terdapat proses perkuliahan berlangsung.

1.6 Sistematika pembahasan

Penelitian ini menggunakan sistematika penyusunan laporan dengan kerangka sebagai berikut:

BAB I Pendahuluan

Bab ini berisi latar belakang, identifikasi masalah, rumusan masalah, tujuan, manfaat, serta batasan masalah yang digunakan dalam penelitian ini.

BAB II Landasan Kepustakaan

Bab ini berisi kajian kepustakaan serta dasar teori yang digunakan dan mendukung implementasi metode Algoritme Genetika untuk optimasi penempatan ruang sidang skripsi.

BAB III Metodologi

Bab ini berisi metode dan langkah kerja yang digunakan dalam membangun sistem optimasi penempatan ruang sidang skripsi menggunakan metode Algoritme Genetika.

BAB IV Algoritme

Bab ini membahas tentang perancangan dari sistem optimasi penempatan ruang sidang skripsi menggunakan metode Algoritme Genetika.

BAB V Implementasi

Bab ini berisi tentang proses implementasi dan pembahasan sistem optimasi penempatan ruang sidang skripsi menggunakan metode Algoritme Genetika.

BAB VI Pengujian dan Pembahasan

Bab ini berisi tentang rencana pengujian dan analisis terhadap hasil dari pengujian sistem optimasi penempatan ruang sidang skripsi menggunakan metode Algoritme Genetika.

BAB VII Penutup

Bab penutup membahas kesimpulan yang didapatkan dari implementasi dan pengujian Algoritme Genetika untuk permasalahan penempatan ruang sidang skripsi serta saran-saran yang mendukung pengembangan penelitian selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Algoritme genetika sudah banyak digunakan untuk menyelesaikan permasalahan penjadwalan. Penelitian dalam masalah penjadwalan yang pernah dilakukan sebelumnya diantaranya yaitu Implementasi Algoritma Genetika Pada Penjadwalan Sidang Tugas Akhir Fakultas Informatika (Ardhianto, et al., 2014), Penjadwalan Sidang Tugas Akhir Prodi Ilmu Komputasi Universitas Telkom Menggunakan Metode Algoritma Genetika Adaptif dan Fuzzy Relation (Pramudita, et al., 2016), Rancang Bangun Sistem Penjadwalan Sidang Skripsi Menggunakan Algoritma Genetika (Adnyana & Wijayana, 2017), dan lain-lain. Dalam proses penjadwalan, algoritme ini memiliki operator genetika yang beragam. Mulai dari representasi kromosom, metode *crossover*, metode mutasi, dan metode seleksi. Beberapa jenis representasi kromosom yang biasa digunakan diantaranya representasi biner, integer, *real code*, dan permutasi. Representasi biner dilakukan dengan cara setiap gen pada kromosom berupa bit biner 1 dan 0. Representasi ini banyak digunakan karena mempermudah pemrograman didalam proses crossover dan mutasi. Penelitian yang pernah menggunakan representasi ini yaitu penelitian milik Pramudita, et al. (2016). Namun pendekatan ini memiliki kelemahan yaitu pada optimasi fungsi yang kompleks dan membutuhkan banyak generasi, proses transformasi biner ke bilangan desimal (real) dan sebaliknya akan membutuhkan waktu komputasi yang lebih lama (Mahmudy, 2015). Representasi integer dilakukan dengan cara setiap gen pada kromosom berupa angka integer. Angka tersebut merupakan bilangan bulat yang menjadi representasi solusi. Pada representasi integer, nilai gen pada satu kromosom boleh bernilai sama. Penelitian sebelumnya pernah dilakukan oleh Ardhianto, et al. (2014). Beberapa penelitian sebelumnya yaitu Adnyana & Wijaya (2017 Luber, et al. (2017), Husada, et al. (2018) menggunakan representasi kromosom permutasi. Permutasi dilakukan dengan cara setiap gen pada kromosom berupa angka integer yang menyatakan nomer atau kode dari tiap simpul (Mahmudy, 2015). Nomer tersebut berisikan *value* dari masing-masing nomer, dapat berupa *value* numerik maupun *value* string. Pada representasi permutasi ini, nilai gen pada satu kromosom tidak boleh bernilai sama.

Beberapa metode *crossover* yang pernah dilakukan dalam penelitian sebelumnya yaitu *one-cut-point* dan *uniform crossover*. Pada proses *crossover* penelitian milik Ardhianto, et al. (2014) menggunakan metode seragam *(uniform)*. Pada metode ini, satu induk akan menghasilkan satu keturunan. Pada proses pembentukan keturunan akan menggunakan pola yang sama pada seluruh induk. Dengan representasi biner, bit yang bernilai sama akan bernilai 0, dan bit yang bernilai berbeda akan bernilai 1. Metode lain yang bisa digunakan yaitu *one-cut-point* yang pernah dilakukan pada penelitian milik Luber, et al. (2017) dan Husada, et al. (2018). Metode *one-cut-point* dilakukan dengan cara memilih secara acak dua individu untuk menjadi induk *(parent)* secara acak dari populasi (Mahmudy, 2015). Metode ini memilih satu titik potong secara acak untuk menghasilkan

keturunan (*offspring)* dengan menukarkan bagian kanan dari masing-masing induk. Satu kali proses *crossover* ini akan menghasilkan dua buah keturunan atau anak.

Pada proses mutasi terdapat beberapa metode yang bisa digunakan diantaranya *reciprocal exchange, insertion, creep,* dan *swap.* Pada penelitian milik Luber, et al. (2017) menggunakan metode *reciprocal exchange*. Metode *reciprocal* merupakan metode mutasi yang paling sederhana (Mahmudy, 2015). Proses mutasi dilakukan dengan memilih dua titik *(exchange point* / XP) secara acak kemudian nilai kromosom akan saling ditukar pada posisi tersebut. Metode lain yaitu *insertion mutation* yang pernah dilakukan pada penelitian milik Husada, et al. (2018). Proses mutasi dilakukan dengan memilih satu titik (*selected point* / SP) secara acak lalu nilai tersebut diambil dan disisipkan pada posisi lain (*insertion point* / IP) secara acak. Metode lain yaitu *creep mutation* dan *swap mutation*. Metode mutasi ini pernah dilakukan pada penelitian milik Ardhianto, et al. (2014). Metode *creep* adalah metode mutasi dengan mengurangi atau menambah nilai dari satu atau beberapa gen dengan angka acak kecil. Sedangkan metode *swap* dilakukan dengan cara menukar nilai gen yang dipilih secara acak dengan nilai gen setelahnya. Jika gen yang akan ditukar berada di akhir kromosom, maka akan ditukar dengan nilai gen yang pertama.

Algoritme genetika memiliki beberapa metode seleksi yang bisa diterapkan diantaranya *roulette wheel, rank based,* dan *elitism*. Pada metode *roulette wheel* dan *rank based* pernah dilakukan pada penelitian milik Ardhianto, et al. (2014) dan Adnyana & Wijayana (2017). *Roulette wheel* yaitu metode seleksi dengan menghitung nilai probabilitas seleksi *(prob*) pada masing-masing individu berdasarkan nilai *fitness*nya. Selanjutnya menghitung nilai probabilitas kumulatif *(probCum)* yang digunakan pada proses seleksi tiap individu. Pada pendekatan ini, individu dengan *fitness* lebih besar akan memiliki peluang terpilih lebih besar, namun tidak menjamin jika individu terbaik akan selalu terpilih untukk masuk generasi selanjutnya (Mahmudy, 2015). Sedangkan pada pendekatan *rank based* semua individu diurutkan berdasarkan nilai *fitness* dari terkecil sampai terbesar. Jika pada *roullete wheel* nilai optimalnya merupakan fungsi fitness dari masing – masing individu, berbeda dengan metode *rank-based* dimana nilai nilai optimalnya bergantung pada *ranking*. Hal tersebut menguntungkan individu dengan *fitness* kurang optimal namun kekurangannya yaitu semua individu dianggap sama tidak ada individu terbaik dan terburuk. Metode seleksi lain yang sering dipakai yaitu *elitism*. Metode ini pernah diterapkan pada penelitian Pramudita, et al. (2016), Adnyana & Wijayana (2017), Luber, et al. (2017), dan Husada, et al. (2018). Pendekatan ini dilakukan dengan memilih individu sebanyak populasi dengan nilai *fitness* tertinggi yang akan masuk generasi berikutnya. Metode ini menjamin individu terbaik yang akan lolos. Kelemahan dari seleksi *elitism* yaitu pada individu yang nilai *fitness*nya rendah tidak diberikan kesempatan untuk bereproduksi. Pada beberapa kasus, solusi optimal justru diperoleh dari hasil reproduksi individu dengan nilai *fitness* rendah (Mahmudy, 2015)

Tabel 2.1 Kajian Pustaka

No	Pustaka	Objek	Metode	Karakteristik	Operator Genetika	Output
1.	Bagas Ardhianto, Bambang Pudjoatmodjo, Mahmud Dwi Suliiyo, 2014	Penjadwalan Sidang Tugas Akhir	Algoritma Genetika	- Proses optimasi penjadwalan sidang tugas akhir dilakukan untuk 2 minggu yaitu senin sampai jumat, dan pada hari jumat hanya slot waktu pagi saja - Terdapat 2 *constraint* yang digunakan yaitu tidak ada bentrok antara jadwal sidang dengan jadwal dosen penguji, dosen penguji harus sesuai dengan topik tugas akhir yang disidangkan	- Representasi kromosom yang digunakan yaitu integer - Menggunakan metode *uniform crossover* untuk proses *crossover* - Menggunakan 2 metode yaitu *creep mutation* dan *swap mutation* untuk proses mutasi - Menggunakan metode seleksi *roulette wheel* dan *rank*	Rekomendasi jadwal sidang tugas akhir pada periode sidang antara Mei sampai dengan Oktober 2012.
2.	Oki Virgiawan Pramudita, Fhira Nhita, Annisa Aditsania, 2016	Penjadwalan Sidang Tugas Akhir	Algoritma Genetika Adaptif dan Fuzzy Relation	- Terdapat 2 *constraint* yang digunakan yaitu tidak ada bentrok antara jadwal sidang dengan jadwal dosen penguji, dosen penguji harus sesuai kelompok bidang tugas akhir mahasiswa	- Representasi kromosom yang digunakan yaitu biner - Metode *crossover* dan mutasi yang digunakan tidak dijelaskan dalam jurnal ini - Metode seleksi yang digunakan yaitu *elitsm*	Rekomendasi jadwal sidang tugas akhir dengan mempertimbangkan seberapa cocok dosen penguji dalam menguji sesuai dengan bidang dari tugas akhir mahasiswa.

3.	I Made Budi Adnyana, I Komang Wijayana, 2017	Penempatan ruang sidang skripsi	Algoritma Genetika	- Panjang kromosom yang digunakan yaitu 60 berdasarkan jumlah mahasiswa yang akan dijadwalkan pada satu penjadwalan tersebut - Terdapat 6 *constraint* yang digunakan yaitu dalam satu sesi sidang skripsi mahasiswa terdapat satu orang dosen pembimbing dan dua orang dosen penguji, jadwal menguji dari dosen pembimbing maupun dosen penguji tidak boleh benturan dengan jadwal mengajarnya masing-masing, sidang skripsi harus dijadwalkan pada ruangan atau kelas yang kosong (tidak ada perkuliahan), durasi sidang skripsi adalah 1.5 jam, jika ada dosen yang menguji lebih dari satu kali secara bersambung maka diusahakan jadwal sidangnya pada ruangan	- Representasi kromosom yang digunakan yaitu permutasi - Metode *crossover* dan mutasi yang digunakan tidak dijelaskan dalam jurnal ini - Metode seleksi yang digunakan yaitu membandingkan metode *elitsm, roulette,* dan *rank*	Rekomendasi jadwal sidang skripsi yang optimal dengan nilai fitness 1 pada STIKOM Bali.

				yang sama, jadwal menguji dari dosen penguji tidak boleh ada yang benturan dengan jadwal sidang pada sesi lainnya		
4.	M. Mart Hans Luber, Imam Cholissodin, Candra Dewi, 2017	Penjadwalan Damping Mahasiswa Difabel	Algoritma Genetika	- Proses optimasi penjadwalan damping mahasiswa difabel dilakukan untuk lima hari jam kerja senin sampai jumat - Panjang kromosom yang digunakan yaitu 25 berdasarkan 5 hari dikalikan 5 slot waktu untuk satu harinya - Terdapat 2 *constraint* yang digunakan yaitu seorang pendamping dalam satu hari dan waktu yang tidak bisa mendampingi pada waktu tersebut, seorang pendamping berada pada kelas yang berbeda salam hari dan waktu yang sama	- Representasi kromosom yang digunakan yaitu permutasi - Menggunakan metode *one cut-point* untuk *crossover* - Menggunakan metode *reciprocal exchange mutation* untuk mutasi - Menggunakan metode seleksi *elitsm*	Rekomendasi jadwal damping mahasiswa difabel 5 waktu dalam sehari dengan hari pendampingan yaitu Senin sampai Jumat dan setiap volunteer bertanggung jawab mendampingi mahasiswa difabel minimal 3 kali dalam seminggu.

5.	Holiyanda Husada, Imam Cholissodin, Fitra A Bachtiar, 2018	Penjadwalan Kuliah Pengganti	Algoritma Genetika	- Proses optimasi penjadwalan kuliah pengganti dilakukan untuk lima hari jam kerja senin sampai jumat - Panjang kromosom yang digunakan berdasarkan banyak ruang kosong yang tersedia - Terdapat 4 *hard constraint* yang digunakan yaitu dosen tidak boleh mengajar dalam 2 perkuliahan dalam satu waktu, mahasiswa tidak boleh bentrok dengan jadwal kuliah utama, perkuliahan pengganti harus menggunakan ruangan kosong sesuai jumlah sks, dosen tidak boleh bentrok dengan jadwal mengajar utama, dan 1 *soft constraint* yaitu peminjaman ruang untuk perkuliahan malam hari harus melapor pada bagian akademik	- Representasi kromosom yang digunakan yaitu permutasi - Menggunakan metode *one cut-point* untuk *crossover* - Menggunakan metode *insertion point* untuk mutasi - Menggunakan metode *elitsm* untuk proses seleksi	Rekomendasi jadwal kuliah pengganti untuk 3 mata kuliah pengganti yaitu Swarm Intelligence, Algoritma Evolusi dan Big Data dengan ketentuan hari pengganti kuliah yaitu dari Senin sampai dengan Jumat.

6.	Penulis	Penjadwalan Sidang Tugas Akhir	Algoritme Genetika	- Proses optimasi penempatan ruang sidang dilakukan untuk satu hari penjadwalan - Panjang kromosom yang digunakan berdasarkan jumlah mahasiswa yang akan dijadwalkan pada hari tersebut - Terdapat 3 *hard constraint* yang digunakan yaitu tidak boleh terdapat jadwal yang sama dalam satu hari, dua dosen penguji atau dua dosen pembimbing tidak boleh bentrok pada waktu sidang yang sama, dua dosen penguji atau dua dosen pembimbing dengan sesi beruntun harus pada ruang sidang yang sama	- Representasi kromosom yang digunakan yaitu permutasi - Proses *crossover* yang digunakan yaitu metode *one cut-point* - Mutasi yang digunakan yaitu metode *random mutation* - Menggunakan metode seleksi *elitsm*	Rekomendasi jadwal sidang tugas akhir dengan masa periode sidang antara Februari sampai dengan Juni 2019.

2.2 Optimasi

Optimasi dapat didefinisikan sebagai proses untuk menemukan kondisi yang memberikan nilai minimum dan maksimum dari sebuah fungsi, peluang, maupun pencarian nilai lainnya dalam berbagai kasus. Optimasi sangat berguna dI berbagai jenis bidang diantaranya ekonomi, arsitektur, jaringan komputer, telekomunikasi, transportasi, perdagangan, dan lain-lain. Tujuan akhir dari proses optimasi yaitu meminimalkan usaha (*effort*) atau memaksimalkan manfaat yang diinginkan (Budi, 2013).

2.3 Penjadwalan

Penjadwalan merupakan suatu proses dalam melakukan menjadwalkan atau memasukkan suatu kegiatan dalam sebuah jadwal. Penjadwalan dalam kehidupan sehari-hari adalah hal yang umum ditempat kerja, dengan tujuan agar tugas dapat terselesaikan dengan cepat yaitu dengan membagi sumber daya secara tepat (Muhyi, 2008). Penjadwalan akan memutuskan proses yang harus berjalan, kapan dan berapa lama proses itu berjalan (Auliyah, et al., 2018).

Pembuatan jadwal secara otomatis merupakan tugas yang sangat penting karena dapat menghemat waktu kerja karyawan pada suatu perusahaan dan institusi, meningkatkan produktivitas dalam bekerja, dalam pemenuhan constraints diberikan solusi yang optimal dan cepat, kualitas pelayanan, kualitas pendidikan, dan juga kualitas hidup (Luber, et al., 2017). Dalam penempatan ruang sidang skripsi juga terdapat constraints yang berarti syarat atau ketentuan. Fungsi dari constraints untuk menyelesaikan masalah penjadwalan adalah sebagai aturan atau syarat ketentuan agar tidak terjadi bentrok dalam penyusunan jadwal (Mariana & Hiryanto, 2013).

Salah satu tujuan adanya penjadwalan yaitu untuk memperkecil keterlambatan pada suatu pekerjaan yang memiliki batas waktu penyelesaian, sehingga dapat mengurangi biaya denda. Penjadwalan juga mempunyai peranan memberi pedoman pada unit pekerjaan terkait batasan waktu dalam memulai dan mengakhiri suatu tugas (Auliyah, et al., 2018).

2.4 Penempatan ruang sidang skripsi di Filkom

Penempatan ruang sidang, waktu sidang, dan penentuan dosen penguji di Fakultas Ilmu Komputer Universitas Brawijaya dilakukan oleh sekretaris jurusan. Dosen penguji untuk sidang ditentukan langsung oleh sekretaris jurusan dengan melihat keminatan jurusan program studi dari mahasiswa dan keterkaitan dosen penguji dengan topik skripsi yang diambil oleh mahasiswa. Dalam sidang, akan dihadiri oleh dua dosen pembimbing dari mahasiswa dan dua dosen penguji. Dalam sehari, dosen dapat menguji lebih dari satu sidang diwaktu yang berbeda. Intensitas pengujian sidang setiap dosen akan berbeda-beda tergantung banyak mahasiswa dari program studi masing-masing di satu periode sidang.

Dalam penempatan ruang sidang skripsi, terdapat batasan *(constraint)* yaitu berupa *hard constraint* dan *soft constraint*. *Hard constraint* yaitu batasan yang nilainya mutlak dan harus dipatuhi. Sedangkan *soft constraint* yaitu batasan yang nilainya fleksibel, boleh dipatuhi atau tidak, namun jika tidak dipatuhi akan tetap diberikan nilai pelanggaran atau penalti yang bobotnya lebih kecil dibandingkan nilai penalti dari *hard constraint.*

2.5 Algoritme Genetika

Algoritme Genetika (GA) yaitu salah satu cabang Algoritma Evolusi (EA) yang kerap kali digunakan untuk mendapatkan solusi dari masalah yang kompleks. Konsep dari algoritma genetika meniru dari konsep evolusi alami pada sebuah populasi individu. Algoritma genetika banyak digunakan dalam menyelesaiakan permasalahan pada ilmu fisika, biologi, ekonomi, dan ilmu lain yang memerlukan metode optimasi dengan model matematika yang kompleks (Mahmudy, 2015). Algoritma Genetika kadang tidak dapat menemukan hasil yang paling baik, namun sering kali dapat menyelesaikan permasalahan dengan cukup baik bahkan mendekati yang terbaik.

Penerapan algoritma genetika dalam memecahkan suatu masalah dimulai dengan pemetaan (*encoding*) pada suatu solusi dari masalah menjadi *string chromosome*. *String chromosome* ini terdiri dari gen-gen yang menggambarkan variabel-variabel keputusan yang digunakan dalam solusi. Representasi *string chromosome* dan fungsi *fitness* digunakan untuk menilai kualitas dari sebuah *chromosome* ditempatkan dalam algoritma genetika. Dari beberapa kasus sebelumnya, telah dibuktikan bahwa kualitas solusi yang dihasilkan bergantung pada bagaimana merepresentasikan sebuah solusi menjadi *chromosome* (Mahmudy, 2015).

Proses Algoritme Genetika diawali dari inisialisasi, reproduksi, evaluasi, dan seleksi (Mahmudy, 2015). Proses reproduksi akan dibagi menjadi dua tahap yaitu proses *crossover* dan mutasi. Berikut penjelasan tahap-tahap pada Algoritme Genetika.

2.5.1 Inisialisasi

Proses inisialisasi adalah proses pembentukan generasi awal. Inisialisasi dilakukan dengan tujuan untuk membangkitkan himpunan dari solusi awal secara acak berisi bait *string chromosome* dan diletakkan pada populasi. Ukuran populasi (*popSize*) harus ditentukan terlebih dahulu dalam tahap ini. Nilai tersebut akan menyatakan banyaknya *chromosome* yang ditempatkan pada populasi. Panjang setiap string *chromosome* ditentukan berdasarkan presisi variabel solusi yang akan dicari. Banyaknya individu di setiap populasi ditentukan dari nilai Popsize (Mahmudy, 2015).

2.5.2 Reproduksi

Proses reproduksi dilakukan untuk menghasilkan individu baru dari hasil penurunan (*offspring*) individu lama yang ada pada populasi. Himpunan keturunan

itu diletakkan dalam penampungan *offspring*. Dua metode yang digunakan dalam proses ini adalah tukar silang (*crossover*) dan mutasi (*mutation*) (Mahmudy, 2015).

2.5.2.1 *Crossover*

Crossover adalah proses reproduksi yang dilakukan dengan cara menukar nilai gen dari dua individu yang terpilih menjadi *parent* secara *random* dan akan menghasilkan keturunan (*offspring*) yang disebut dengan anak *(child)*. Jumlah anak yang akan dihasilkan berdasarkan pada nilai dari *Crossover rate (Cr*). Nilai *Cr* dibangkitkan antara 0 sampai dengan 1. Nilai *Cr* akan menyatakan rasio *offspring* yang dihasilkan oleh proses *crossover* terhadap ukuran populasi (Mahmudy, 2015). Formulasi mencari jumlah *offspring* dapat dilihat pada Persamaan 2.1.

$$Offspring = Cr\ x\ Popsize \quad \textbf{(2.1)}$$

Keterangan :

$Offspring$ = keturunan/anak
Cr = nilai *crossover rate*
$Popsize$ = ukuran populasi

Dari Persamaan 2.1, misal dibangkitkan nilai *Cr* =0.5 dan popSize sebanyak 4, maka akan dihasilkan 0.5 x 4=2 buah *offspring* dari proses tukar silang. Salah satu metode *crossover* yang dapat digunakan yaitu *one-cut-point*.

Pada penelitian ini, akan digunakan metode yaitu *one-cut-point crossover.* Metode ini dilakukan dengan cara mengambil secara acak 2 individu yang akan menjadi *parent.* Kemudian akan ditentukan titik potong *(cut-point)* dari susunan gen kromosom tersebut secara acak. Titik potong akan digunakan sebagai titik pertukaran gen yang akan dilakukan saat proses *crossover.* Proses *crossover* dengan *one-cut-point* digambarkan pada Gambar 2.1. Misal terpilih parent P1 dan P3, maka akan menghasilkan *offspring* yaitu C1 dan C2 (Mahmudy, 2015).

P_1 [0 0 1 1]
P_3 [1 0 0 1]
C_1 [0 0 0 1]
C_2 [1 0 1 1]

Gambar 2.1 Proses Crossover
Sumber : (Mahmudy, 2015)

2.5.2.2 Mutasi

Mutasi yaitu proses reproduksi yang dilakukan dengan cara memilih satu buah induk secara acak dari populasi yang akan menjadi *parent* yang akan menghasilkan anak *(child)* dengan cara mengubah susunan kromosom sehingga berbeda antara induk dan anak. Jumlah *child* yang akan dihasilkan berdasarkan pada nilai dari *Mutation rate (Mr*). Nilai *Cr* dibangkitkan antara 0 sampai dengan

1. Nilai *Mr* akan menyatakan rasio *offspring* yang dihasilkan oleh proses mutasi terhadap ukuran populasi (Mahmudy, 2015). Formulasi mencari jumlah *offspring* dapat dilihat pada Persamaan 2.2.

$$Offspring = Mr\ x\ Popsize \quad \textbf{(2.2)}$$

Keterangan :

$Offspring$ = keturunan/anak
Mr = nilai *mutation rate*
$Popsize$ = ukuran populasi

Dari Persamaan 2.2, misal dibangkitkan nilai *Mr* =0.2 dan popSize sebanyak 4, maka akan dihasilakn 0.2 x 4=0.8 (dilakukan pembulatan ke atas menjadi 1) buah *offspring* dari proses mutasi. Pada proses mutasi, 1 *parent* hanya menghasilkan 1 *child.* Jika pada proses mutasi dibutuhkan 2 *child,* maka akan dilakukan mutasi sebanyak 2 kali hingga memenuhi jumlah *offspring.*

Pada penelitian ini, proses mutasi yang digunakan yaitu dengan cara mengubah salah satu nilai gen yang dipilih secara acak. Proses mutasi ini digambarkan pada Gambar 2.2. Misal terpilih parent P4, maka akan menghasilkan *offspring* ke-3 yaitu C3 (Mahmudy, 2015).

P_4 [0 1 0 1]
C_3 [0 1 0 0]

Gambar 2.2 Proses Mutasi
Sumber : (Mahmudy, 2015)

2.5.3 Evaluasi

Proses evaluasi bertujuan untuk menghitung nilai kebugaran (*fitness*) setiap kromosom. Semakin besar nilai *fitness* yang dihasilkan maka semakin baik kromosom tersebut untuk dijadikan calon solusi (Mahmudy, 2015). Nilai *fitness* adalah nilai yang menjadi tolak ukur baik tidaknya suatu solusi (individu).

Nilai *fitness* yaitu representasi permasalahan yang diangkat sehingga akan dimiliki oleh setiap kromosom. Nilai *fitness* dihasilkan dari seberapa banyak pelanggaran atau penalti terhadap *constraint* yang telah ditetapkan sebelumnya. Semakin banyak jumlah penalti, maka akan semakin kecil nilai *fitness* kromosom tersebut. Formula menghitung nilai fitness dituliskan dalam Persamaan 2.3.

$$Fitness = \frac{1}{1 + (FB_1 + FB_2 + \cdots + FB_n)} \quad \textbf{(2.3)}$$

Keterangan :

B_i = nilai pelanggaran ke-*i*
F = pelanggaran
n = jumlah *constraint*

2.5.4 Seleksi

Proses seleksi merupakan tahapan pemilihan individu yang akan bertahan untuk kembali menjadi *parent* dan masuk proses *crossover* ataupun mutasi. Proses seleksi ini bertujuan untuk mencari individu terbaik dalam satu generasi. *Parent* yang baik akan menghasilkan keturunan yang baik walaupun tidak selalu seperti itu (Mahmudy, 2015).

Pada penelitian ini, metode seleksi yang digunakan yaitu Elitism. Eltism adalah metode seleksi terpopuler dalam Algoritme Genetika karena pengimplementasiannya sederhana dan cukup efisien. Pada seleksi dengan metode elitism, sejumlah n individu dengan *fitness* paling baik akan dipilih. kumpulan Individu tersebut akan menjadi parent pada generasi berikutnya (Mahmudy, 2015).

2.5.5 Kondisi Berhenti

Algoritme Genetika adalah algoritme yang bersifat *iterative* dalam pencarian solusi. Iterasi pada Algoritme Genetika dilakukan berulang terus sampai kondisi berhenti tercapai. Beberapa kriteria yang dapat digunakan untuk kondisi berhenti diantaranya (Mahmudy, 2015):

1. Iterasi akan berhenti sampai generasi *n,* dimana nilai *n* sudah didefinisikan sebelumnya berdasarkan eksperimen pendahuluan. Semakin tinggi dan kompleks masalah, maka akan semakin besar nilai *n*. Nilai *n* akan ditentukan sedemikian rupa sehingga konvergensi populasi tercapai dan sulit didapatkan solusi yang lebih baik setelah iterasi ke-*n*.
2. Iterasi akan berhenti setelah *n* generasi yang berurutan tidak ditemukan solusi yang lebih baik dari iterasi sebelumnya. Kondisi ini akan menunjukkan bahwa Algoritme Genetika sulit mendapatkan solusi yang lebih baik.
3. Iterasi akan berhenti berdasarkan satuan waktu yang telah ditentukan sebelumnya. Hal tersebut dapat digunakan untuk membandingkan performa dari beberapa algoritme.

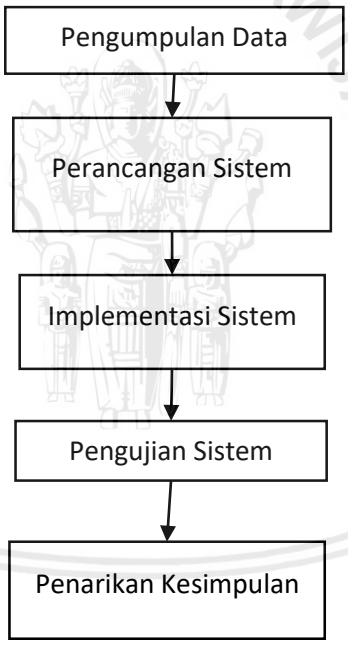
BAB 3 METODOLOGI

3.1 Tipe Penelitian

Tipe penelitian yang digunakan dalam penelitian ini yaitu non-implementatif analitik (*analytical/explanatory*). Pada penelitian ini, dikatakan non-implementatif karena memfokuskan pada kegiatan analisis dengan menjelaskan hubungan antar elemen objek penelitian pada fenomena yang terjadi. Fenomena yang ingin dianalisis pada penelitian ini yaitu performa dari Algoritme Genetika untuk permasalahan penempatan ruang sidang skripsi.

3.2 Strategi Penelitian

Pada tahap strategi penelitian akan dijelaskan tahapan-tahapan yang akan dilakukan dalam pembuatan sistem. Terdapat beberapa tahapan diantaranya pengumpulan data, perancangan sistem, implementasi sistem, pengujian sistem, dan penarikan kesimpulan. Tahapan ini dijabarkan pada Gambar 3.1.



Gambar 3.1 Alur strategi penelitian

3.3 Pengumpulan Data

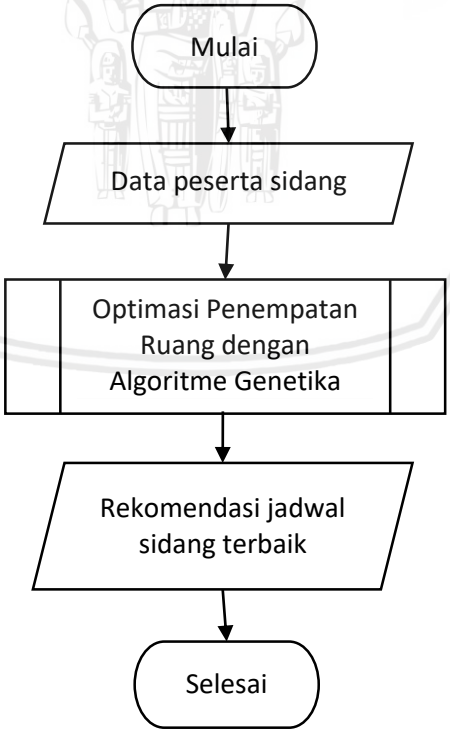
Data yang digunakan dalam penelitian ini yaitu data mahasiswa yang mendaftar sidang skripsi, dosen pembimbing dan dosen penguji, ruang kelas dan waktu yang tersedia untuk sidang skripsi dari Fakultas Ilmu Komputer Universitas Brawijaya. Data-data tersebut yang akan dijadikan sebagai data dalam optimasi proses penempatan ruang. Data diambil pada tanggal 26 September 2018 dan data ruang sidang yang digunakan yaitu ruang pada semester ganjil 2018/2019.

3.4 Perancangan Sistem

Perancangan sistem merupakan gambaran alur dari optimasi penempatan ruang sidang skripsi menggunakan algoritme genetika untuk mempermudah implementasi, pengujian dan analisis dari sistem. Tahapan-tahapan yang dilakukan dalam perancangan sistem ini diantaranya diagram alir sistem, arsitektur algoritme genetika pada perhitungan manual, perhitungan manual, dan perancangan pengujian.

3.5 Implementasi Sistem

Pada tahap implementasi sistem ini, data yang telah diperoleh akan diimplementasi pada sistem menggunakan Algoritme Genetika yang telah dibuat. Sistem dibuat dengan memasukkan data peserta sidang terlebih dahulu. Data mahasiswa yang mendaftarkan sidang tersebut lalu diproses menggunakan Algoritme Genetika. Pada tahapan penjadwalan dengan Algoritme Genetika diawali dengan menentukan input dari sistem terlebih dahulu. Selanjutnya, dilakukan proses inisialisasi populasi awal, proses *crossover*, mutasi, evaluasi dengan mencari nilai *fitness* dari masing-masing kromosom, seleksi menggunakan metode *elitsm* kemudian output yang dihasilkan sistem yaitu berupa kromosom terbaik yang merupakan solusi jadwal dengan penempatan ruang yang paling optimal yang dihasilkan sistem. Alur gambaran umum sistem dapat dilihat pada Gambar 3.2.



Gambar 3.2 Gambaran umum sistem

Implementasi akan menggunakan bahasa pemrograman Python. Adapun perangkat keras yang digunakan dalam pembuatan sistem ini berupa :

1. Laptop *processor* AMD Dual Core A9 3.6 Hz
2. *Memory* 4 GB
3. *Hardisk* 500 GB

Perangkat lunak pendukung dalam implementasi sistem diantaranya :

1. *Microsoft Windows* 10
2. *Python* 3.6

3.6 Pengujian Sistem dan Analisis

Pada tahap ini dilakukan pengujian yang bertujuan mengukur seberapa baik hasil optimasi akurasi yang dilakukan sistem. Pengujian yang akan dilakukan pada penelitian ini yaitu pengujian pada nilai parameter-parameter Algoritma Genetika. Pengujian dilakukan untuk mengetahui nilai parameter dari Algoritma Genetika yang dapat menghasilkan solusi yang paling optimal. Pengujian yang akan dilakukan terhadap sistem ini antara lain :

1. Pengujian terhadap kombinasi nilai *crossover rate (cr)* dan *mutation rate (mr).*
2. Pengujian terhadap populasi untuk menentukan ukuran *population size* yang optimal.
3. Pengujian terhadap generasi atau iterasi maksimum.

3.6.1 Perancangan pengujian kombinasi *crossover rate (cr)* dan *mutation rate (mr)*

Perancangan pengujian kombinasi nilai *crossover rate* (*cr*) dan *mutation rate (mr)* digunakan untuk mencari kombinasi nilai dari *cr* dan *mr* yang dapat menghasilkan nilai *fitness* yang optimal dan mengetahui waktu komputasinya. Kombinasi nilai *Cr* dan *Mr* yang digunakan yaitu bilangan dengan rentang antara 0,1 hingga 0,9. Pengujian dilakukan sebanyak 10 kali untuk setiap kombinasi nilai *Cr* dan *Mr*. Perancangan pengujian kombinasi nilai *crossover rate* (*cr*) dan *mutation rate (mr)* dapat dilihat pada Tabel 3.1 dan 3.2.

Tabel 3.1 Perancangan pengujian nilai *fitness* kombinasi nilai *cr* dan *mr*

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

0.9;0.1											

Tabel 3.2 Perancangan pengujian waktu komputasi kombinasi nilai *cr* dan *mr*

Cr;Mr	Pengujian Ke-										Rata-rata Waktu Komputasi
	1	2	3	4	5	6	7	8	9	10	
0.1;0.9											
0.2;0.8											
0.3;0.7											
0.4;0.6											
0.5;0.5											
0.6;0.4											
0.7;0.3											
0.8;0.2											
0.9;0.1											

3.6.2 Perancangan pengujian ukuran populasi

Perancangan pengujian ukuran populasi dilakukan untuk menentukan nilai banyaknya individu dalam populasi. Pengujian ini digunakan untuk menentukan pengaruh ukuran populasi terhadap besarnya nilai *fitness* yang akan dihasilkan dan mengetahui waktu komputasinya. Pengujian ukuran populasi akan dilakukan sebanyak 10 kali uji coba dengan ukuran populasi kelipatan 10, sampai dengan ukuran populasi 100. Perancangan pengujian ukuran populasi dapat dilihat pada Tabel 3.3 dan Tabel 3.4.

Tabel 3.3 Perancangan pengujian nilai *fitness* ukuran populasi

Populasi	Pengujian Ke-										Rata-rata Nilai *Fitness*
	1	2	3	4	5	6	7	8	9	10	
10											
20											
30											
40											
50											
60											
70											
80											
90											
100											

Tabel 3.4 Perancangan pengujian waktu komputasi ukuran populasi

Populasi	Pengujian Ke-										Rata-rata Waktu Komputasi
	1	2	3	4	5	6	7	8	9	10	
10											
20											
30											
40											
50											
60											
70											
80											
90											
100											

3.6.3 Perancangan pengujian ukuran generasi

Perancangan pengujian ukuran generasi digunakan untuk mengetahui jumlah generasi terbaik dalam menghasilkan hasil optimal dalam penjadwalan dan waktu komputasi yang dihasilkan. Pada pengujian ukuran generasi ini dilakukan sebanyak 10 kali dengan ukuran generasi kelipatan 10, sampai dengan 100 generasi. Perancangan pengujian ukuran generasi dapat dilihat pada Tabel 3.5 dan Tabel 3.6.

Tabel 3.5 Perancangan pengujian nilai *fitness* ukuran generasi

Generasi	Pengujian Ke-										Rata-rata Nilai *Fitness*
	1	2	3	4	5	6	7	8	9	10	
10											
20											
30											
40											
50											
60											
70											
80											
90											
100											

Tabel 3.6 Perancangan pengujian waktu komputasi ukuran generasi

Generasi	Pengujian Ke-										Rata-rata Waktu Komputasi
	1	2	3	4	5	6	7	8	9	10	
10											
20											
30											
40											
50											
60											
70											
80											
90											
100											

Setelah tahap pengujian selesai, maka akan dilakukan proses analisis untuk menentukan seberapa baik sistem bekerja. Analisis dilakukan dengan cara menentukan nilai *fitness* paling baik sebagai solusi yang dihasilkan oleh sistem dalam proses optimasi penempatan ruang sidang skripsi menggunakan algoritme genetika.

3.7 Penarikan Kesimpulan

Proses penarikan kesimpulan dilakukan setelah semua tahap penelitian selesai dilakukan. Kesimpulan dilakukan untuk menjawab pertanyaan dari rumusan masalah yang telah dirumuskan sebelumnya. Jawaban dari perumusan masalah tersebut harus didasarkan pada hasil implementasi dan pengujian dari sistem. Tahap terakhir dari penarikan kesimpulan adalah pemberian saran yang digunakan dalam memperbaiki kesalahan-kesalahan yang terjadi dalam penitian ini untuk dikembangan dalam penelitian selanjutnya.

BAB 4 ALGORITME

4.1 Formulasi Permasalahan

Pada penentuan jadwal perkuliahan, Fakultas Ilmu Komputer Universitas Brawijaya (FILKOM UB) mempertimbangkan beberapa faktor diantaranya ketersediaan waktu dan ruang. Penentuan jadwal sidang skripsi sendiri juga mempertimbangkan lebih banyak faktor diantaranya waktu dan ruang yang tersedia untuk sidang selain jadwal perkuliahan, waktu ketersediaan dosen penguji dan dosen pembimbing, serta kesesuaian keminatan dosen penguji dengan topik skripsi yang diambil oleh mahasiswa. Pada penelitian ini, faktor utama yang akan diselesaikan yaitu terkait keterlibatan ruang dan waktu pada sidang. Dalam hal ini, dua dosen penguji atau dosen pembimbing yang dijadwalkan sidang tidak boleh bentrok pada waktu yang sama dan jadwal dosen penguji atau pembimbing sidang yang beruntun harus berada pada ruang yang sama.

Dalam penentuan ruang sidang skripsi, terdapat 7 pembagian waktu untuk sidang dan hanya beberapa ruangan di gedung FILKOM UB yang dapat digunakan untuk sidang. Gedung yang digunakan diantaranya Gedung B, E, dan F Total maksimal ruang kelas yang digunakan sebagai ruang sidang skripsi sampai bulan September tahun 2018 yaitu sebanyak 16 ruang. Data ruang kelas dan waktu sidang yang digunakan untuk sidang skripsi pada tahun 2018 dapat dilihat pada Tabel 4.1 dan Tabel 4.2.

Tabel 4.1 Data ruang kelas sidang skripsi tahun 2018

No.	Gedung		
	B	E	F
1	B1.6	E2.1	F3.3
2		E2.2	F3.4
3		E2.3	F3.5
4		E2.4	F4.1
5		E2.5	F4.5
6		E2.7	F4.6
7		E2.8	F4.7
8			F4.8

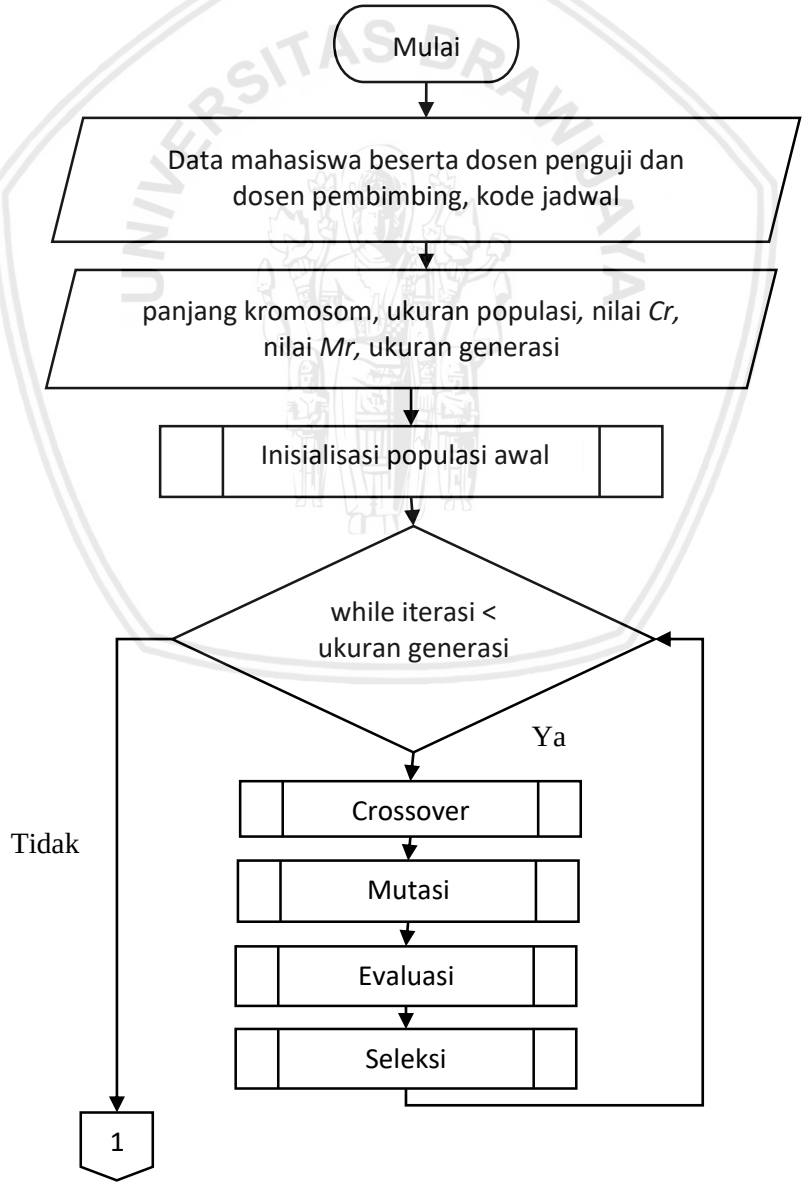
Tabel 4.2 Data pembagian waktu sidang skripsi tahun 2018

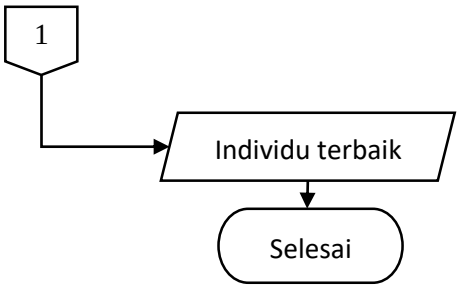
No.	Senin sampai Kamis			Jum'at		
	Waktu	Jam Mulai	Jam Selesai	Waktu	Jam Mulai	Jam Selesai
1	Sesi 1	7:30:00	8:45:00	Sesi 1	8:00:00	9:15:00
2	Sesi 2	8:45:00	10:00:00	Sesi 2	9:15:00	10:30:00
3	Sesi 3	10:00:00	11:15:00	Sesi 3	10:30:00	11:45:00

4	Sesi 4	12:45:00	14:00:00	Sesi 4	12:45:00	14:00:00
5	Sesi 5	14:00:00	15:15:00	Sesi 5	14:00:00	15:15:00
6	Sesi 6	15:15:00	16:30:00	Sesi 6	15:15:00	16:30:00
7	Sesi 7	16:30:00	17:45:00	Sesi 7	16:30:00	17:45:00

4.2 Perancangan Algoritme Genetika

Pada perancangan algoritme genetika, terdapat tahapan-tahapan yang dilakukan diantaranya inisialisasi populasi awal sebagai calon solusi, kemudian reproduksi yang terdiri dari proses *crossover* dan proses mutasi untuk menghasilkan keturunan baru, selanjutnya tahap evaluasi digunakan untuk mendapatkan nilai *fitness* pada setiap kromosom, dan yang terakhir yaitu tahap seleksi untuk mendapatkan individu-individu yang lolos bertahan hidup untuk generasi selanjutnya sekaligus individu terbaik. Diagram alir algoritme genetika dapat dilihat pada Gambar 4.1.

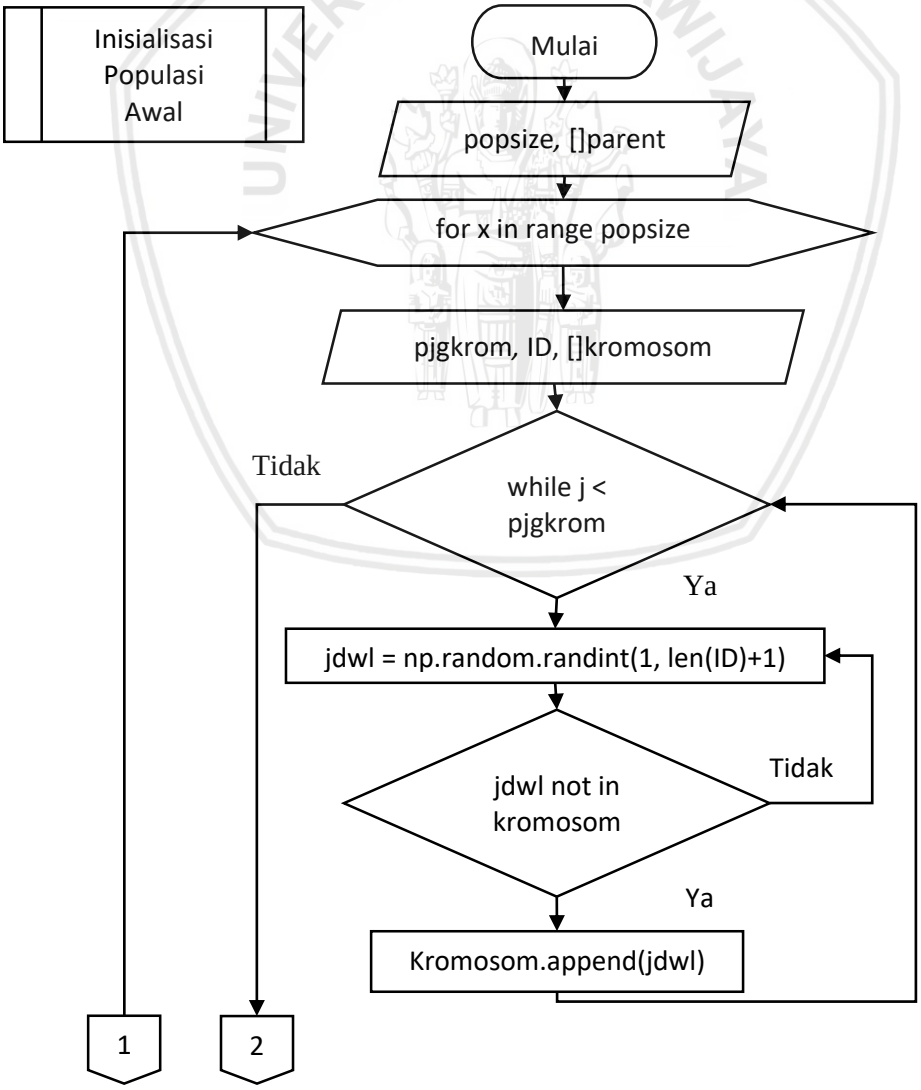


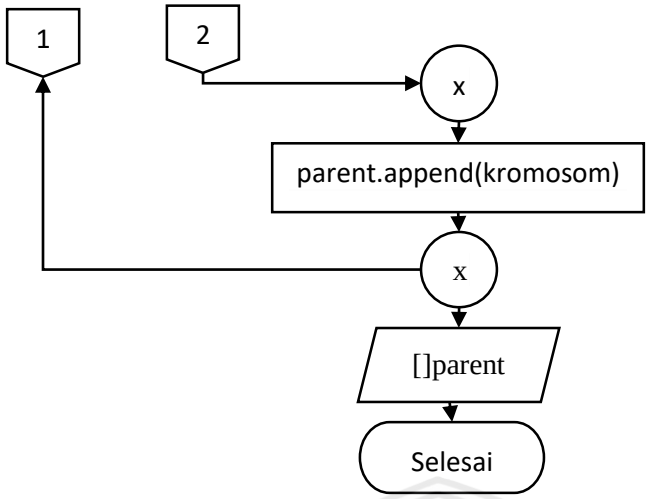


Gambar 4.1 Diagram alir algoritme genetika

4.2.1 Inisialisasi populasi awal

Pada tahap ini akan dilakukan inisialisasi populasi awal yang berisi individu-individu calon solusi. Langkah pertama adalah menentukan *population size* yaitu banyaknya populasi. Langkah selanjutnya adalah membangkitkan nilai gen dari tiap kromosom dengan bilangan integer secara acak yang merepresentasikan kode jadwal sidang. Diagram alir proses inisialisasi populasi awal dapat dilihat pada Gambar 4.2.



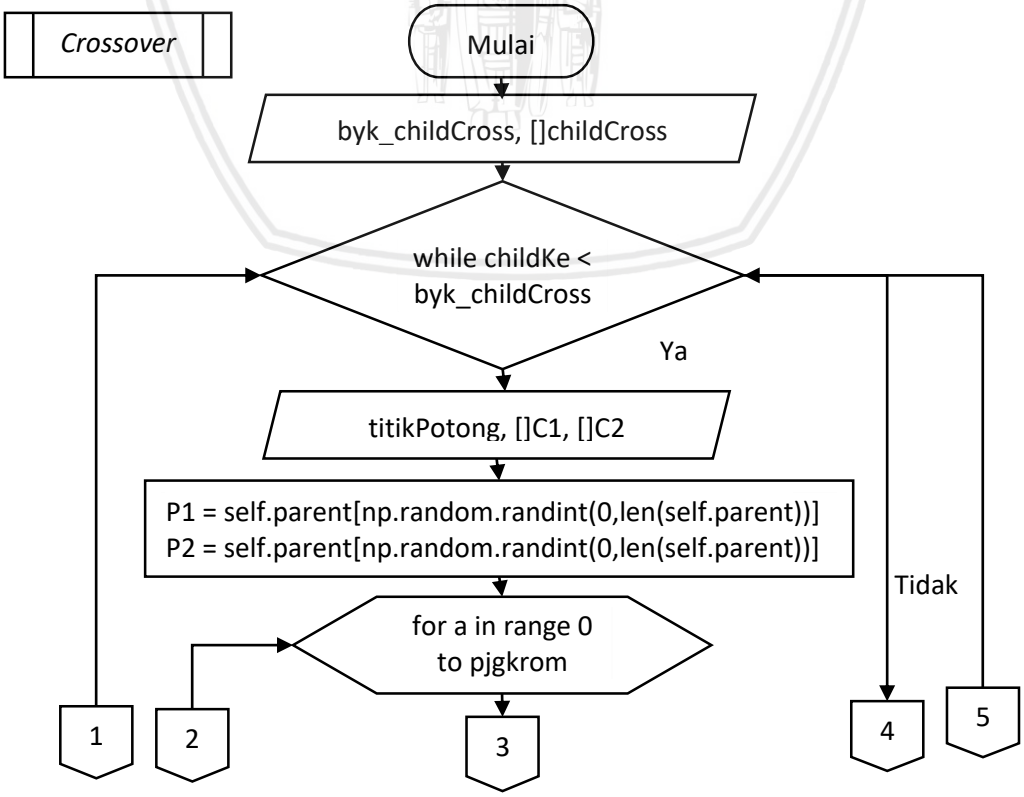


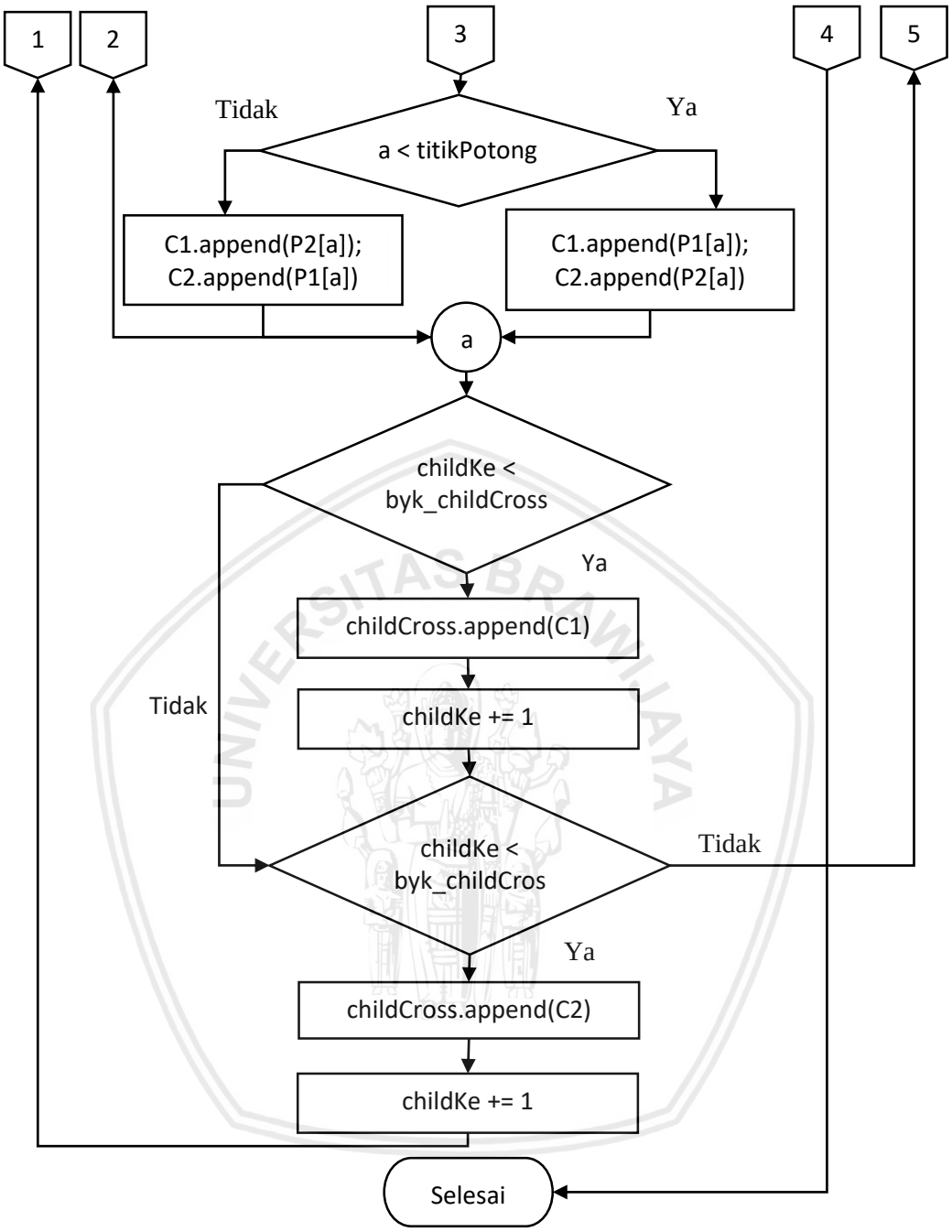
Gambar 4.2 Diagram alir proses inisialisasi populasi awal

Berdasarkan Gambar 4.2, representasi kromosom yang digunakan yaitu permutasi dengan bilangan nilai acak dengan batas maksimum yaitu sejumlah kode jadwal. Perulangan *x* merupakan perulangan yang digunakan untuk membangkitkan individu sebanyak *popsize*, dan perulangan *j* merupakan perulangan yang digunakan untuk menentukan nilai setiap gen sebanyak panjang kromosom.

4.2.2 *Crossover*

Pada proses *crossover*, metode digunakan yaitu *one cut point crossover*. Diagram alir proses *crossover* dapat dilihat pada Gambar 4.3.





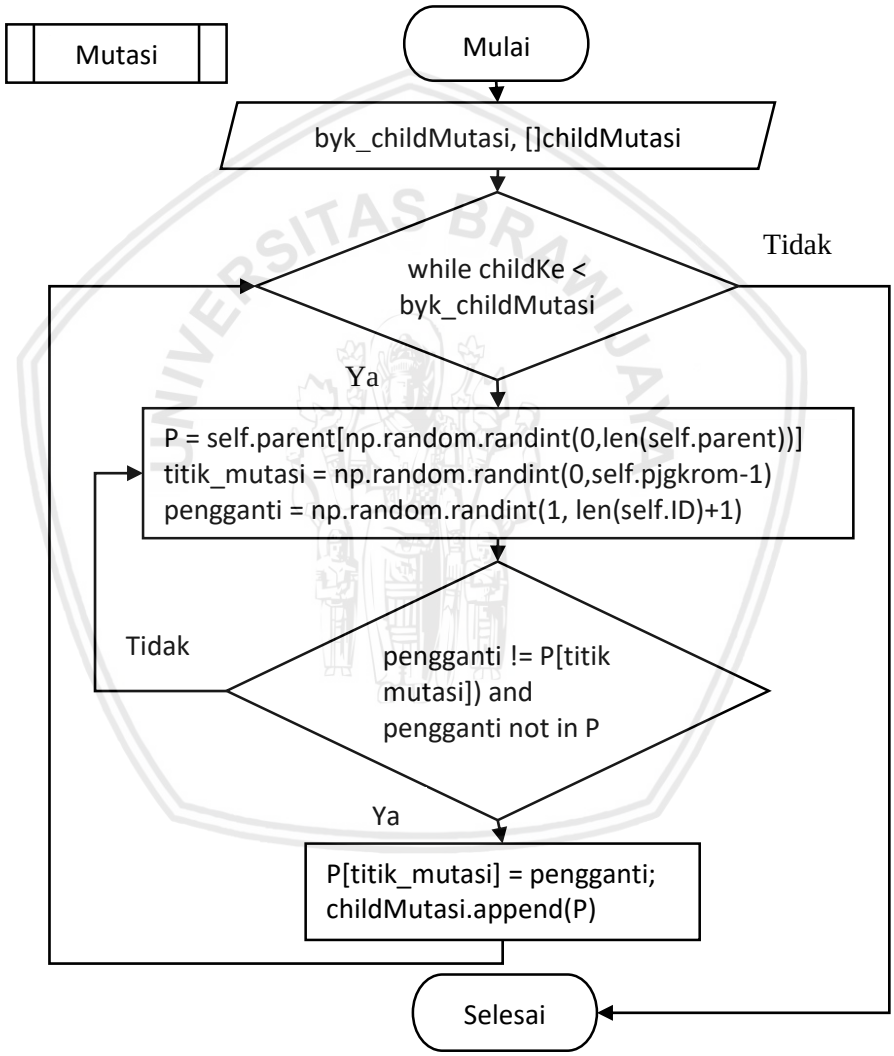
Gambar 4.3 Diagram alir proses *crossover*

Berdasarkan Gambar 4.3 tahap pertama proses *crossover* yaitu menentukan banyak *child* yang akan dihasilkan pada proses *crossover.* Perulangan childKe dilakukan sebanyak anak yang akan dihasilkan, dimana pada langkah ini dilakukan pengisian childCross sejumlah anak hasil reproduksi proses *crossover* yang telah ditentukan. Kemudian menentukan titik potong, dan memilih dua individu secara acak untuk dijadikan sebagai *parent.* Perulangan *a* dilakukan dalam rentang 0 sampai panjang kromosom, dan masuk tahap seleksi kondisi dengan syarat *a* kurang dari titikPotong. Pada proses seleksi kondisi ini akan dilakukan pengisian kromosom dari childCross. Pada kondisi ya, nilai gen index ke 0 sampai

titikPotong individu pertama akan diisikan pada C1 dan nilai gen index ke 0 sampai titikPotong individu kedua akan diisikan pada C2. Sedangkan pada kondisi tidak, nilai gen index setelah titikPotong sampai panjang kromosom individu kedua akan diisikan pada C1 dan nilai gen index setelah titikPotong sampai panjang kromosom individu pertama akan diisikan pada C2.

4.2.3 Mutasi

Proses reproduksi yang dilakukan setelah proses *crossover* yaitu proses mutasi. Mutasi yang digunakan dalam permasalahan ini adalah *random mutation.* Diagram alir proses mutasi dapat dilihat pada Gambar 4.4.



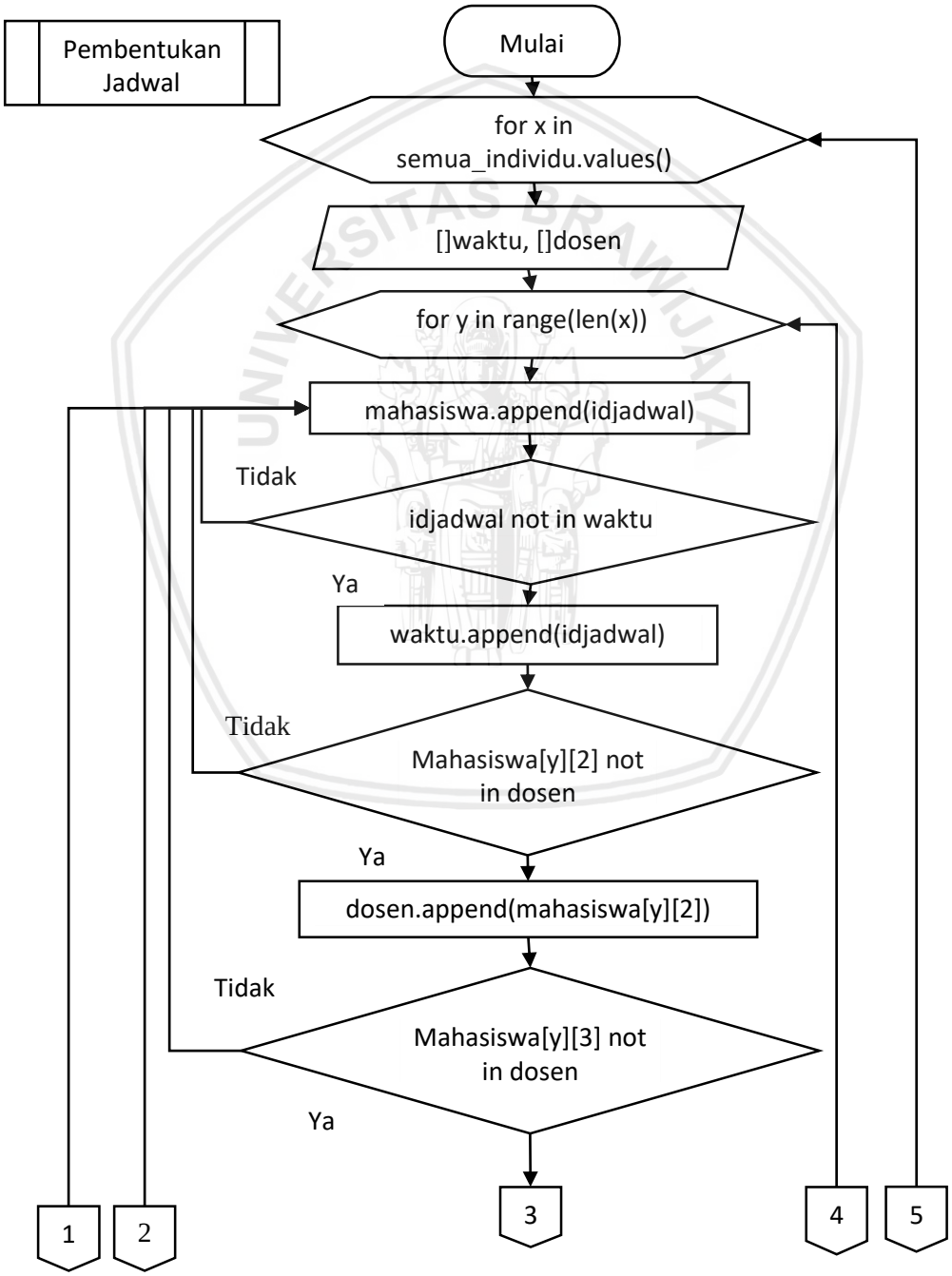
Gambar 4.4 Diagram alir proses mutasi

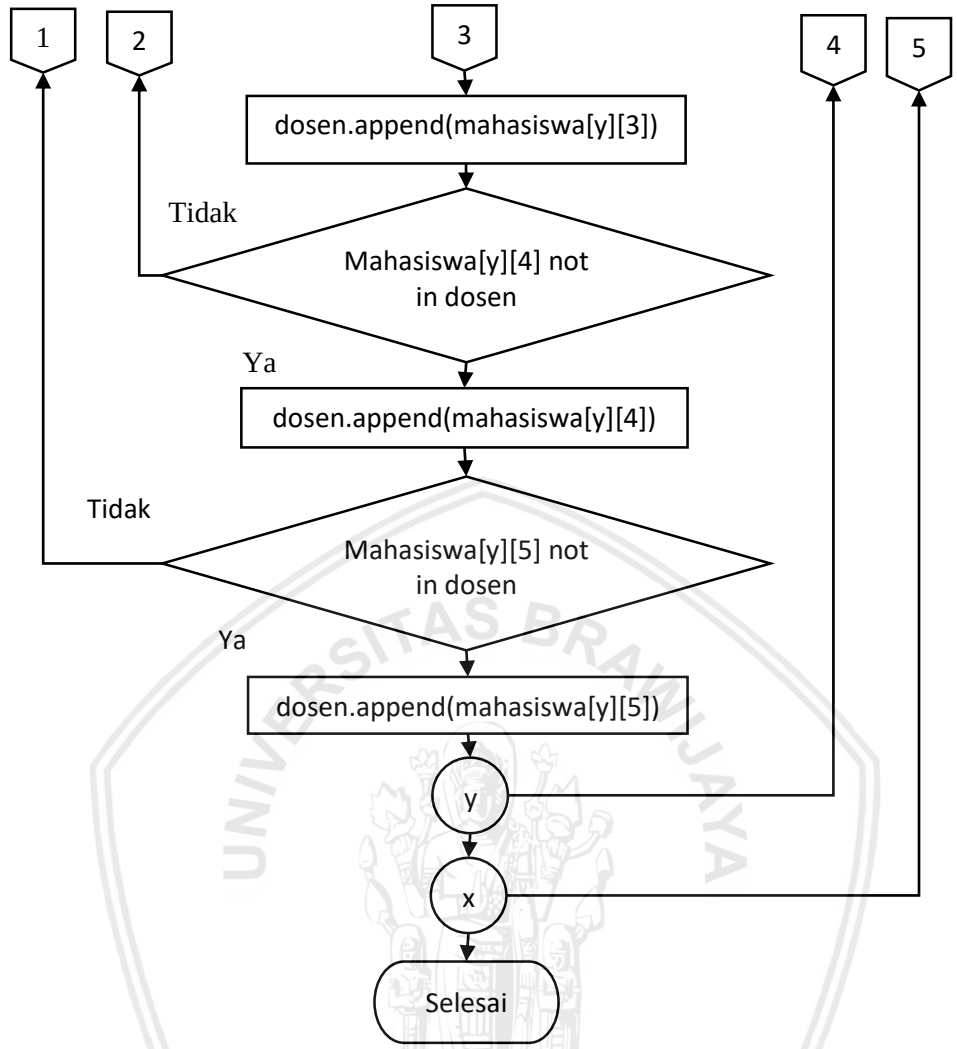
Berdasarkan Gambar 4.4 tahap pertama proses mutasi yaitu menentukan banyak *child* yang akan dihasilkan pada proses mutasi. Perulangan childKe dilakukan sebanyak anak yang akan dihasilkan, dimana pada langkah ini dilakukan pengisian childCross sejumlah anak hasil reproduksi proses mutasi yang telah ditentukan. Selanjutnya proses memilih satu induk secara acak, menentukan titik mutasi, dan nilai gen pengganti. Proses seleksi kondisi dilakukan dengan kondisi

jika nilai pengganti tidak sama dengan nilai gen pada induk dan nilai pengganti belum ada pada induk maka nilai gen induk pada titik mutasi akan diganti dengan nilai pengganti.

4.2.4 Pembentukan jadwal

Proses pembentukan jadwal dilakukan untuk menggabungkan antara data mahasiswa dengan data kode jadwal. Selain itu juga digunakan untuk mengambil nilai dari masing-masing kode dosen dan waktu sidang yang selanjutnya akan digunakan untuk proses pengecekan pelanggaran. Diagram alir proses pembentukan jadwal dapat dilihat pada Gambar 4.5.





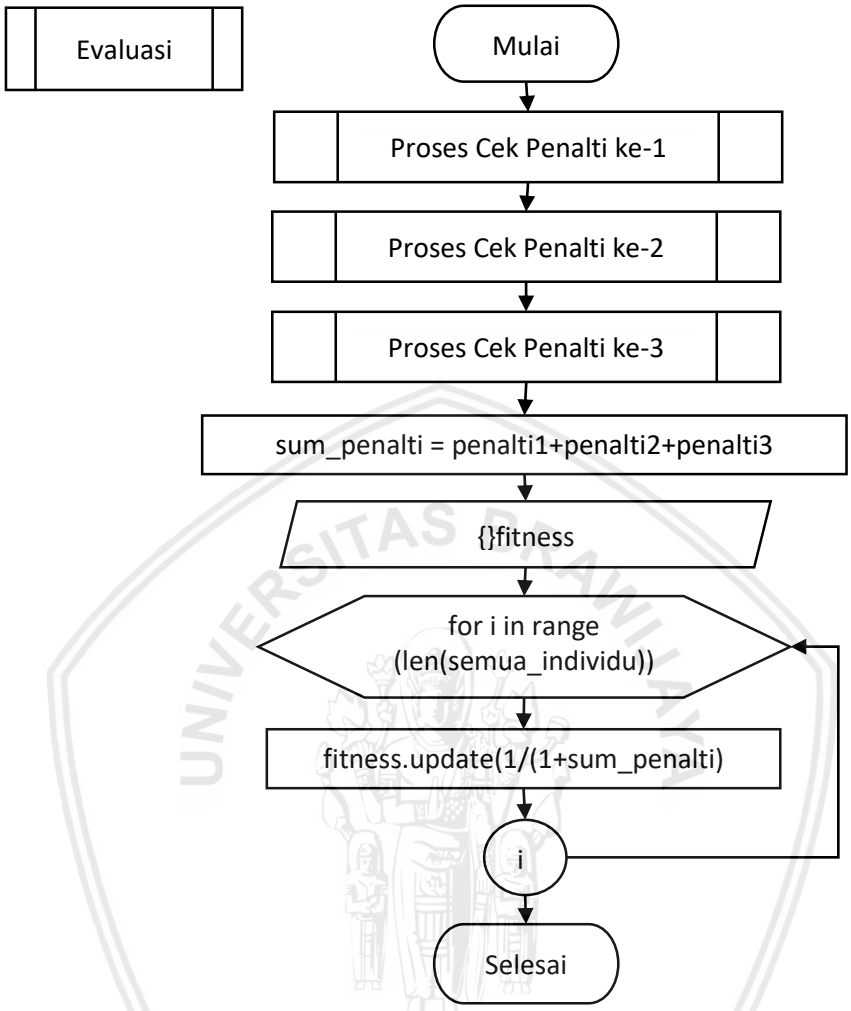
Gambar 4.5 Diagram alir proses pembentukan jadwal

Berdasarkan Gambar 4.5 proses pembentukan jadwal dilakukan sebanyak *value* dari semua individu. *Value* kromosom pada semua individu selanjutnya akan dikonversi menjadi kode jadwal dan digabungkan dengan data mahasiswa yang akan dijadwalkan. Perulangan *y* dilakukan sebanyak panjang value semua individu. Selanjutnya akan dilakukan proses seleksi kondisi untuk mengambil pembagian waktu sidang pada semua individu dan disimpan pada list waktu. Setelah didapatkan pembagian waktu, dilakukan proses seleksi kondisi untuk mengambil kode dosen pada masing-masing mahasiswa yaitu indeks 2 untuk dosen pembimbing 1, indeks 3 untuk dosen pembimbing 2, indeks 4 untuk dosen penguji 1, dan indeks 5 untuk dosen penguji 2 yang selanjutnya akan disimpan pada list dosen.

4.2.5 Evaluasi

Pada tahap ini, akan dilakukan proses perhitungan nilai *fitness* pada masing-masing individu. Sebelum menghitung nilai fitness, akan dilakukan pembentukan matriks jadwal sidang skripsi. Pembentukan matriks jadwal digunakan untuk melakukan pencarian bobot pelanggaran pada tiap individu

untuk menghitung nilai fitness dari suatu individu. Diagram alir proses pembentukan matriks jadwal dapat dilihat pada Gambar 4.6.

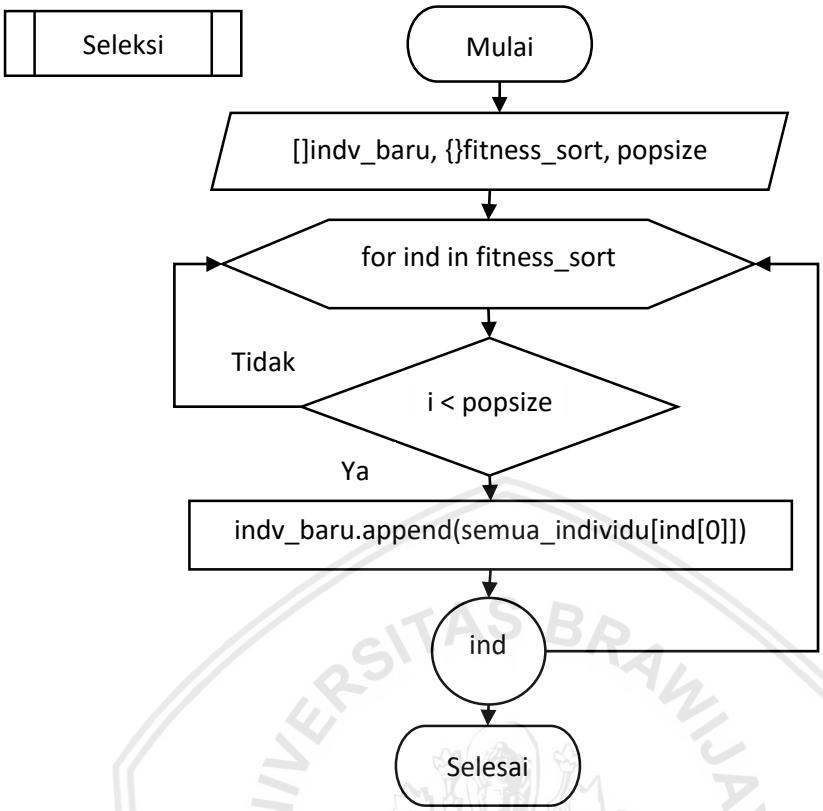


Gambar 4.6 Diagram alir proses evaluasi

Berdasarkan Gambar 4.6 langkah pertama yang dilakukan yaitu dengan mencari penalti dari pelanggaran *constraint* 1, *constraint* 2, *constraint* 3. Setelah diketahui semua penalti, kemudian dijumlahkan dan disimpan pada sum_penalti. Perulangan *i* dilakukan sebanyak panjang semua individu, dimana akan diisikan variabel *fitness* dengan formula nilai *fitness* yang telah ditentukan.

4.2.6 Seleksi

Permasalahan ini dilakukan proses seleksi dengan metode *elitism*. Semua individu yang berisi individu populasi awal dan anak hasil reproduksi ditempatkan dalam satu penampungan, kemudian diurutkan berdasarkan nilai *fitness* tertinggi, dan mengambil individu dengan *fitness* terbaik sebanyak *popsize*. Individu dengan nilai *fitness* tertinggi terpilih sebagai solusi. Diagram alir proses seleksi dapat dilihat pada Gambar 4.7.



Gambar 4.7 Diagram alir proses seleksi

Berdasarkan Gambar 4.7 langkah pertama yang dilakukan yaitu memanggil nilai *fitness* yang telah diurutkan dan ukuran popsize untuk menentukan berapa banyak individu yang lolos untuk generasi selanjutnya. Perulangan ind dilakukan sebanyak nilai *fitness* yang telah diurutkan, kemudian masuk proses seleksi kondisi jika *i* kurang dari *popsize,* maka individu yang lolos seleksi akan disimpan pada variabel indv_baru.

4.3 Perhitungan Manual Penempatan ruang sidang skripsi

Langkah pertama adalah mengubah data ruang dan waktu yang tersedia untuk sidang menjadi bentuk kode *integer*. Pada kasus ini, akan dilakukan kombinasi antara 16 ruang kelas dan 7 pembagian waktu yang telah dituliskan pada Tabel 4.1 dan Tabel 4.2, sehingga dihasilkan representasi 112 kode jadwal. Kode jadwal ini nantinya akan menentukan ruang dan waktu sidang untuk masing-masing mahasiswa. Data Kode jadwal dapat dilihat pada Tabel 4.3.

Tabel 4.3 Data kode jadwal

ID	Ruang	Waktu
1	B1.6	Sesi 1
2	B1.6	Sesi 2
3	B1.6	Sesi 3
4	B1.6	Sesi 4
5	B1.6	Sesi 5

6	B1.6	Sesi 6
7	B1.6	Sesi 7
⋮		
112	F4.8	Sesi 7

4.3.1 Inisialisasi parameter algoritme genetika

Pada tahap ini, akan ditentukan parameter-parameter pada Algoritme Genetika diantaranya nilai *crossover rate* (*cr*), nilai *mutation rate* (*mr*), *population size*, dan jumlah generasi. Dalam permasalahan ini akan ditentukan nilai parameternya sebagai berikut :

- Nilai *crossover rate* : 0,5
- Nilai *mutation rate* : 0,5
- *Popsize* : 3
- Generasi : 1

4.3.2 Representasi kromosom

Diberikan contoh permasalahan penempatan ruang sidang skripsi dengan jumlah mahasiswa yang akan dijadwalkan sebanyak 10. Mahasiswa yang akan dijadwalkan telah ditetapkan dua dosen penguji sidang skripsi sebelumnya. Dosen penguji dan pembimbing telah dikodekan sebelumnya menurut ID yang ditentukan. Pada proses penentuan jadwal, mahasiswa telah terpaket dengan dua dosen pembimbing dan dua dosen penguji. ID dosen dapat dilihat pada Tabel 4.4.

Tabel 4.4 ID dosen

ID	Nama Dosen
1	Agus Wahyu Widodo, S.T, M.Cs
2	Admaja Dwi Herlambang , S.Pd., M. Pd.
3	Agi Putra Kharisma, S.T, M.T
4	Alfi Nur Rusydi, S.Si., M.Sc.
5	Arief Andy Soebroto, S.T, M.Kom
⋮	
83	Djoko Pramono, S.T., M.Kom.

Selanjutnya akan dibangkitkan kromosom awal secara acak. Panjang kromosom ditentukan oleh jumlah mahasiswa. Kromosom yang digunakan menggunakan representasi permutasi untuk mendapatkan susunan kombinasi solusi yang tepat. Setiap gen dari setiap kromosom berisi integer angka yang menyatakan nomor ID jadwal yang telah dituliskan pada Tabel 4.3. Panjang kromosom pada penjadwalan ini yaitu 10, sejumlah mahasiswa yang akan dijadwalkan untuk sidang. Data mahasiswa dengan dosen pembimbing serta dosen penguji dan representasi kromosom dalam menyelesaikan permasalahan sidang skripsi ditunjukkan pada Tabel 4.5 dan Tabel 4.6.

Tabel 4.5 Data mahasiswa dengan dosen pembimbing dan dosen penguji

Mahasiswa	Pembimbing 1	Pembimbing 2	Penguji 1	Penguji 2
1	58	42	64	80
2	76	19	13	62
3	14	50	26	31
4	16	59	40	74
5	61	22	64	33
6	53	75	25	79
7	61	64	22	58
8	57	18	70	40
9	34	3	63	39
10	79	75	25	53

Tabel 4.6 Representasi kromosom

Individu	1	2	3	4	5	6	7	8	9	10
P1	112	2	62	80	104	106	31	63	50	76
P2	66	75	12	13	28	26	106	86	20	82
P3	44	49	105	87	81	8	95	2	57	112

4.3.3 Inisialisasi populasi awal

Panjang kromosom pada permasalahan ini dicontohkan sebesar 10, sejumlah mahasiswa yang akan dijadwalkan untuk sidang dengan individu-individu populasi awal yaitu P1, P2, dan P3 yang telah ditetapkan berdasarkan *population size*-nya yaitu 3. Inisialisasi populasi awal yang digunakan dapat dilihat pada Tabel 4.7.

Tabel 4.7 Inisialisasi populasi awal

Individu	Kromosom									
P1	112	2	62	80	104	106	31	63	50	76
P2	66	75	12	13	28	26	106	86	20	82
P3	44	49	105	87	81	8	95	2	57	112

4.3.4 *Crossover*

Metode *crossover* yang digunakan pada permasalahan ini adalah *one cut point*. Dalam proses *crossover* akan menghasilkan sejumlah keturunan *(offspring)*. Jumlah *offspring* yang dihasilkan didapatkan dengan mengalikan nilai *population size* dengan *crossover rate* yang telah ditentukan sebelumnya. Nilai *offspring* pada proses *crossover* ini dapat diformulasikan sebagai berikut:

$$offspring = 0.5 \times 3 = 1.5 \text{ (dibulatan ke atas menjadi 2)}$$

Berdasarkan formula diatas, dihasilkan *offspring* sejumlah 2 yang akan menjadi individu baru. Langkah selanjutnya, memilih dua individu awal dalam populasi secara acak untuk menjadi induk *(parent)* yang akan menghasilkan *offspring* atau

biasa disebut anak *(child).* Dimisalkan 2 induk yang terpilih yaitu P3 dan P1. Induk yang telah terpilih kemudian ditentukan titik potong *crossover* yang merupakan titik pertukaran gen dari kedua induk untuk menghasilkan anak. Dimisalkan titik potong yang terpilih yaitu 2, maka akan dihasilkan 2 anak yaitu C1 dan C2 yang merupakan kombinasi gen dari kedua induk dengan titik pertukaran pada gen ke-2 untuk masing-masing induk. Proses *crossover* pada permasalahan ini dapat dilihat pada Tabel 4.8.

Tabel 4.8 Proses *crossover*

Individu	Kromosom									
P2	66	75	12	13	28	26	106	86	20	82
P1	112	2	62	80	104	106	31	63	50	76
C1	66	75	12	13	28	26	31	63	50	76
C2	112	2	62	80	104	106	106	86	20	82

4.3.5 Mutasi

Metode mutasi yang digunakan yaitu *random mutation.* Langkah awal yang dilakukan yaitu menentukan jumlah *offspring* yang dihasilkan dengan mengalikan nilai *population size* dengan *mutation rate* yang telah ditentukan sebelumnya. Nilai *offspring* pada proses mutasi ini dapat diformulasikan sebagai berikut :

$$offspring = 0.5 \times 3 = 1.5 \text{ (dibulatan ke atas menjadi 2)}$$

Berdasarkan formula diatas, dihasilkan *offspring* sejumlah 2 yang akan menjadi individu baru dengan dua kali proses mutasi. Pada tiap proses mutasi hanya menggunakan satu induk. Langkah selanjutnya, memilih satu induk secara acak yang akan menghasilkan anak. Dimisalkan induk yang terpilih pada mutasi pertama yaitu P3 dan induk yang terpilih pada mutasi kedua yaitu P3. Kemudian menentukan gen keberapa yang akan dimutasi. Misalkan P3 mutasi pertama yang dimutasi adalah gen ke-2 dan P2 mutasi kedua adalah gen ke-9. Nilai gen tersebut selanjutnya akan diganti dengan nilai gen lain secara acak. Pada mutasi P3 pertama akan menghasilkan anak ke-3 yaitu C3 dan mutasi P3 kedua akan menghasilkan anak ke-4 yaitu C4. Proses mutasi pada permasalahan ini dapat dilihat pada Tabel 4.9.

Tabel 4.9 Proses mutasi

Individu	Kromosom									
P3	44	49	105	87	81	8	95	2	57	112
P3	44	49	105	87	81	8	95	2	57	112
C3	44	49	105	87	81	8	95	2	61	112
C4	44	49	105	87	81	8	95	2	25	112

4.3.6 Evaluasi

Setelah proses *crossover* dan mutasi maka akan didapatkan individu gabungan yaitu gabungan populasi awal dan populasi *offspring* yang dihasilkan

melalui proses *crossover* dan mutasi. Individu gabungan dapat dilihat pada Tabel 4.10.

Tabel 4.10 Individu gabungan

Individu	Kromosom									
P1	112	2	62	80	104	106	31	63	50	76
P2	66	75	12	13	28	26	106	86	20	82
P3	44	49	105	87	81	8	95	2	57	112
C1	66	75	12	13	28	26	31	63	50	76
C2	112	2	62	80	104	106	106	86	20	82
C3	44	49	105	87	81	8	95	2	61	112
C4	44	49	105	87	81	8	95	2	25	112

Langkah selanjutnya yaitu dilakukan konversi nilai gen tiap kromosom yang berisi daftar ruang sidang, jam mulai, dan jam selesai ke masing-masing mahasiswa. Konversi kromosom P1 ditunjukkan pada Tabel 4.11.

Tabel 4.11 Konversi kromosom P1 menjadi jadwal sidang skripsi

Nilai Gen	Mahasiswa	Pembimbing 1	Pembimbing 2	Penguji 1	Penguji 2	Ruang	Waktu
112	1	56	40	62	78	E2.3	Sesi 4
2	2	74	17	12	60	E2.7	Sesi 6
62	3	13	48	24	29	E2.5	Sesi 2
80	4	14	57	38	72	F3.3	Sesi 7
104	5	59	20	62	31	E2.3	Sesi 5
106	6	51	73	23	77	E2.7	Sesi 2
31	7	59	62	20	56	F4.6	Sesi 7
63	8	55	16	68	38	B1.6	Sesi 1
50	9	32	3	61	37	F3.5	Sesi 7
76	10	77	73	23	51	F4.7	Sesi 6

Sebelum mencari total bobot penalti dari setiap kromosom. Bobot penalti didapatkan berdasarkan batasan *(constraint)* yang telah ditetapkan. Pada permasalahan ini, hanya digunakan *hard constraint*. Bobot penalti pada permasalahan ini ditunjukkan pada Tabel 4.12.

Tabel 4.12 Batasan penempatan ruang sidang skripsi

No.	Batasan *(Constraint)*	Bobot Penalti
1.	Tidak boleh terdapat jadwal yang sama dalam satu hari	1
2.	Dua dosen penguji atau dua dosen pembimbing tidak boleh bentrok pada waktu sidang yang sama	1
3.	Dua dosen penguji atau dua dosen pembimbing dengan jadwal beruntun tidak berada pada ruang sidang yang sama	1

Pada perhitungan bobot penalti ke-1, tidak boleh ada jadwal yang sama dalam satu hari itu, karena secara otomatis akan bentrok ruang dan waktu sidang. Hal tersebut dapat dilihat dari gen-gen yang mengisi dalam satu kromosom tersebut. Jika terdapat gen yang nilainya sama dalam satu kromosom, maka terjadi pelanggaran pada *constraint* 1. Tiap satu kali pelanggaran *constraint* 1, dikenakan satu kali penalti dan berlaku kelipatannya. Total nilai penalti semua individu pada *constraint* ke-1 ditunjukkan pada Tabel 4.13.

Tabel 4.13 Total nilai penalti pada *constraint* ke-1

Individu	Total Nilai Penalti
P1	0
P2	0
P3	0
C1	0
C2	1
C3	0
C4	0

Selanjutnya pada perhitungan bobot penalti ke-2, dosen-dosen akan dipetakan pada pembagian waktu sidang yang telah dijadwalkan. Dosen yang dipetakan lebih dari satu kali pada waktu yang sama, maka akan melanggar *constraint* kedua. Tiap satu kali pelanggaran *constraint*, dikenakan satu kali penalti dan berlaku kelipatannya. Pemetaan dosen pada kromosom P1 ke masing-masing waktu sidang dan total nilai penalti semua individu pada *constraint* ke-2 ditunjukkan pada Tabel 4.14 dan Tabel 4.15.

Tabel 4.14 Pemetaan dosen kromosom P1 ke masing-masing waktu sidang

Waktu	Dosen											
Sesi 1	53	75	25	79	34	3	63	39				
Sesi 2	76	19	13	62								
Sesi 3	16	59	40	74	61	64	22	58				
Sesi 4												
Sesi 5												
Sesi 6	14	50	26	31	61	22	64	33	79	75	25	53
Sesi 7	58	42	64	80	57	18	70	40				

Tabel 4.15 Total nilai penalti pada *constraint* ke-2

Individu	Total Nilai Penalti
P1	0
P2	0
P3	3
C1	2
C2	0
C3	3
C4	3

Selanjutnya yaitu proses perhitungan bobot penalti ke-3. Pada perhitungan bobot penalti ke-3, dosen-dosen akan dipecah satu persatu berdasarkan ID nya. Kemudian, pembagian waktu pada sidang akan dikelompokkan menjadi 5 waktu terusan yaitu sesi 1 ke sesi 2 adalah terusan 1, sesi 2 ke sesi 3 adalah terusan 2, sesi 4 ke sesi 5 adalah terusan 3, sesi 5 ke sesi 6 adalah terusan 4, sesi 6 ke sesi 7 adalah terusan 5. Dosen yang mendapatkan jadwal sidang terusan tetapi pada ruangan yang berbeda, maka akan melanggar *constraint* ketiga. Tiap satu kali pelanggaran *constraint*, dikenakan satu kali penalti dan berlaku kelipatannya. Pemetaan dosen pada kromosom P1 ke masing-masing pembagian waktu terusan dan total nilai semua individu penalti pada *constraint* ke-3 ditunjukkan pada Tabel 4.16 dan Tabel 4.17.

Tabel 4.16 Pemetaan dosen kromosom P1 ke pembagian waktu terusan

Dosen	Sesi 1 ke Sesi 2	Sesi 2 ke Sesi 3	Sesi 4 ke Sesi 5	Sesi 5 ke Sesi 6	Sesi 6 ke Sesi 7
58	-	-	-	-	-
42	-	-	-	-	-
⋮	-	-	-	-	-
64	-	-	-	-	F4.7 sesi 6 ke F4.8 sesi 7
⋮	-	-	-	-	-
53	-	-	-	-	-

Tabel 4.17 Total nilai penalti pada *constraint* ke-3

Individu	Total Nilai Penalti
P1	1
P2	0
P3	1
C1	5
C2	2
C3	1
C4	1

Selanjutnya akan dilakukan proses perhitungan nilai kebugaran *(fitness)* tiap individu. Semakin besar nilai *fitness,* maka akan semakin baik individu tersebut untuk dijadikan sebagai calon solusi. Nilai *fitness* pada permasalahan ini diperoleh dengan formula yang telah dituliskan pada Persamaan (2.3). Nilai *fitness* yang dihasilkan untuk masing-masing individu ditunjukkan pada Tabel 4.18.

Tabel 4.18 Perhitungan nilai *fitness*

Individu	Kromosom										Penalti	*Fitness*
P1	112	2	62	80	104	106	31	63	50	76	1	0,500
P2	66	75	12	13	28	26	106	86	20	82	0	1,000
P3	44	49	105	87	81	8	95	2	57	112	4	0,200

C1	66	75	12	13	28	26	31	63	50	76	7	0,125
C2	112	2	62	80	104	106	106	86	20	82	3	0,250
C3	44	49	105	87	81	8	95	2	61	112	4	0,200
C4	44	49	105	87	81	8	95	2	25	112	4	0,200

4.3.7 Seleksi

Setelah diperoleh nilai *fitness* pada tiap individu, selanjutnya yaitu melakukan proses seleksi. Pada permasalahan ini, proses seleksi dilakukan dengan metode *elitsm* yaitu dengan memilih individu berdasarkan nilai *fitness* tertinggi untuk tetap bertahan pada generasi selanjutnya. Harapannya agar individu terbaik akan menghasilkan calon solusi yang baik juga. Sejumlah n individu dengan nilai *fitness* terkecil akan dieliminasi. Berdasarkan Tabel 4.14, individu yang lolos untuk generasi selanjutnya yaitu P2, P3, C3. Individu yang lolos untuk generasi selanjutnya dapat dilihat pada Tabel 4.19.

Tabel 4.19 Hasil seleksi *elitsm*

Individu	*Fitness*
P2	1,000
P1	0,500
C2	0,250
P3	0,200
C3	0,200
C4	0,200
C1	0,125

Ket:

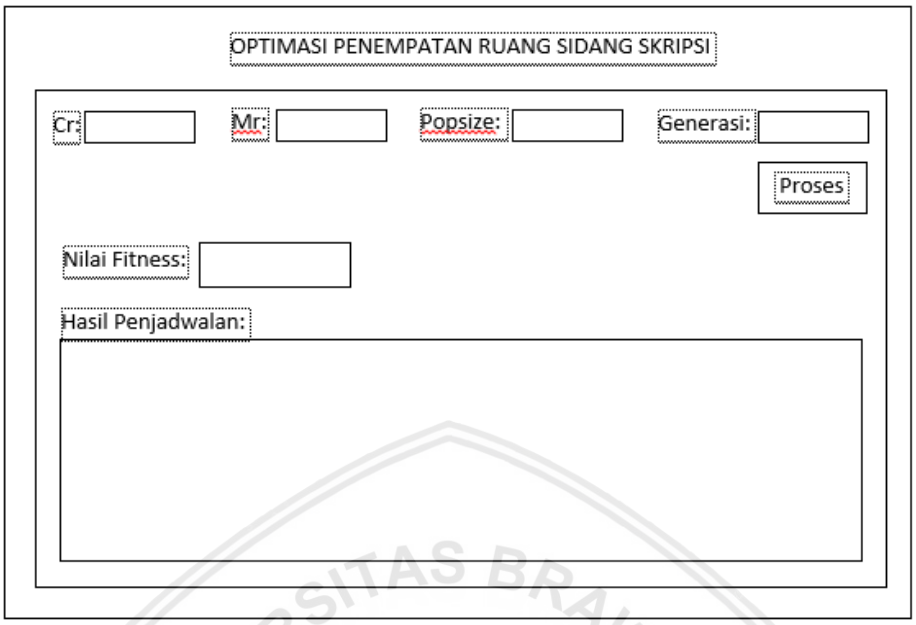
= Individu yang lolos seleksi

4.4 Perancangan Antarmuka

Perangcangan antarmuka digunakan untuk menjelaskan antarmuka sistem agar memudahkan pengguna dalam berinteraksi dengan sistem. Perancangan antarmuka pada sistem ini terdiri dari halaman beranda saja.

4.4.1 Perancangan Antarmuka Beranda

Pada antarmuka beranda terdiri dari kolom *input* parameter algoritme genetika, tombol proses, *listbox* untuk menampilkan nilai *fitness*, dan *scrolledlistbox* untuk menampilkan hasil penjadwalan sistem. Perancangan antarmuka beranda ditunjukkan Gambar 4.8.



Gambar 4.8 Perancangan Antarmuka Beranda

BAB 5 IMPLEMENTASI

Bab ini menjelaskan terkait implementasi sistem yang dibuat berdasarkan perancangan sistem sebelumnya yang akan diimplementasikan ke dalam bentuk yang lebih aplikatif untuk proses penempatan ruang sidang skripsi.

5.1 Implementasi Penggunaan Perangkat Lunak

Dalam melakukan implementasi penempatan ruang sidang skripsi menggunakan algoritme genetika, spesifikasi perangkat lunak yang digunakan ditunjukkan pada Tabel 5.1.

Tabel 5.1 Spesifikasi perangkat lunak

Komponen	Spesifikasi
Sistem operasi	Windows 10
Bahasa pemrograman	Python 3.6
Tools pemrograman	Spyder IDE

5.2 Implementasi Penggunaan Perangkat Keras

Dalam melakukan implementasi penempatan ruang sidang skripsi menggunakan algoritme genetika, spesifikasi perangkat keras yang digunakan ditunjukkan pada Tabel 5.2.

Tabel 5.2 Spesifikasi perangkat keras

Komponen	Spesifikasi
Prosesor	AMD Dual Core A9-9420 up to 3.6 GHz
Memori	4.0 GB
Hardisk	500 GB

5.3 Implementasi Program

Pada Implementasi program akan dijelaskan mengenai proses algoritme genetika untuk proses penempatan ruang sidang skripsi dalam bentuk kode program.

5.3.1 Implementasi proses inisialisasi populasi awal

Implementasi proses inisialisasi ini berdasarkan dari perancangan diagram alir Gambar 4.2, dengan proses awal yaitu akan dibangkitkan populasi secara acak sebanyak nilai *popsize*. Panjang kromosom berdasarkan jumlah mahasiswa yang akan dijadwalkan untuk 1 hari penempatan ruang sidang skripsi. Proses inisialisasi populasi awal ditunjukkan pada Gambar 5.1.

No	Kode Program
1	`def inisialisasi_parent(self):`
2	`    parent = []`
3	`    for x in range(self.popsize):`
4	`        Kromosom = []`
5	`        j = 0`
6	`        while j < self.pjgkrom:`
7	`            jdwl = np.random.randint(1, len(self.ID)+1)`
8	`            if jdwl not in kromosom:`
9	`                kromosom.append(jdwl)`
10	`                j+=1`
11	`        parent.append(kromosom)`
12	`   return parent`

Gambar 5.1 Proses Inisialisasi Populasi Awal

Berikut penjelasan Gambar 5.1:

1. Baris 1, inisialisasi nama method dengan parameter *self*.
2. Baris 2, inisialiasi *list* parent untuk menampung semua nilai dari list kromosom.
3. Baris 3, proses perulangan sampai sebanyak popsize.
4. Baris 4, inisialiasi *list* kromosom untuk menampung nilai gen.
5. Baris 5, inisialisasi variabel j sama dengan 0 untuk proses perulangan.
6. Baris 6, proses perulangan dari nilai j kurang dari panjang kromosom.
7. Baris 7, inisialisasi variabel jadwal yang berisi nilai random, dengan nilai dari index 1 sampai panjang index ID yaitu id jadwal yang ada ditambah 1.
8. Baris 8, proses seleksi kondisi jika nilai random pada variabel jdwl tidak ada pada kromosom.
9. Baris 9, proses menambahkan nilai acak jadwal pada *list* kromosom.
10. Baris 10, proses penambahan 1 nilai perulangan.
11. Baris 11, proses menambahkan nilai dari *list* kromosom ke dalam *list* parent .
12. Baris 12, proses pengembalian nilai parent.

5.3.2 Implementasi proses *crossover*

Implementasi *crossover* berdasarkan dari perancangan diagram alir Gambar 4.3. Pada proses ini akan dibangkitkan 2 individu secara acak untuk dijadikan sebagai induk yang akan menghasilkan keturunan . Dalam sekali proses *crossover* akan dihasilkan 2 individu baru. Proses *crossover* ditunjukkan pada Gambar 5.2.

No	Kode Program
1	`def crossover(self):`
2	`    childKe = 0`

```
3       childCross = []
4       while childKe < self.byk_childCross:
5           C1 = []
6           C2 = []
7           P1 = self.parent[np.random.randint(0,len(self.parent))]
8           P2 = self.parent[np.random.randint(0,len(self.parent))]
9           self.titikPotong = self.titik_potong()
10          if P1 != P2:
11              for a in range(0,self.pjgkrom):
12                  if a < self.titikPotong:
13                      C1.append(P1[a])
14                      C2.append(P2[a])
15                  else:
16                      C1.append(P2[a])
17                      C2.append(P1[a])
18              if childKe < self.byk_childCross:
19                  childCross.append(C1)
20                  childKe += 1
21              if childKe < self.byk_childCross:
22                  childCross.append(C2)
23                  childKe += 1
24      return childCross
```

Gambar 5.2 Proses *crossover*

Berikut penjelasan Gambar 5.2:

1. Baris 1, inisialisasi nama method dengan parameter *self*.
2. Baris 2, inisialisasi variabel childKe sama dengan 0 untuk proses perulangan.
3. Baris 3, inisialiasi *list* childCross untuk menampung nilai anak hasil *crossover.*
4. Baris 4, proses perulangan dari nilai childKe kurang dari banyak anak hasil *crossover* yang akan dihasilkan dengan memanggil fungsi byk_childCross.
5. Baris 5-6, inisialiasi *list* C1 dan C2 untuk menampung nilai anak hasil *crossover* pertama dan kedua.
6. Baris 7-8, inisialisasi variabel P1 dan P2 untuk parent proses *crossover* yang berisi individu random dari semua parent awal, dengan nilai dari index 0 sampai panjang indek parent
7. Baris 9, memanggil fungsi titik_potong untuk penentuan titik potong *crossover*
8. Baris 10, proses seleksi jika P1 tidak sama dengan P2
9. Baris 11-17, proses perulangan dari indeks 0 sampai panjang kromosom untuk menampung gen-gen hasil perpotongan antara P1 dan P2.

10. Baris 18-22, proses seleksi kondisi jika nilai childKe kurang dari banyak *child* hasil kromosom, maka menampung kromosom hasil perpotongan pada *list* childCross.

11. Baris 3-6, proses pengembalian nilai childCross.

5.3.3 Implementasi proses mutasi

Implementasi mutasi berdasarkan pada perancangan diagram alir Gambar 4.4. Pada proses ini akan dibangkitkan 1 individu secara acak untuk dijadikan sebagai induk yang akan menghasilkan keturunan. Dalam sekali proses mutasi akan dihasilkan 1 individu baru. Proses mutasi ditunjukkan pada Gambar 5.3.

No	Kode Program
1	`def mutasi(self):`
2	`    childMutasi = []`
3	`    childKe = 0`
4	`    while childKe < self.byk_childMutasi:`
5	`        titik_mutasi = np.random.randint(0,self.pjgkrom-1)`
6	`        P = self.parent[np.random.randint(0,len(self.parent))][:]`
7	`        pengganti = np.random.randint(1, len(self.ID)+1)`
8	`        if (pengganti != P[titik_mutasi]) and (pengganti not in P)`
9	`            P[titik_mutasi] = pengganti`
10	`            childMutasi.append(P)`
11	`            childKe+=1`
12	`    return childMutasi`

Gambar 5.3 Proses mutasi

Berikut penjelasan Gambar 5.3:

1. Baris 1, inisialisasi nama method dengan parameter *self*.
2. Baris 2, inisialiasi *list* childMutasi untuk menampung nilai anak hasil mutasi.
3. Baris 3, inisialisasi variabel childKe sama dengan 0 untuk proses perulangan.
4. Baris 4, proses perulangan dari nilai childKe kurang dari banyak anak hasil mutasi yang akan dihasilkan dengan memanggil fungsi byk_childMutasi.
5. Baris 5, inisialisasi variabel titik_mutasi untuk menentukan titik mutasi yaitu dengan titik random dari index 0 sampai indeks panjang kromosom kurang 1.
6. Baris 6, inisialisasi variabel P untuk parent proses mutasi yang berisi individu random dari semua parent awal, dengan nilai dari index 0 sampai panjang indek parent dan nilai dari parent awal tidak berubah jika sudah di proses mutasi.
7. Baris 7, inisialisasi variabel pengganti yang berisi nilai random, dengan nilai dari index 1 sampai panjang index ID yaitu id jadwal yang ada ditambah 1.

8. Baris 8-11, proses seleksi kondisi jika nilai pengganti tidak sama dengan nilai titik mutasi pada parent dan nilai gen belum ada pada kromosom, maka nilai pada titik mutasi akan diganti dengan nilai pengganti dan dimasukkan pada list childMutasi.
9. Baris 12, proses pengembalian nilai childMutasi.

5.3.4 Implementasi proses pembentukan jadwal

Implementasi proses perhitungan *fitness* berdasarkan pada perancangan diagram alir Gambar 4.6. Nilai *fitness* digunakan untuk mengetahui kualitas dari masing-masing individu untuk dijadikan sebagai solusi. Nilai ini diperoleh dengan formula pembagian antara nilai maksimal *fitness* yang didefinisikan dibagi dengan bobot total penalti ditambah 1. Proses perhitungan *fitness* ditunjukkan pada Gambar 5.4.

No	Kode Program
1	`def jadwal(self):`
2	`    for x in self.semua_individu.values():`
3	`        self.waktu = []`
4	`        self.dosen = []`
5	`        for y in range(len(x)):`
6	`            self.mahasiswa[y].append(self.idjadwal[str(x[y])])`
7	`            if self.idjadwal[str(x[y])][1] not in self.waktu:`
8	`                self.waktu.append(self.idjadwal[str(x[y])][1])`
9	`            if self.mahasiswa[y][2] not in self.dosen:`
10	`                self.dosen.append(self.mahasiswa[y][2])`
11	`            if self.mahasiswa[y][3] not in self.dosen:`
12	`                self.dosen.append(self.mahasiswa[y][3])`
13	`            if self.mahasiswa[y][4] not in self.dosen:`
14	`                self.dosen.append(self.mahasiswa[y][4])`
15	`            if self.mahasiswa[y][5] not in self.dosen:`
16	`                self.dosen.append(self.mahasiswa[y][5]`

Gambar 5.4 Proses pembentukan jadwal

Berikut penjelasan Gambar 5.4:

1. Baris 1, inisialisasi nama method dengan parameter *self*.
2. Baris 2, proses perulangan sampai sebanyak *value* semua individu.
3. Baris 3-4, inisialiasi *list* waktu dan *list* dosen untuk menampung pembagian slot waktu dan dosen dari data jadwal yang terbentuk.
4. Baris 5, proses perulangan sebanyak panjang *x*.
5. Baris 6, proses mengisi data mahasiswa dengan kode jadwal yang untuk masing-masing individu.

6. Baris 7-8, proses seleksi kondisi jika nilai idjadwal index ke 1 yaitu waktu belum ada pada *list* waktu, maka nilai waktu pada idjadwal akan diisikan pada *list* waktu.
7. Baris 9-10, proses seleksi kondisi jika nilai mahasiswa index ke 2 yaitu dosen pembimbing 1 belum ada pada *list* dosen, maka akan diisikan pada *list* dosen.
8. Baris 11-12, proses seleksi kondisi jika nilai mahasiswa index ke 3 yaitu dosen pembimbing 2 belum ada pada *list* dosen, maka akan diisikan pada *list* dosen.
9. Baris 13-14, proses seleksi kondisi jika nilai mahasiswa index ke 4 yaitu dosen penguji 1 belum ada pada *list* dosen, maka akan diisikan pada *list* dosen.
10. Baris 15-16, proses seleksi kondisi jika nilai mahasiswa index ke 5 yaitu dosen penguji 2 belum ada pada *list* dosen, maka akan diisikan pada *list* dosen.

5.3.5 implementasi proses perhitungan *fitness*

Implementasi proses perhitungan *fitness* berdasarkan pada perancangan diagram alir Gambar 4.6. Nilai *fitness* digunakan untuk mengetahui kualitas dari masing-masing individu untuk dijadikan sebagai solusi. Nilai ini diperoleh dengan formula pembagian antara nilai maksimal *fitness* yang didefinisikan dibagi dengan bobot total penalti ditambah 1. Proses perhitungan *fitness* ditunjukkan pada Gambar 5.5.

No	Kode Program
1	`def hitung_fitness(self):`
2	`    penalti1 = self.constraint1()`
3	`    penalti2 = self.constraint2()`
4	`    penalti3 = self.constraint3()`
5	`    sum_penalti = np.add(np.add(penalti1,penalti2),penalti3)`
6	`    fitness = {}`
7	`    for i in range(len(self.semua_individu)):`
8	`        fitness.update({"Individu_" +str(i+1):float(1)/ (1+sum_penalti[i])})`
9	`    return fitness`

Gambar 5.5 Proses perhitungan *fitness*

Berikut penjelasan Gambar 5.5:

1. Baris 1, inisialisasi nama method dengan parameter *self*.
2. Baris 2-4, inisialiasi variabel penalti1, penalti2, penalti3 dengan memanggil fungsi constraint1(), constraint2() dan constraint3() .
3. Baris 4, proses perhitungan total nilai penalti1, penalti2, dan penalti3 dan disimpan pada variabel sum_penalti.
4. Baris 5, inisialisasi *dictionary* fitness.
5. Baris 6-7, proses perulangan sebanyak panjang indeks semua individu untuk menghitung nilai *fitness* dari total nilai penalti.

6. Baris 12, proses pengembalian nilai *fitness*.

5.3.6 Implementasi proses seleksi

Implementasi proses perhitungan seleksi berdasarkan pada perancangan diagram alir Gambar 4.7, yang bertujuan untuk menentukan individu-individu mana saja yang dipertahankan untuk proses generasi selanjutnya. Proses seleksi ini dilakukan dengan mengurutkan individu-individu secara *descending* berdasarkan nilai *fitness* tertinggi sampai terendah, lalu diambil individu dengan nilai *fitness* tertinggi sejumlah *popsize*. Implementasi proses seleksi ditunjukkan pada Gambar 5.6.

No	Kode Program
1	`def seleksi(self):`
2	`    i = 0`
3	`    indv_baru = []`
4	`    for ind in self.fitness_sort:`
5	`        if  i < self.popsize:`
6	`            indv_baru.append(self.semua_individu[ind[0]])`
7	`            i += 1`
8	`    return indv_baru`

Gambar 5.6 Proses seleksi

Berikut penjelasan Gambar 5.6 :

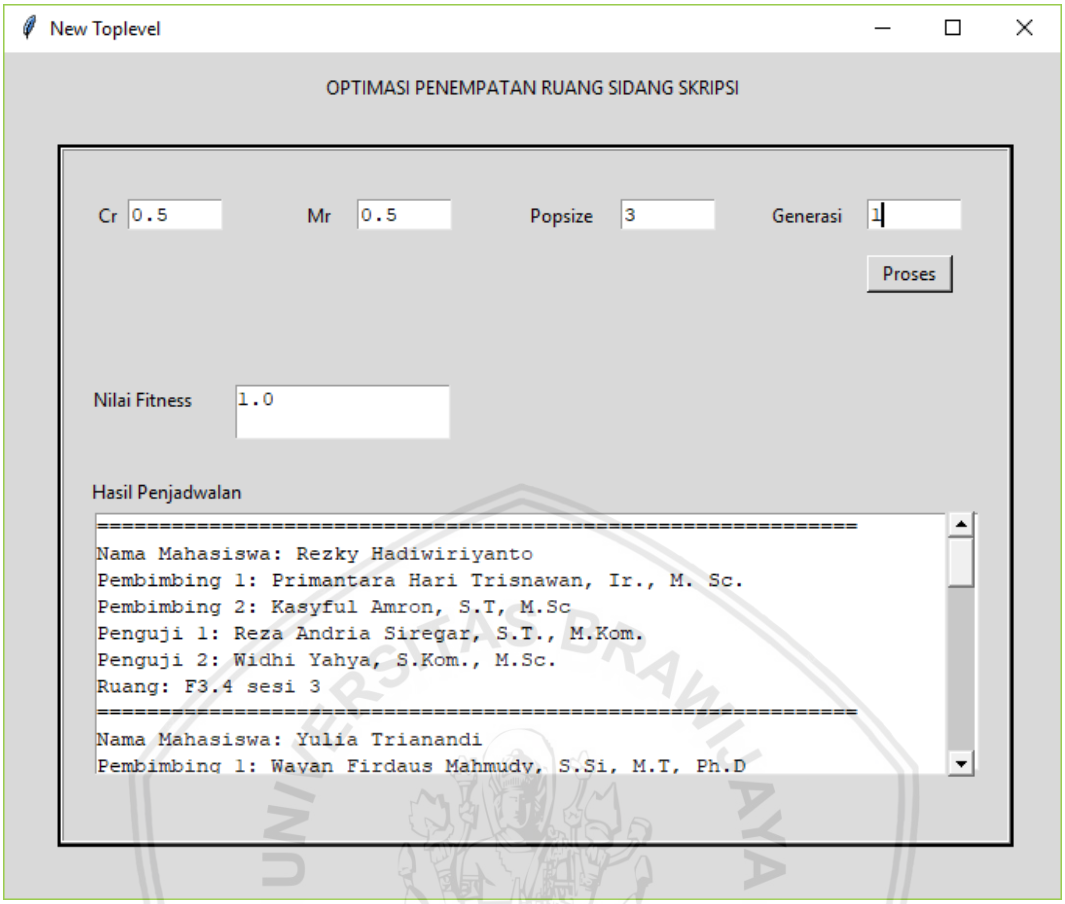
1. Baris 1, inisialisasi nama method dengan parameter *self*.
2. Baris 2, inisialisasi variabel i sama dengan 0 untuk seleksi kondisi.
3. Baris 3, inisialiasi *list* indv_baru untuk menampung nilai semua individu yang lolos seleksi.
4. Baris 4-7, proses perulangan sebanyak nilai dari pemanggilan fungsi fitness_sort dengan seleksi kondisi jika nilai variabel i kurang dari *popsize*, maka individu yang lolos seleksi dimasukkan pada *list* indv_baru.
5. Baris 8, proses pengembalian nilai indv_baru.

5.4 Impelementasi Antarmuka

Pada implementasi antarmuka akan ditunjukkan antarmuka sistem agar memudahkan pengguna dalam berinteraksi dengan sistem. Impelentasi antarmuka pada sistem ini terdiri dari halaman beranda saja.

5.4.1 Implementasi Antarmuka Beranda

Antarmuka beranda ini terdiri dari kolom *input* parameter algoritme genetika, tombol proses, *listbox* untuk menampilkan nilai *fitness*, dan *scrolledlistbox* untuk menampilkan hasil penjadwalan sistem. Implementasi antarmuka beranda ditunjukkan Gambar 5.7.



Gambar 5.7 Implementasi Antarmuka Beranda

BAB 6 PENGUJIAN DAN PEMBAHASAN

Pada bab ini akan dijelaskan tentang hasil pengujian yang telah dilakukan beserta analisis performa dari algoritme genetika pada sistem penempatan ruang sidang skripsi. Jumlah mahasiswa yang akan dijadwalkan pada proses pengujian ini sebanyak 25 mahasiswa. Pengujian yang dilakukan pada bab ini diantaranya pengujian kombinasi nilai *crossover rate* dan *mutation rate*, pengujian ukuran populasi, dan pengujian ukuran generasi.

6.1 Pengujian Kombinasi Crossover Rate dan Mutation Rate

Pada sub bab pengujian kombinasi nilai *crossover rate* (*cr*) dan *mutation rate* (*mr*) ini dilakukan dengan uji coba kombinasi nilai *cr* dan *mr* antara 0,1 sampai 0,9. Proses pengujian kombinasi *crossover rate (cr)* dan *mutation rate (mr)* dilakukan untuk mendapatkan kombinasi *cr* dan *mr* yang dapat menghasilkan rata-rata nilai *fitness* yang optimal dan mengetahui rata-rata waktu komputasi yang dihasilkan. Pengujian kombinasi *crossover rate* (*cr*) dan *mutation rate* (*mr*) mengacu pada Tabel 4.18 dan Tabel 4.19 pada sub bab perancangan pengujian kombinasi nilai *crossover rate* (*cr*) dan *mutation rate (mr)*. Proses yang dilakukan pertama kali adalah mencari nilai konvergen dari proses dengan ukuran populasi dan generasi sebesar-besarnya. Pada pengujian pertama kali akan digunakan ukuran populasi sebesar 100 dengan jumlah generasi 100, dan didapatkan hasil yang konvergen saat generasi ke 100. Setelah didapatkan hasil yang sama, selanjutnya dilakukan pengujian dengan ukuran populasi sebesar 100 dan ukuran generasi 100 sebanyak 10 kali. Hasil pengujian kombinasi *cr* dan *mr* dapat ditunjukkan Tabel 6.1 dan Tabel 6.2.

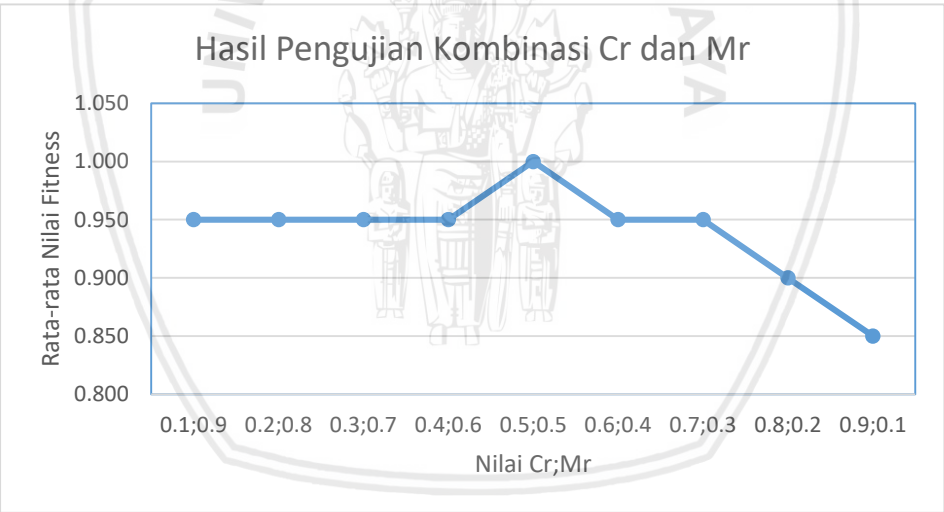
Tabel 6.1 Hasil pengujian nilai *fitness* kombinasi *crossover rate* dan *mutation rate*

Cr;Mr	Pengujian ke										Rata-rata Nilai *Fitness*
	1	2	3	4	5	6	7	8	9	10	
0,1;0,9	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	0,500	1,000	0,950
0,2;0,8	1,000	1,000	1,000	1,000	1,000	1,000	1,000	0,500	1,000	1,000	0,950
0,3;0,7	1,000	1,000	1,000	1,000	0,500	1,000	1,000	1,000	1,000	1,000	0,950
0,4;0,6	1,000	1,000	1,000	1,000	1,000	1,000	1,000	0,500	1,000	1,000	0,950
0,5;0,5	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
0,6;0,4	1,000	1,000	1,000	1,000	1,000	0,500	1,000	1,000	1,000	1,000	0,950
0,7;0,3	1,000	1,000	0,500	1,000	1,000	1,000	1,000	1,000	1,000	1,000	0,950
0,8;0,2	1,000	0,500	1,000	1,000	1,000	1,000	0,500	1,000	1,000	1,000	0,900
0,9;0,1	1,000	0,500	0,500	0,500	1,000	1,000	1,000	1,000	1,000	1,000	0,850

Tabel 6.2 Hasil pengujian waktu komputasi kombinasi *crossover rate* dan *mutation rate*

Cr;Mr	Pengujian ke										Rata-rata Waktu Komputasi (menit)
	1	2	3	4	5	6	7	8	9	10	
0,1;0,9	26,49	26,72	26,56	27,51	26,06	25,97	27,77	26,33	31,83	26,50	27,17
0,2;0,8	25,84	25,82	26,51	26,67	26,51	26,74	25,35	25,42	25,84	26,37	26,11
0,3;0,7	25,75	26,08	25,75	26,83	25,50	26,17	25,31	25,73	25,62	25,66	25,84
0,4;0,6	25,03	25,52	25,63	25,29	25,13	25,58	26,61	26,91	25,73	26,54	25,80
0,5;0,5	25,71	25,52	26,14	25,15	25,52	25,17	25,41	25,14	25,65	24,81	25,42
0,6;0,4	24,82	25,47	25,65	25,90	25,50	26,11	27,75	26,01	26,24	25,10	25,85
0,7;0,3	25,83	26,39	25,73	25,31	25,69	26,75	25,47	25,64	25,25	25,89	25,79
0,8;0,2	25,15	25,65	25,78	25,98	25,49	25,57	25,25	25,21	25,45	25,15	25,47
0,9;0,1	25,22	25,50	25,55	25,41	25,43	25,49	25,53	25,12	25,73	25,53	25,45

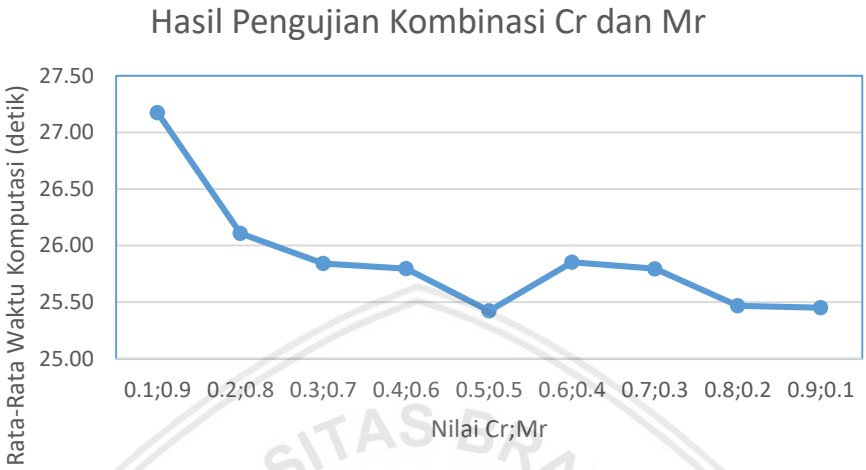
Berdasarkan pada Tabel 6.1 dan Tabel 6.2 didapatkan grafik nilai rata-rata *fitness* dan waktu komputasi hasil pengujian kombinasi *cr* dan *mr* yang ditunjukkan pada Gambar 6.1 dan Gambar 6.2.



Gambar 6.1 Hasil pengujian nilai *fitness* kombinasi *crossover rate* dan *mutation rate*

Berdasarkan Gambar 6.1 menunjukkan nilai *fitness* tertinggi didapatkan saat kombinasi nilai *cr* 0,5 dan *mr* 0,5 sebesar 1,000 dan rata-rata nilai *fitness* terendah saat kombinasi nilai *cr* 0,9 dan *mr* 0,1 sebesar 0,850. Gambar 6.1 menunjukkan grafik nilai fitness mengalami nilai yang sama mulai dari kombinasi *cr* 0,1 dan *mr* 0,9 sampai dengan *cr* 0,4 dan *mr* 0,6, kemudian mengalami kenaikan nilai pada kombinasi *cr* 0,5, *mr* 0,5 yaitu bernilai 1,000, setelahnya mengalami penurunan nilai sampai dengan *cr* 0,9 dan *mr* 0,1. Keadaan tersebut menunjukkan bahwa untuk mendapatkan nilai *fitness* yang optimal maka memerlukan kombinasi nilai yang seimbang antara *cr* dan *mr.* Dalam proses pencarian solusi, nilai *cr* yang terlalu tinggi akan cenderung bersifat eksploitasi yaitu pada saat

proses reproduksi akan dihasilkan anak yang mirip dengan induknya sehingga keragaman individu semakin berkurang (Azzakky, et al., 2018). Begitu juga jika nilai *mr* terlalu tinggi akan cenderung bersifat eksplorasi sehingga tidak menutup kemungkinan akan memperlama dalam mendapatkan solusi yang optimal.



Gambar 6.2 Hasil pengujian waktu komputasi kombinasi *crossover rate* dan *mutation rate*

Berdasarkan Gambar 6.2 ditunjukkan waktu komputasi tertinggi didapatkan saat kombinasi nilai *cr* 0,1 dan *mr* 0,9 sebesar 27,17 detik dan waktu komputasi terendah saat kombinasi nilai *cr* 0,5 dan *mr* 0,5 sebesar 25,42 detik. Gambar 6.1 menunjukkan waktu komputasi mengalami penurunan nilai mulai dari kombinasi *cr* 0,1 dan *mr* 0,9 sampai dengan *cr* 0,5 dan *mr* 0,5, namun pada kombinasi *cr* 0,6 dan *mr* 0,4 mengalami peningkatan nilai dari kombinasi sebelumnya, setelahnya mengalami penurunan waktu komputasi kembali sampai nilai *cr* 0,9 dan *mr* 0,1. Keadaan tersebut menunjukkan nilai *mr* yang terlalu tinggi akan memperbesar waktu komputasinya saat proses reproduksi tahap mutasi.

6.2 Pengujian Ukuran Populasi

Pengujian ukuran populasi akan dilakukan untuk mencari ukuran populasi terbaik untuk mendapatkan rata-rata nilai *fitness* yang optimal dan dan mengetahui rata-rata waktu komputasi yang dihasilkan. Pengujian ukuran populasi ini mengacu pada perancangan pengujian ukuran populasi pada Tabel 4.20 dan Tabel 4.21. Uji coba dilakukukan dengan populasi kelipatan 10, yaitu rentang nilai 10 hingga 100 dengan 10 kali uji coba pada masing-masing ukuran populasi. Nilai *cr* dan *mr* yang digunakan yaitu 0,5 dan 0,5 dengan ukuran generasi yaitu 100. Hasil Pengujian ukuran populasi ditunjukkan pada Tabel 6.3 dan Tabel 6.4.

Tabel 6.3 Hasil pengujian nilai *fitness* ukuran populasi

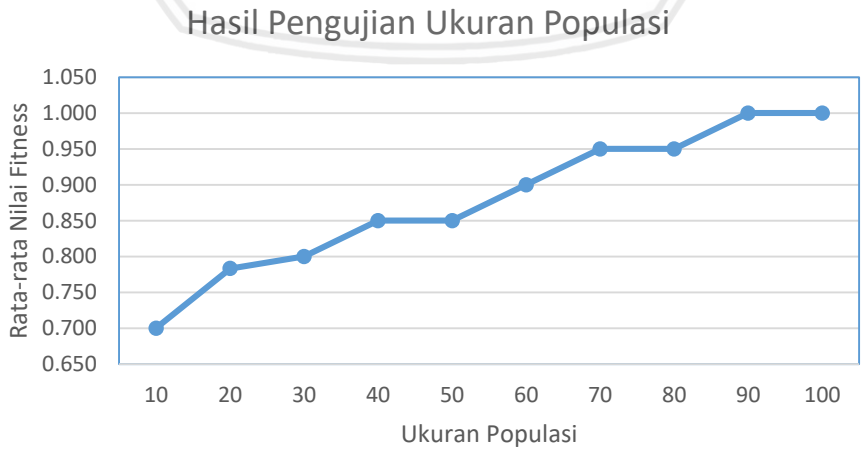
Populasi	Pengujian ke										Rata-rata Nilai *Fitness*
	1	2	3	4	5	6	7	8	9	10	
10	0,500	0,500	1,000	1,000	0,333	0,333	1,000	1,000	0,333	1,000	0,700

20	1,000	1,000	0,500	0,500	0,333	1,000	1,000	1,000	1,000	0,500	0,783
30	1,000	0,500	0,500	1,000	1,000	0,500	1,000	1,000	0,500	0,500	0,800
40	1,000	0,500	1,000	1,000	0,500	1,000	1,000	1,000	1,000	0,500	0,850
50	1,000	1,000	0,500	1,000	0,500	1,000	0,500	1,000	1,000	1,000	0,850
60	1,000	0,500	1,000	1,000	1,000	1,000	1,000	1,000	0,500	1,000	0,900
70	1,000	1,000	0,500	1,000	1,000	1,000	1,000	1,000	1,000	1,000	0,950
80	1,000	1,000	1,000	1,000	1,000	1,000	1,000	0,500	1,000	1,000	0,950
90	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
100	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000

Tabel 6.4 Hasil pengujian waktu komputasi ukuran populasi

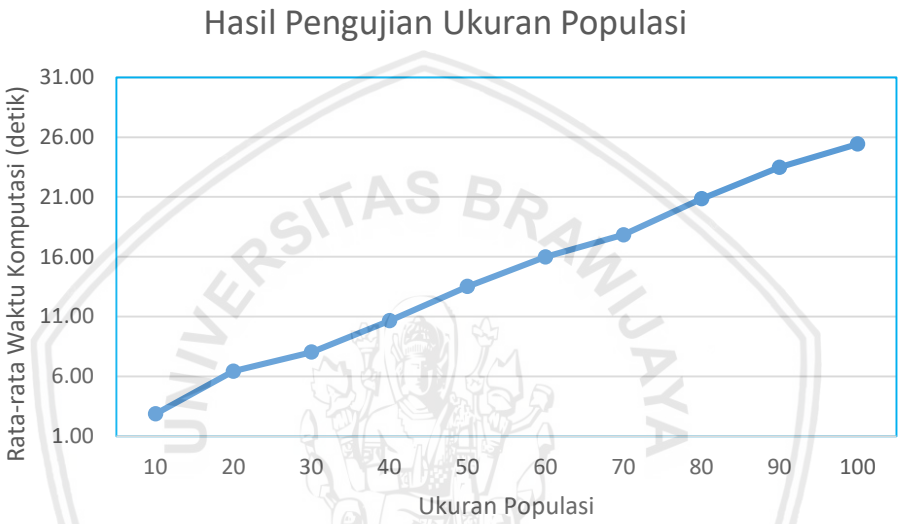
Populasi	Pengujian ke										Rata-rata Waktu Komputasi (menit)
	1	2	3	4	5	6	7	8	9	10	
10	2,88	2,91	2,78	2,81	2,89	2,87	2,86	2,88	2,89	2,83	2,86
20	7,08	7,16	5,33	5,37	5,44	7,10	7,09	7,16	7,18	5,36	6,43
30	7,83	7,91	7,94	7,95	8,72	7,86	8,19	7,92	7,99	7,91	8,02
40	11,02	10,26	10,37	10,34	10,36	10,53	10,51	10,42	12,02	10,63	10,65
50	13,49	13,50	13,11	13,90	12,88	14,00	13,80	13,31	13,45	13,65	13,51
60	15,29	15,37	15,69	15,29	15,36	15,22	16,43	17,45	16,02	17,74	15,99
70	17,74	17,53	17,87	17,92	18,09	17,67	17,50	17,96	17,86	18,15	17,83
80	20,21	20,86	20,25	20,47	20,10	20,47	20,74	24,96	20,08	20,35	20,85
90	22,92	23,06	23,12	22,58	23,66	23,05	23,59	24,08	23,89	24,85	23,48
100	25,71	25,52	26,14	25,15	25,52	25,17	25,41	25,14	25,65	24,81	25,42

Berdasarkan pada Tabel 6.3 dan Tabel 6.4 didapatkan grafik nilai rata-rata *fitness* dan waktu komputasi hasil pengujian ukuran populasi yang ditunjukkan pada Gambar 6.3 dan Gambar 6.4.



Gambar 6.3 Hasil pengujian nilai *fitness* ukuran populasi

Berdasarkan Gambar 6.3 menunjukkan nilai *fitness* tertinggi didapatkan saat ukuran populasi 90 sebesar 1,000 dan rata-rata nilai *fitness* terendah saat ukuran populasi 10 sebesar 0,700. Gambar 6.3 menunjukkan grafik nilai *fitness* mengalami peningkatan nilai mulai dari ukuran populasi 10 sampai dengan 90, setelahnya mengalami konvergensi nilai. Hal tersebut disebabkan karena ukuran populasi yang besar akan dihasilkan individu yang semakin beragam melalui proses reproduksi, namun ukuran populasi yang besar juga tidak memberikan jaminan akan menghasilkan nilai *fitness* yang tinggi disebabkan pembangkitan populasi yang dilakukan secara acak (Sanapiah, et al., 2018). Ukuran populasi terlalu kecil akan memperkecil peluang dalam memperoleh solusi yang optimal.



Gambar 6.4 Hasil pengujian waktu komputasi ukuran populasi

Berdasarkan Gambar 6.4 menunjukkan rata-rata waktu komputasi tertinggi didapatkan saat ukuran populasi 100 sebesar 25,42 detik dan rata-rata waktu komputasi terendah saat ukuran populasi 10 sebesar 2,86 detik. Gambar 6.4 menunjukkan grafik waktu komputasi mengalami peningkatan nilai mulai dari ukuran populasi 10 sampai dengan 100. Hal tersebut menunjukkan bahwa semakin besar ukuran populasi yang digunakan maka waktu komputasi yang diperlukan juga akan semakin besar.

6.3 Pengujian Ukuran Generasi

Pengujian ukuran generasi akan dilakukan untuk mengetahui pengaruh ukuran generasi dalam mendapatkan rata-rata nilai *fitness* dan mengetahui waktu komputasi yang dihasilkan. Pengujian ukuran generasi ini mengacu pada perancangan pengujian ukuran populasi pada Tabel 4.21 dan Tabel 4.22. Uji coba dilakukukan dengan ukuran generasi dengan rentang nilai 10 hingga 100 sebanyak 10 kali uji coba pada masing-masing ukuran generasi. Ukuran populasi yang digunakan sebesar 90 dengan kombinasi nilai *cr* 0,5 dan *mr* 0,5. Hasil pengujian jumlah generasi ditunjukkan pada Tabel 6.5 dan Tabel 6.6.

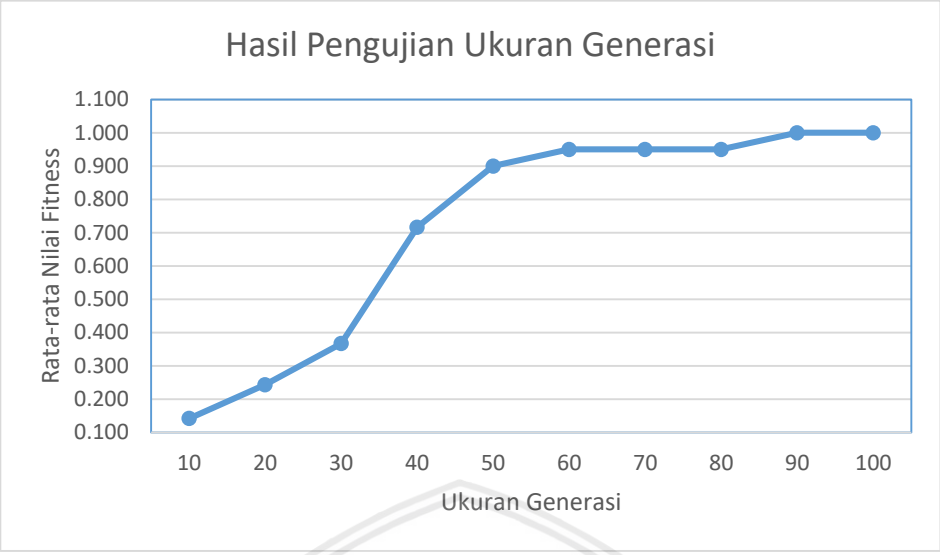
Tabel 6.5 Hasil pengujian nilai *fitness* ukuran generasi

Generasi	Pengujian ke										Rata-rata Nilai *Fitness*
	1	2	3	4	5	6	7	8	9	10	
10	0,143	0,143	0,167	0,143	0,143	0,143	0,125	0,143	0,125	0,143	0,142
20	0,333	0,250	0,200	0,200	0,250	0,250	0,200	0,250	0,250	0,250	0,243
30	0,333	0,500	0,333	0,500	0,250	0,500	0,333	0,333	0,333	0,250	0,367
40	1,000	0,500	1,000	0,500	0,500	1,000	1,000	0,333	1,000	0,333	0,717
50	1,000	1,000	1,000	0,500	1,000	1,000	1,000	0,500	1,000	1,000	0,900
60	1,000	1,000	1,000	0,500	1,000	1,000	1,000	1,000	1,000	1,000	0,950
70	1,000	1,000	1,000	0,500	1,000	1,000	1,000	1,000	1,000	1,000	0,950
80	1,000	1,000	1,000	0,500	1,000	1,000	1,000	1,000	1,000	1,000	0,950
90	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
100	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000

Tabel 6.6 Hasil pengujian waktu komputasi ukuran generasi

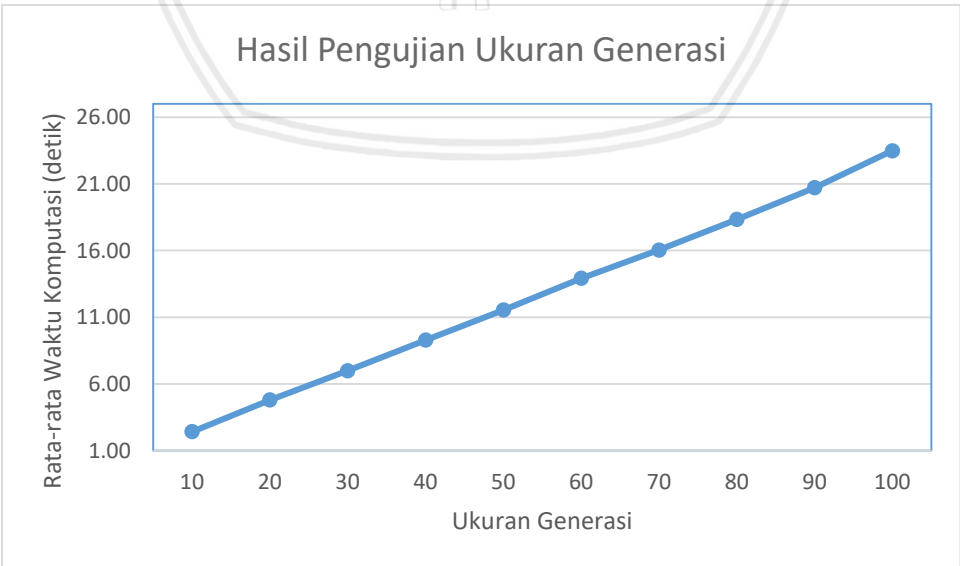
Generasi	Pengujian ke										Rata-rata Waktu Komputasi (menit)
	1	2	3	4	5	6	7	8	9	10	
10	2,34	2,32	2,34	2,36	2,85	2,44	2,41	2,42	2,33	2,33	2,41
20	5,55	4,73	4,78	4,65	4,66	4,88	4,72	4,64	4,65	4,66	4,79
30	6,91	7,04	6,97	6,93	6,98	6,93	7,08	7,02	7,08	6,91	6,99
40	9,22	9,42	9,32	9,13	9,27	9,32	9,29	9,48	9,29	9,11	9,28
50	11,89	11,43	11,47	11,27	11,68	12,02	11,51	11,37	11,38	11,37	11,54
60	14,31	14,18	14,05	13,82	13,68	14,02	13,67	13,83	13,63	13,93	13,91
70	16,20	16,21	15,98	15,93	15,89	16,08	16,15	16,10	16,02	15,80	16,04
80	18,26	18,60	18,33	18,23	18,54	18,33	18,07	18,51	18,54	17,83	18,32
90	20,40	18,63	21,97	20,74	22,14	20,60	20,45	20,59	20,75	20,75	20,70
100	22,92	23,06	23,12	22,58	23,66	23,05	23,59	24,08	23,89	24,85	23,48

Berdasarkan pada Tabel 6.5 dan Tabel 6.6 didapatkan grafik nilai rata-rata *fitness* dan waktu komputasi hasil pengujian ukuran generasi yang ditunjukkan pada Gambar 6.5 dan Gambar 6.6.



Gambar 6.5 Hasil pengujian nilai *fitness* ukuran generasi

Berdasarkan Gambar 6.5 menunjukkan nilai *fitness* tertinggi didapatkan saat ukuran generasi 90 sebesar 1,000 dan rata-rata nilai *fitness* terendah saat ukuran generasi 10 sebesar 0,142. Hal tersebut dikarenakan ukuran generasi yang terlalu kecil akan membatasi eksplorasi ruang pencarian dalam menemukan solusi yang optimal begitu juga sebaliknya ukuran generasi yang besar akan memberi peluang untuk melakukan eksplorasi ruang pencarian dalam menemukan solusi optimal yang lebih besar. Gambar 6.5 menunjukkan grafik nilai *fitness* mengalami peningkatan nilai mulai dari ukuran populasi 10 sampai dengan 90, namun setelahnya nilai *fitness* tidak mengalami perubahan. Jika pengujian ukuran generasi dilanjutkan dengan ukuran generasi lebih tinggi maka nilai *fitness* tidak memiliki selisih yang besar dan individu yang dihasilkan tidak berbeda jauh dengan induknya (Shafaat, et al., 2018).



Gambar 6.6 Hasil pengujian waktu komputasi ukuran generasi

Berdasarkan Gambar 6.6 menunjukkan grafik rata-rata waktu komputasi mengalami peningkatan nilai mulai dari ukuran generasi 10 sampai dengan 100. Selisih rata-rata waktu komputasi pada ukuran generasi tiap kelipatan 100 yaitu sebesar 2,34 detik. Peningkatan waktu komputasi tersebut menunjukkan bahwa semakin besar besar ukuran generasi yang digunakan maka proses eksplorasi yang dilakukan semakin besar sehingga waktu komputasi yang diperlukan juga akan besar.

6.4 Hasil Analisis Global

Berdasarkan tiga pengujian sebanyak 25 mahasiswa yang telah dilakukan, yaitu pengujian kombinasi nilai *crossover rate* (*cr*) dan *mutation rate (mr)*, pengujian ukuran populasi, dan pengujian ukuran generasi diperoleh nilai *fitness* yang optimal pada kombinasi nilai *crossover rate* (*cr*) dan *mutation rate (mr)* adalah 0,5 dan 0,5. Nilai masing-masing *crossover rate* dan *mutation rate* yang semakin besar dapat meningkatkan kemampuan eksploitasi dan eksplorasi dalam proses mencari solusi yang optimal. Akan tetapi hal tersebut akan membebani waktu komputasi karena bisa jadi algoritme genetika mengeksploitasi pada area yang sedikit memiliki nilai optimal atau mengeksplorasi area yang tidak memiliki nilai optimal. Pada pengujian ukuran populasi diperoleh nilai *fitness* yang optimal pada ukuran sebesar 90 populasi. Ukuran populasi yang besar menghasilkan individu yang semakin beragam, namun ukuran populasi yang besar juga tidak memberikan jaminan akan menghasilkan nilai *fitness* yang tinggi disebabkan pembangkitan populasi yang dilakukan secara acak dan membebani waktu komputasi. Kemudian pada pengujian ukuran generasi diperoleh nilai *fitness* yang optimal pada ukuran sebesar 90 generasi. Ukuran populasi yang besar akan memberikan peluang memperoleh solusi optimal yang lebih besar.

Setelah semua nilai parameter dari algoritme genetika yang optimal didapatkan, selanjutnya diterapkan ke dalam sistem penempatan ruang sidang skripsi dan diperoleh rata-rata nilai *fitness* yang optimal sebesar 1. Hal tersebut menunjukkan bahwa tidak ada *hard constraint* maupun *soft constraint* yang dilanggar.

Tabel 6.7 Perbandingan detail pelanggaran *constraint*

No	*Constraint*	Jumlah pelanggaran penjadwalan dengan sistem	Jumlah pelanggaran penjadwalan manual
1	Tidak boleh terdapat jadwal yang sama dalam satu hari	0	0
2	Dua dosen penguji atau dua dosen pembimbing tidak boleh bentrok pada waktu sidang yang sama	0	0
3	Dua dosen penguji atau dua dosen pembimbing dengan sesi beruntun harus pada ruang sidang yang sama	0	22
Jumlah Penalti		0	22
Fitness		1,000	0,043

Tabel 6.8 Perbandingan detail pelanggaran *constraint* ke-3

No	Waktu	Jumlah pelanggaran penjadwalan dengan sistem	Jumlah pelanggaran penjadwalan manual
1	Sesi 1 ke sesi 2	0	3
2	Sesi 2 ke sesi 3	0	10
3	Sesi 4 ke sesi 5	0	6
4	Sesi 5 ke sesi 6	0	3
5	Sesi 6 ke sesi 7	0	0

Berdasarkan pada detail pelanggaran Tabel 6.7 diperoleh jadwal yang optimal dengan nilai *fitness* sebesar 1 yaitu pada kombinasi nilai *cr* 0,5 dan *mr* 0,5, ukuran populasi sebesar 90, dan ukuran generasi sebesar 90. Hasil penempatan ruang sidang skripsi yang optimal ditunjukkan pada Tabel 6.9.

Tabel 6.9 Hasil penjadwalan sistem

No	Mahasiswa	Pembimbing 1	Pembimbing 2	Penguji 1	Penguji 2	Ruang	Waktu
1	Irma Pujadayanti	Mochammad Ali Fauzi, S.Kom, M.Kom	Yuita Arum Sari, S.Kom., M.Kom	Imam Cholissodin, S.Si, M.Kom	Sigit Adinugroho, S.Kom., M.Sc	E2.5	sesi 1
2	Rizky Kharisma	Dr. Eng. Herman Tolle, S.T, M.T	Niken Hendrakusma Wardani, S.Kom., M.Kom.	Adam Hendra Brata, S.Kom., M.T., M.Sc.	Yuita Arum Sari, S.Kom., M.Kom	E2.3	sesi 4
3	Muhammad Rasyid Ridho	Aryo Pinandito, S.T, M.MT	Ratih Kartika Dewi, S.T., M.Kom	Agi Putra Kharisma, S.T, M.T	Komang Candra Brata, S.Kom., M.T., M.Sc.	E2.3	sesi 1
4	Jodi Prayoga Wahyudwi	Rakhmadhany Primananda, S.T, M.Kom	Achmad Basuki, S.T, M.MG, Ph.D	Eko Sakti Pramukantoro, S.Kom, M.Kom	Fariz Andri Bakhtiar, S.T., M.Kom.	F3.4	sesi 3
5	Regina Anky Chandra	Edy Santoso, S.Si, M.Kom	Sigit Adinugroho, S.Kom., M.Sc	Putra Pandu Adikara, S.Kom, M.Kom	Candra Dewi, S.Kom, M.Sc	E2.3	sesi 5
6	Ahmad Nur Royyan	Indriati, S.T, M.Kom	Lailil Muflikhah, S.Kom, M.Sc	Mochammad Ali Fauzi, S.Kom, M.Kom	Rizal Setya Perdana, S.Kom, M.Kom	F4.1	sesi 5
7	Yoshua Aditya Kurnia	Eriq Muhammad Adams Jonemaro, S.T, M.Kom	Muhammad Aminul Akbar , S.Kom., M.T	Wibisono Sukmo Wardhono, S.T, M.T	Tri Afirianto, S.T, M.T	F3.3	sesi 7

8	Yane Marita Febrianti	Indriati, S.T, M.Kom	Agus Wahyu Widodo, S.T, M.Cs	Suprapto, S.T, M.T	Ratih Kartika Dewi, S.T., M.Kom	E2.3	sesi 3
9	Roliand Prasetya	Fajar Pradana, S.ST, M.Eng	Achmad Arwan, S.Kom, M.Kom	Tri Astoto Kurniawan, S.T, M.T, Ph.D	Denny Sagita Rusdianto, S.Kom, M.Kom	F4.8	sesi 3
10	Robihamanto	Adam Hendra Brata, S.Kom., M.T., M.Sc.	Komang Candra Brata, S.Kom., M.T., M.Sc.	Mahardeka Tri Ananta, S.Kom., M.T., M.Sc.	Agi Putra Kharisma, S.T, M.T	F3.4	sesi 7
11	M. Rizzo Irfan	Mochammad Ali Fauzi, S.Kom, M.Kom	Tibyani , S.T, M.T	Putra Pandu Adikara, S.Kom, M.Kom	Sigit Adinugroho, S.Kom., M.Sc	E2.8	sesi 7
12	Fredianto	Ari Kusyanti, S.T, M.Sc	Kasyful Amron, S.T, M.Sc	Dany Primanita Kartikasari, S.T., M.Kom	Mahendra Data, S.Kom., M.Kom	F3.5	sesi 6
13	Rheza Raditya Andrianto	Lailil Muflikhah, S.Kom, M.Sc	Bayu Rahayudi , S.T, M.T	Candra Dewi, S.Kom, M.Sc	Imam Cholissodin, S.Si, M.Kom	F3.5	sesi 3
14	Jumerlyanti Mase	Muhammad Tanzil Furqon, S.Kom, M.CompSc	Bayu Rahayudi , S.T, M.T	Achmad Arwan, S.Kom, M.Kom	Dian Eka Ratnawati, S.Si, M.Kom	E2.3	sesi 7
15	Salma Mutiasanti	Mahardeka Tri Ananta, S.Kom., M.T., M.Sc.	Hanifah Muslimah Az-Zahra, S.Sn., M.Ds.	Lutfi Fanani, S.Kom., M.T., M.Sc.	Ratih Kartika Dewi, S.T., M.Kom	F3.3	sesi 5
16	Amirrulloh Acmad K. A.	Fajar Pradana, S.ST, M.Eng	Bayu Priyambadha, S.Kom, M.Kom	Denny Sagita Rusdianto, S.Kom, M.Kom	Nanang Yudi Setiawan, S.T., M.Kom.	E2.7	sesi 5

17	Yoga Faodiansyah	Kasyful Amron, S.T, M.Sc	Eko Sakti Pramukantoro, S.Kom, M.Kom	Ir. Primantara Hari Trisnawan, M. Sc.	Mahendra Data, S.Kom., M.Kom	E2.1	sesi 4
18	Yobel Leonardo Tampubolon	Lailil Muflikhah, S.Kom, M.Sc	Indriati, S.T, M.Kom	Budi Darma Setiawan, S.Kom, M.Cs	Muhammad Aminul Akbar , S.Kom., M.T	F4.1	sesi 4
19	Irwin Deriyan Ferdiansyah	Sigit Adinugroho, S.Kom., M.Sc	Mochammad Ali Fauzi, S.Kom, M.Kom	Dr.Eng. Fitra Abdurrachman Bachtiar, S.T, M.Eng	Yuita Arum Sari, S.Kom., M.Kom	F3.3	sesi 3
20	Andika Eka Putra	Nurul Hidayat, S.Pd, M.Sc	Imam Cholissodin, S.Si, M.Kom	Edy Santoso, S.Si, M.Kom	Issa Arwani, S.Kom, M.Sc	F4.8	sesi 7
21	Mercury Fluorida Fibrianda	Adhitya Bhawiyuga, S.Kom, M.Sc	Achmad Basuki, S.T, M.MG, Ph.D	Dany Primanita Kartikasari, S.T., M.Kom	Reza Andria Siregar, S.T., M.Kom.	F3.4	sesi 1
22	Fitria Dwi Nurhayati	Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D	Achmad Arwan, S.Kom, M.Kom	Satrio Agung Wicaksono, S.Kom, M.Kom	Dian Eka Ratnawati, S.Si, M.Kom	E2.2	sesi 4
23	Dhimas Tungga Satya	Nurul Hidayat, S.Pd, M.Sc	Ir. Sutrisno, M.T	Yuita Arum Sari, S.Kom., M.Kom	Suprapto, S.T, M.T	F4.8	sesi 6
24	Hermawan Wijaya	Wibisono Sukmo Wardhono, S.T, M.T	Issa Arwani, S.Kom, M.Sc	Muhammad Aminul Akbar , S.Kom., M.T	Tri Afirianto, S.T, M.T	B1.6	sesi 3
25	Firadi Surya Pramana	Eriq Muhammad Adams Jonemaro, S.T, M.Kom	Muhammad Aminul Akbar , S.Kom., M.T	Wibisono Sukmo Wardhono, S.T, M.T	Tri Afirianto, S.T, M.T	E2.8	sesi 1

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan penelitian yang telah dilakukan untuk permasalahan optimasi penempatan ruang sidang skripsi didapatkan kesimpulan sebagai berikut:

1. Berdasarkan hasil pengujian yang telah dilakukan, dalam proses optimasi menggunakan algoritme genetika terdapat beberapa parameter yang mempengaruhinya yaitu kombinasi nilai *crossover rate (cr)* dan *mutation rate (mr),* ukuran populasi, dan jumlah generasi. Pada proses pengujian kombinasi nilai *crossover rate (cr)* dan *mutation rate (mr)* didapatkan solusi yang optimal dengan rata-rata nilai *fitness* tertinggi sebesar 1,000 pada nilai *cr* 0,5 dan *mr* 0,5 sedangkan solusi terendah dengan rata-rata nilai *fitness* sebesar 0,850 pada kombinasi nilai *cr* 0,9 dan *mr* 0,1. Nilai masing-masing crossover rate dan mutation rate yang semakin besar dapat meningkatkan kemampuan eksploitasi dan eksplorasi dalam proses mencari solusi yang optimal. Akan tetapi hal tersebut akan membebani waktu komputasi karena bisa jadi algoritme genetika mengeksploitasi pada area yang sedikit memiliki nilai optimal atau mengeksplorasi area yang tidak memiliki nilai optimal. Pada proses pengujian ukuran populasi diperoleh rata-rata nilai *fitness* tertinggi yaitu 1,000 pada ukuran 90 populasi sedangakan rata-rata nilai *fitness* terendah saat ukuran populasi 10 sebesar 0,700. Ukuran populasi yang besar menghasilkan individu yang semakin beragam, namun ukuran populasi yang besar juga tidak memberikan jaminan akan menghasilkan nilai fitness yang tinggi disebabkan pembangkitan populasi yang dilakukan secara acak dan juga membebani waktu komputasi. Selanjutnya pada proses pengujian ukuran generasi dengan menggunakan ukuran 10 sampai 100 generasi diperoleh rata-rata nilai *fitness* optimal sebesar 1,000 pada ukuran generasi 90. Ukuran populasi yang besar akan memberikan peluang memperoleh solusi optimal yang lebih besar. Akan tetapi, ukuran generasi yang telah mencapai pada nilai *fitness* yang optimal jika diteruskan dengan ukuran yang lebih besar, dapat diperoleh nilai *fitness* dan individu yang konvergen atau tidak berbeda jauh dari hasil sebelumnya.
2. Kualitas solusi yang diperoleh dapat diukur dari hasil akhir penjadwalan yang dilakukan oleh sistem dibandingkan dengan dilakukan manual. Pada proses penjadwalan tanpa sistem, akan terjadi banyak pelanggaran di *constraint* 3 yaitu ruang yang berbeda pada sesi yang berlanjutan. Pada proses dengan sistem didapatkan hasil solusi dengan tidak melanggar *constraint* sama sekali.

7.2 Saran

Berdasarkan penelitian yang telah dilakukan, masih terdapat kekurangan pada sistem dan untuk pengembangan penelitian selanjutnya diperlukan beberapa saran dalam menghasilkan solusi optimasi sebagai berikut:

1. Algoritme genetika merupakan algoritme yang bersifat *stochastic* yaitu setiap kali proses dijalankan akan menghasilkan solusi yang berbeda-beda sehingga pada penelitian selanjutnya diperlukan perbaikan pada strategi dalam merepresentasikan kromosom ketika solusi yang dihasilkan jauh dari optimal.
2. Untuk proses penjadwalan berikutnya dapat menambahkan slot sesi sidang tambahan sehingga untuk penjadwalan dengan jumlah mahasiswa yang banyak tetap dihasilkan solusi yang optimal.

DAFTAR REFERENSI

Adnyana, I. M. B. & Wijayana, I. K., 2017. Rancang Bangun Sistem Penempatan ruang sidang skripsi Menggunakan Algoritma Genetika. *JURNAL SISTEM DAN INFORMATIKA STIKOM Bali,* 12(1), pp. 38-47.

Ardhianto, B., Pudjoatmodjo, B. & Suliiyo, M. D., 2014. Implementasi Algoritma Genetika pada Penjadwalan Sidang Tugas Akhir Teknik Informatika.

Auliyah, N., Putri, R. R. M. & Suprapto, 2018. Optimasi Penjadwalan Moving Class Menggunakan Algoritma Genetika (Studi Kasus: SMA Negeri 1 Turatea). *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* 2(4), pp. 1768-1777.

Azzakky, R. U., Setiawan, B. D. & Wijoyo, S. H., 2018. Optimasi Penjadwalan Mata Pelajaran Pondok Pesantren Mahasiswa Menggunakan Algoritme Genetika (Studi Kasus: Yayasan Bina Insani Sukses Malang). *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* 2(11), pp. 4407-4413.

Budi, W. P. S., 2013. Optimasi Traveling Salesman Problem dengan Algoritma Genetika menggunakan Operator Partially Matched Crossover.

Husada, H., Cholissodin, I. & Bachtiar, F. A., 2018. Optimasi Penjadwalan Kuliah Pengganti Menggunakan Algoritme Genetika. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* 2(9), pp. 2829-2834.

Luber, M. M. H., Cholissodin, I. & Dewi, C., 2017. Optimasi Penjadwalan Damping Mahasiswa Difabel Menggunakan Algoritma Genetika (Studi Kasus PSLD Universitas Brawijaya). *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* 1(9), pp. 774-782.

Mahmudy, W. F., 2015. Dasar-Dasar Algoritma Evolusi. *Malang : Program Teknologi Informasi dan Ilmu Komputer (PTIIK).*

Mariana & Hiryanto, L., 2013. Penjadwalan Kelas MataKuliah Menggunakan Vertex Graph Coloring dan Simulated Annealing.

Muhyi, Y., 2008. Penjadwalan Kuliah Otomatis dengan Constraint Programming.

Pramudita, O. V., Nhita, F. & Aditsania, A., 2016. Penjadwalan Sidang Tugas Akhir Prodi Ilmu Komputasi Universitas Telkom Menggunakan Metode Algoritma Genetika Adaptif dan Fuzzy Relation. *e-Proceeding of Engineering ,* 3(2), p. 3825.

Sanapiah, M. D. S., Setiawan, B. D. & Widodo, A. W., 2018. Optimasi Menu Makanan Atlet Berdasarkan Jadwal Latihan Menggunakan Algoritme Genetika. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* 2(11), pp. 4802-4811.

Shafaat, M., Cholissodin, I. & Santoso, E., 2018. Optimasi Komposisi Makanan Diet Bagi Penderita Hipertensi menggunakan Algoritme Genetika. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* pp. 226-236.

Sihombing, R. S., 2014. Pemanfaatan Algoritma Genetika Pada Aplikasi Penempatan Buku Untuk Perpustakaan Sekolah. *Pelita Informatika Budi Darma,,* 4(2), pp. 113-118.