APLIKASI PENJADWALAN PRAKTIKUM LABORATORIUM INFORMATIKA UNIVERSITAS AHMAD DAHLAN MENGGUNAKAN ALGORITMA GENETIKA

SKRIPSI

Disusun untuk memenuhi sebagian persyaratan
mencapai derajat Sarjana

Disusun Oleh:

ARVIN DESTANTYA RYANANDRA RAMADHAN
1900018239

PROGRAM STUDI S1 INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS AHMAD DAHLAN

2024

LEMBAR PERSETUJUAN PEMBIMBING

SKRIPSI

APLIKASI PENJADWALAN PRAKTIKUM LABORATORIUM INFORMATIKA UNIVERSITAS AHMAD DAHLAN MENGGUNAKAN ALGORITMA GENETIKA

Dipersiapkan dan disusun oleh:

ARVIN DESTANTYA RYANANDRA RAMADHAN
1900018239

Program Studi S1 Informatika
Fakultas Teknologi Industri
Universitas Ahmad Dahlan

Telah disetujui oleh:

Pembimbing

Dr. Ardiansyah, S.T., M.Cs.

NIPM. 19790723 200309 111 0932301

LEMBAR PENGESAHAN

SKRIPSI

APLIKASI PENJADWALAN PRAKTIKUM LABORATORIUM INFORMATIKA UNIVERSITAS AHMAD DAHLAN MENGGUNAKAN ALGORITMA GENETIKA

Dipersiapkan dan disusun oleh:

ARVIN DESTANTYA RYANANDRA RAMADHAN
1900018239

Telah dipertahankan di depan Dewan Penguji
pada 15 Mei 2024
dan dinyatakan telah memenuhi syarat

Susunan Dewan Penguji

Ketua	**: Dr. Ardiansyah, S.T., M.Cs.**	**……**
Penguji 1	**: Murein Miksa Mardhia, S.T., M.T.**	**……**
Penguji 2	**: Miftahurrahma Rosyda, S.Kom., M.Eng.**	**……**

Yogyakarta, 24 Mei 2024
Dekan Fakultas Teknologi Industri
Universitas Ahmad Dahlan

Prof. Dr. Ir. Siti Jamilatun, M.T.
NIPM. 196608121996010110784324

LEMBAR PERNYATAAN KEASLIAN

SURAT PERNYATAAN

Yang bertanda tangan di bawah ini:

Nama : Arvin Destantya Ryanandra Ramadhan

NIM : 1900018239

Prodi : Informatika

Judul TA/Skripsi : Aplikasi Penjadwalan Praktikum Laboratorium Informatika Universitas Ahmad Dahlan menggunakan Algoritma Genetika

Dengan ini saya menyatakan bahwa Laporan Tugas Akhir ini tidak terdapat karya yang pernah diajukan untuk memperoleh gelar Ahli Madya/Kesarjanaan di suatu Perguruan Tinggi, dan sepanjang pengetahuan saya juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka.

Yogyakarta, 7 Mei 2024

Mengetahui, Dosen Pembimbing	Yang menyatakan,
Dr. Ardiansyah, S.T., M.Cs.	Arvin Destantya Ryanandra Ramadhan
NIP/NIPM. 19790723 200309 111 0932301	1900018239

KATA PENGANTAR

Puji Syukur penulis panjatkan kehadirat Allah SWT, karena atas Rahmat dan hidayahnya, penulis dapat menyelesaikan peulisan skripsi dengan judul “Aplikasi Penjadwalan Praktikum Laboratorium Informatika Universitas Ahmad Dahlan menggunakan Algoritma Genetika “. Penulis ingin menyampaikan penghargaan dan terima kasih yang setinggi-tingginya kepada semua pihak yang telah memberikan dukungan, bimbingan, motivasi selama penulisan skripsi ini. Penulis juga ingin mengucapkan terima kasih kepada teman-teman seperjuangan yang telah memberikan semangat, inspirasi, dan bantuan selama penulisan skripsi. Terima kasih kepada Dr. Ardiansyah, S.T., M.Cs. selaku dosen pembimbing skripsi.

Yogyakarta, 7 Mei 2024

Penulis

ABSTRAK

Penjadwalan praktikum adalah salah satu kegiatan penting menjelang awal semester di semua laboratorium perguruan tinggi. Selama penjadwalan, hal penting yang biasanya harus dicegah adalah terjadinya bentrok antar mata praktikum pada ruangan-ruangan yang akan digunakan dan juga bentrok jadwal praktikum dengan jadwal mengajar dosen dan jadwal kuliah mahasiswa. Selain itu, ada faktor lain yang perlu diperhatikan misalnya ruang laboratorium yang tersedia, jumlah SKS, hingga ketersediaan dosen. Akibatnya, kegiatan penjadwalan praktikum menjadi rumit, kompleks, dan memakan waktu yang tidak sedikit dikarenakan banyaknya faktor dan ketentuan tersebut.

Penelitian ini memanfaatkan algoritma genetika untuk melakukan penjadwalan praktikum yang diimplementasikan pada Laboratorium Informatika UAD. Eksperimen dilakukan dengan menggunakan dua data jadwal praktikum, yaitu semester Genap 2022/2023, dan Gasal 2023/2024 serta data jadwal dosen yang diperoleh dari website SIMERU UAD. Eksperimen ini menjalankan algoritma genetika sebanyak 5 kali untuk setiap dataset dengan jumlah kromosom yang berbeda setiap kali algoritma dijalankan, yaitu 5, 10, 15, 20, 25, 30, iterasi maksimum sebesar 15, dan merancang bangun aplikasi penjadwalan berbasis web.

Hasil eksperimen menunjukkan solusi terbaik berupa nilai fitness 0.0108 dan objektif 92 memiliki 3 kelas praktikum yang bentrok dengan jadwal mengajar dosen dan 1 kelas praktikum yang berbeda hari pada dataset Genap 2022/2023. Sedangkan pada dataset Gasal 2023/2024, hasil menunjukkan solusi terbaik berupa nilai fitness 0.0222 dan objektif 44 memiliki 1 kelas praktikum yang bentrok dengan jadwal mengajar dosen dan 1 kelas praktikum yang berbeda hari. Berdasarkan analisis konvergen menunjukkan bahwa solusi terbaik terdapat pada saat iterasi ke-15 dengan jumlah kromosom sebesar 30. Hasil penelitian ini diharapkan dapat membantu para laboran untuk melakukan penjadwalan praktikum secara efektif dan efisien.

Kata kunci: *penjadwalan; algoritma genetika; praktikum; laboratorium; optimasi metaheuristik*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LISTING

BAB 1.
PENDAHULUAN

1.1 Latar Belakang Masalah

Perguruan Tinggi memiliki peran penting dalam menggerakkan roda pembangunan dan peningkatan kualitas sumber daya manusia di Indonesia. Saat ini, Indonesia telah memiliki ribuan perguruan tinggi yang tersebar di berbagai wilayah, termasuk universitas negeri dan swasta. Menurut data dari Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi, pada tahun 2023, jumlah perguruan tinggi di Indonesia mencapai lebih dari 4.500 institusi. Tak hanya itu, perguruan tinggi juga menjadi rumah bagi jutaan mahasiswa yang berusaha mengasah ilmu dan keterampilan untuk mencapai cita-cita mereka. Berdasarkan data yang sama, jumlah mahasiswa di Indonesia pada tahun 2023 mencapai lebih dari 9 juta orang. Dengan peran Perguruan Tinggi dalam menyediakan pendidikan berkualitas dan membentuk individu yang siap bersaing di era global, Perguruan Tinggi memegang peran kunci dalam membangun masa depan Indonesia.

Berdasarkan perannya menyediakan pendidikan berkualitas, setiap semester, perguruan tinggi harus menyusun jadwal kuliah yang optimal untuk memastikan bahwa semua mata kuliah yang diperlukan oleh mahasiswa dapat diajarkan tanpa tumpang tindih dan bentrok waktu. Penyusunan jadwal kuliah di sini meliputi jadwal mata kuliah dan juga praktikum. Proses ini melibatkan berbagai faktor, seperti ketersediaan dosen yang mengajar, ketersediaan ruang kelas, dan kebijakan kurikulum.

Penjadwalan praktikum adalah kegiatan untuk menentukan waktu dan tempat aktivitas praktikum yang akan dilaksanakan dengan memperhatikan sumber daya yang terbatas. Penyusunan jadwal praktikum mesti dilakukan secara hati-hati, agar tidak terjadi bentrok antar mata kuliah praktikum pada ruangan-ruangan yang akan digunakan. Jika jadwal praktikum disusun baik, maka akan membantu kegiatan perkuliahan berjalan dengan maksimal.

Laboratorium Informatika Universitas Ahmad Dahlan merupakan fasilitas untuk kegiatan perkuliahan dalam bentuk praktikum. Laboratorium biasanya memiliki jadwal praktikum. Membuat jadwal praktikum harus memperhatikan jumlah laboratorium, jadwal kuliah mahasiswa dan dosen.

Penjadwalan praktikum merupakan kegiatan yang rumit, sulit, dan kompleks jika masih dilakukan secara manual sehingga akan membutuhkan waktu yang relatif lama dan sering terjadinya kesalahan dalam proses penyelesainnya [1], [2], [3], [4].

Beberapa penelitian sebelumnya telah dilakukan untuk memecahkan masalah penjadwalan praktikum. Penelitian-penelitian tersebut, menggunakan algoritma optimasi sebagai dasar instruksi-instruksi untuk menyelesaikan masalah-masalah yang ditemui. Dalam penelitian Olipio Sayudas dkk. [1], memanfaatkan algoritma genetika untuk membuat aplikasi penjadwalan laboratorium berbasis website yang memberikan informasi secara otomatis sehingga dapat memudahkan pengguna dalam menentukan jadwal dan meminimalisir kemungkinan adanya tabrakan jadwal antar kelas. Penelitian Muhammad Alda, telah berhasil menggunakan algoritma Shortest Job First Scheduling untuk membuat aplikasi penjadwalan laboratorium berbasis mobile android yang dapat mempermudah pihak SMK Bina Satria Medan dalam mengolah jadwal laboratorium dan menyampaikan informasi jadwal laboratorium serta dengan adanya aplikasi ini dapat mempermudah guru dan murid untuk memperoleh informasi mengenai jadwal penggunaan laboratorium melalui mobile android.

Melakukan penjadwalan praktikum secara manual tentu tidak mudah, karena banyak faktor-faktor yang harus diperhatikan. Faktor-faktor yang dimaksud antara lain mata kuliah praktikum, sks, semester, ruang, hari, dan waktu. Selain rumit, dengan menyusun jadwal praktikum secara manual dapat memunculkan beberapa masalah. Permasalahan yang terjadi adalah dibutuhkan waktu yang relatif lama karena banyak dan kompleksnya data, dan juga berpotensi akan terjadinya kesalahan dalam proses penyusunan.

Algoritma Genetika merupakan salah satu algoritma yang dapat memecahkan permasalahan optimasi dengan konsep mengikuti apa yang terjadi di alam seperti pewarisan, seleksi, mutasi gen, dan crossover. Algoritma Genetika dikemukakan pertama kali oleh John Holland pada 1975 melalui bukunya yang berjudul "Adaption in Natural and Artificial Systems". Saat ini, Algoritma genetika cukup populer dan seringkali digunakan oleh beberapa peneliti untuk dijadikan sebagai dasar metode pembangunan suatu sistem. Pada penelitian ini pun, algoritma yang digunakan adalah algoritma genetika, karena algoritma genetika ini mempunyai beberapa kelebihan, diantaranya yaitu mempunyai ruang pencarian yang sangat luas, waktu eksekusi relatif cepat, dan memiliki kemampuan untuk mendapatkan hasil paling optimal.

Berdasarkan permasalahan yang terjadi, penelitian ini dilakukan untuk menciptakan sebuah sistem yang bisa digunakan agar proses penyusunan jadwal praktikum pada Laboratorium Informatika menjadi lebih mudah dan cepat serta meminimalisir kesalahan yang terjadi sehingga bisa mendapatkan hasil yang maksimal dan efektif. Terdapat banyak cara membuat sebuah sistem yang dapat membantu untuk memecahkan permasalahan di atas. Namun pada penelitian ini, sistem yang akan dibuat merupakan teknik optimasi yang menggunakan algoritma genetika untuk mencari solusi terbaik dalam penyusunan jadwal-jadwal mata kuliah praktikum pada ruang-ruang yang tersedia serta tidak terjadi adanya dua atau lebih jadwal mata kuliah praktikum di satu ruang pada hari dan waktu yang sama.

1.2 Rumusan Masalah

Berdasarkan latar belakang masalah, didapatkan rumusan masalah, yaitu bagaimana meningkatkan efisiensi penjadwalan praktikum dengan menerapkan algoritma genetika.

1.3 Tujuan Penelitian

Berdasarkan rumusan masalah, maka tujuan penelitian ini adalah sebagai berikut:

1. Mengembangkan aplikasi penjadwalan praktikum.

2. Membuat program yang menerapkan algoritma genetika pada aplikasi penjadwalan praktikum.
3. Menerapkan aplikasi penjadwalan praktikum pada Laboratorium Informatika UAD.
4. Mengukur tingkat efisiensi sebelum dan sesudah aplikasi penjadwalan praktikum diterapkan.

1.4 Manfaat Penelitian

Manfaat yang diharapkan dari penelitian ini adalah membantu laboran agar lebih meningkatkan efisiensi dari segi waktu maupun resiko terjadinya kesalahan dalam menyusun jadwal praktikum berdasarkan jadwal kuliah dan jumlah laboratorium yang tersedia.

BAB 2.
TINJAUAN PUSTAKA

2.1 Kajian Penelitian Terdahulu

Berdasarkan penelitian yang dilakukan oleh [4] didapatkan permasalahan, yaitu penyusunan jadwal asisten dosen di STIKOM Bali masih dilakukan secara tradisional sehingga waktu yang dibutuhkan akan sangat banyak karena besarnya jumlah asisten dosen dan jumlah matakuliah yang bisa diampu terbatas serta kombinasi jenis mata kuliah yang dapat diampu dengan menyesuaikan jam mengajar asisten dosen. Algoritma yang digunakan adalah Algoritma Genetika. Penerapan algoritma ini dapat mendukung penyusunan jadwal mengajar asisten pengajar STIKOM di Bali menjadi lebih optimal, serta dapat mengatasi berbagai kendala yang dihadapi dalam penyusunan jadwal asisten pengajar. Hasilnya adalah Algoritma Genetika mampu melaksanakan penyusunan jadwal asisten dosen di STIKOM Bali dengan otomatis sehingga bisa mengeluarkan hasil jadwal mengajar asisten dosen yang efisien.

Kemudian, penelitian yang dilakukan oleh [5] dilatarbelakangi masalah yaitu sistem penjadwalan laboratorium di SMK Bina Satria yang terletak di Kota Medan masih dilakukan secara konvensional dengan cara jadwal penggunaan laboratorium dicatat pada sebuah buku kemudian disosialisasikan kepada masing-masing guru yang menyebabkan guru membutuhkan waktu dan tenaga lebih dalam menerima informasi mengenai jadwal laboratorium, serta admin juga kesulitan dalam mencocokkan jadwal laboratorium dan mengolah penggunaan jadwal laboratorium jika ada perubahan oleh guru. Aplikasi berbasis android dibangun sebagai solusi dari permasalahan yang ada dengan memanfaatkan algoritma Shortest Job First Scheduling. Penelitian dilakukan dengan metode XP (Extreme Programming) yang dapat membangun aplikasi berjalan baik berdasarkan kebutuhan pengguna menggunakan Algoritma Shortest Job First Scheduling yang memiliki latensi lebih cepat dalam proses penjadwalan laboratorium sehingga memudahkan pihak SMP Bina Satria Medan dalam mengolah jadwal laboratorium dan menyampaikan

informasi jadwal laboratorium. Selain itu, aplikasi ini juga bisa membantu guru dan murid untuk mendapatkan informasi tentang jadwal penggunaan laboratorium melalui mobile android.

Penelitian selanjutnya dilakukan oleh [2], didasari oleh permasalahan adanya beberapa kendala dalam proses penjadwalan praktikum di Teknologi Informasi Universitas Semarang karena pembuatan jadwal masih menggunakan pengolahan data dengan Microsoft Excel. Dengan mengimplementasikan Algoritma Genetika, sebuah sistem penjadwalan praktikum secara otomatis dapat dibangun guna membantu pihak program studi dalam menyusun penjadwalan praktikum dengan cepat dan mudah. Data jadwal kuliah praktikum dapat diwakilkan ke dalam individu-individu untuk mengeluarkan jadwal praktikum yang efisien dengan melihat konstrain yang tersedia.

Berikutnya, penelitian [6] berupa sistem informasi berbasis CodeIgniter menggunakan Algoritma Genetika. Dilatarbelakangi oleh Laboratorium Teknik Elektro Universitas Muhammadiyah Malang yang sistem pengelolaannya masih menggunakan perangkat lunak Microsoft Office membuatnya masih tidak efisien, terutama dalam hal manajemen data dan penyimpanan data. Terdapat beberapa kelemahan dalam penggunaan sistem pemberkasan tersebut, yaitu sistem tersebut kurang efisien dan pengolahan nilai tetap tidak praktis ketika mahasiswa ingin memeriksa kembali status kelulusan dan nilai praktikumnya. Oleh karena itu, diperlukan suatu sistem informasi yang dapat memfasilitasi kinerja laboratorium dengan menangani jadwal praktikum, membantu proses pelaporan, dan membantu menyampaikan informasi secara cepat dan akurat. Hasilnya, Algoritma Genetika dapat menciptakan jadwal praktikum secara efisien dengan waktu eksekusi rata-rata 300 detik per generate dan mengambil kromosom yang paling optimum dari jumlah generasi yang ada melalui proses inisialisasi, seleksi, crossover, dan mutasi.

Penelitian terakhir dilakukan oleh [3], permasalahan yang ada, yaitu proses penyusunan jadwal masih dilaksanakan oleh admin laboratorium dengan mengumpulkan data-data yang dibutuhkan lalu

melaksanakan kombinasi untuk membuat jadwal yang tepat dengan pelaksanaan yang kadang kala terjadi kesalahan-kesalahan manusia dan proses yang membutuhkan waktu relatif lama karena inputan data banyak dan parameter yang kompleks. Pembuatan jadwal secara sistem dengan melakukan pendekatan Algoritma Genetika menghasilkan jadwal yang terdiri atas beberapa proses diantaranya memasukkan data, pembangkitan populasi awal, evaluasi, seleksi, crossover, dan mutasi. Dengan demikian, mewujudkan jadwal dengan memperhatikan 8 jenis aturan yaitu jadwal tidak bentrokan dengan kelas lain, menggunakan ruangan kosong / tersedia, pengajar tidak mengajar dalam waktu yang sama, mata kuliah laboratorium mendapatkan 2 jenis ruangan, mata kuliah lab mendapatkan waktu operasional lab, semua mata kuliah diajarkan dan sesuai dengan dosen, semua kelas mendapatkan semua mata kuliah dan sesuai dengan semeste

Tabel 2.1 Kajian Penelitian Terdahulu

Peneliti (Sitasi) *	Dataset	Variabel	Metode	Hasil*
I Made Budi Adnyana	data asisten dosen, kesediaan jam mengajar asdos, jadwal perkuliahan, ruangan, dan waktu	Jadwal Asdos	Algoritma Genetika	Penerapa algoritma Genetika pada Sistem Penjadwalan Asdos STIKOM Bali dapat menghasilkan jadwal asdos secara otomatis dan optimal.
Muhama Alda	Data Laboratorium SMK Bina Satria, Data Guru SMK Bina Satria Data Mata Pelajaran Praktek SMK Bina Satria dan Data Jadwal Praktek SMK Bina Satria	Waktu dan tenaga guru	Shortest Job First Scheduling	Menghasilkan aplikasi penjadwalan laboratorium berbasis mobile android.
Bernadus Very Christioko, Siti Asmiatun, Susanto	Data Dosen, Data Matakuliah, Data Hari, Data Jam, Data Ruang, Data Pengampu	pengampu, mata kuliah, dan dosen	Algoritma Genetika	Data jadwal kuliah praktikum yang terdiri dari data dosen, matakuliah, hari, jam, ruang dan pengampu dapat direpresentasikan ke dalam kromosom-kromosom untuk menghasilkan jadwal praktikum yang optimal dengan memperhatikan konstrain yang ada.
Yoga Nur Firmansyah, Nur Kasan, Merinda Lestandy	data hari, waktu, ruang, mata praktikum dan jadwal asisten.	Jumlah Populasi, Jumlah Generasi, Probabilitas Crossover, Probabilitas Mutasi, Lama Praktikum, Tanggal Mulai, Mata Praktikum	Algoritma Genetika	Menghasilkan jadwal praktikum secara optimal dengan rata-rata waktu eksekusi 300 detik per generate, mengutamakan tidak terjadinya bentrok jadwal.
Muh Syawala, Poetri Lestari Lokapitasaria, Abdul Rachman Manga	data matakuliah, data dosen, data kelas mahasiswa, data ruangan, data waktu dan data asisten	mata kuliah dan kelas	Algoritma Genetika	Berhasil melakukan penjadwalan dengan memperhatikan 8 rule

Berdasarkan kajian penelitian terdahulu yang telah dipaparkan di atas, dapat disimpulkan bahwa penelitian-penelitian tersebut dilatarbelakangi permasalahan yang hampir sama satu sama lain yaitu sistem yang mereka miliki masih dilaksanakan secara manual dan algoritma yang digunakan adalah Algoritma Genetika dan Shortest Job First Scheduling. Pada penelitian ini akan membuat aplikasi penjadwalan praktikum menggunakan Algoritma Genetika. Sedangkan perbedaan dengan penelitian yang akan dilakukan yaitu pemanfaatan Algoritma Genetika untuk melakukan penjadwalan praktikum dengan tempat yang spesifik di Laboratorium Informatika Universitas Ahmad Dahlan.

2.2 Landasan Teori

2.2.1 Penjadwalan

Menurut [7] penjadwalan adalah kegiatan yang dilakukan untuk mengalokasikan fasilitas, peralatan, maupun tenaga kerja, dan menentukan urutan pelaksanaan bagi suatu kegiatan operasi. Penjadwalan sering digunakan untuk mengatur suatu kegiatan, seperti jadwal kuliah, jadwal kelas di sekolah, jadwal laboratorium, jadwal kerja, dan lain sebagainya. Untuk melaksanakan penjadwalan tentu sangat sulit jika elemen-elemen penyusunannya dalam jumlah besar dan juga kompleks.

Tujuan penjadwalan, yaitu untuk membuat suatu kegiatan dapat berjalan dengan lancar dan efisien dengan memanfaatkan sumber daya yang ada namun terbatas. Selain itu, penjadwalan juga memiliki tujuan untuk meminimalkan waktu yang dibutuhkan atau mengurangi jumlah tugas dilihat dari situasi dan kondisi yang ada. Penjadwalan juga bisa membuat suatu kegiatan untuk tidak bertabrakan dengan kegiatan yang lainnya. Dengan memanfaatkan penjadwalan, produktivitas dari sumber daya yang ada juga akan meningkat dan akan mengurangi waktu sumber daya untuk menganggur.

Pada dasarnya, penjadwalan adalah kegiatan untuk menentukan waktu dan tempat suatu aktivitas yang akan dilaksanakan dengan memperhatikan sumber daya yang terbatas. Penjadwalan sangat dibutuhkan karena dengan adanya penjadwalan, aktivitas yang ingin dilaksanakan berpeluang akan

berjalan dengan baik dan lancar. Dengan demikian, penjadwalan dapat diharapkan sebagai alat ukur untuk perencanaan terlaksananya suatu aktivitas.

2.2.2 Algoritma Genetika

Pada awalnya, Algoritma Genetika dikemukakan pertama kali oleh John Holland pada tahun 1975 melalui bukunya yang berjudul “Adaption in Natural and Artificial Systems”. Kemudian pada tahun 1980-an, seorang ilmuan bernama David Goldberg yang merupakan murid dari John Holland, berhasil mengaplikasikan Algoritma Genetika melalui perancangan sistem perpipaan untuk distribusi gas alam [8]. Goldberg menilai perancangan sistem tersebut sangat rumit untuk diselesaikan. Namun, dengan memanfaatkan Algoritma Genetika, ia mampu memecahkan permasalahan tersebut. Algoritma Genetika adalah algoritma yang dapat memecahkan permasalahan optimasi dengan konsep seperti evolusi biologis alam untuk menentukan individu terbaik dalam suatu populasi. Sebagai cabang dari algoritma evolusi, Algoritma Genetika merupakan metode adaptif yang sering digunakan untuk menyelesaikan masalah pencarian nilai dalam masalah optimasi.

Algoritma genetika menggunakan analogi langsung dengan kebiasaan alam, yaitu seleksi alam. Algoritma ini bekerja pada kelompok individu atau kromosom. Masing-masing mewakili kemungkinan solusi untuk masalah yang dihadapi. Dalam hal ini, individu atau kromosom diwakili oleh nilai fitness yang akan digunakan untuk mencari solusi terbaik dari suatu masalah yang ada.

Algoritma Genetika membentuk individu-individu sebagai calon solusi yang akan digunakan untuk menyelesaikan suatu masalah. Kumpulan dari individu-individu ini disebut sebagai populasi. Individu-individu ini sendiri tersusun dari gen yang bernilai bilangan numerik, biner, ataupun karakter tergantung permasalahan yang akan diselesaikan. Individu-individu akan terus berevolusi secara terus-menerus yang disebut sebagai generasi. Dalam tiap generasi kromosom, akan dilakukan proses evaluasi yang akan mengukur tingkat keberhasilan nilai solusi dari masalah yang ingin diselesaikan ataupun bisa disebut juga

dengan nilai fitness. Kemudian, akan dilakukan proses seleksi dimana kromosom-kromosom yang memiliki nilai fitness yang tinggi mendapatkan peluang lebih besar akan terpilih lagi pada generasi berikutnya. Setelah itu, proses crossover akan dilakukan untuk membentuk individu-individu baru yang bisa disebut dengan offspring, dengan cara melakukan perkawinan antar individu-individu dalam satu generasi. Jumlah kromosom dalam populasi yang telah melalui proses crossover, ditentukan oleh parameter yang dinamakan crossover rate.

Mekanisme perubahan komposisi unsur penyusun suatu organisme karena faktor alam disebut mutasi, yang diwujudkan dengan adanya proses perubahan satu atau lebih nilai gen dalam kromosom dengan nilai acak. Total gen pada populasi telah melalui proses mutasi, ditentukan oleh parameter yang biasa disebut dengan mutation rate. Setelah memperoleh beberapa generasi, akan didapatkan kromosom-kromosom yang nilai gen-gennya memusat ke suatu nilai tertentu yang akan digunakan sebagai solusi terbaik algoritma genetika terhadap permasalahan optimasi yang butuh diselesaikan.

2.2.2.1 Alur Algoritma Genetika

Langkah-langkah dalam algoritma genetika dapat dijabarkan sebagai berikut:

1. Pembangkitan kromosom atau individu

   Kromosom dibentuk sebagai solusi awal dari algoritma genetika yang tersusun dari sekumpulan gen yang bernilai bilangan numerik, biner, ataupun karakter. Berikut ini merupakan contoh dari kromosom.

0	1	2	3	4	5	6	7	8	9

2. Pembangkitan populasi

   Didapatkan secara acak berdasarkan kumpulan-kumpulan kromosom yang telah dibangkitkan pada tahapan sebelumnya. Berikut ini merupakan contoh dari populasi.

0	1	2	3	4	5	6	7	8	9

Kromosom 1

4	5	6	7	8	9	10	11	12	13

Kromosom 2

8	9	10	11	12	13	0	1	2	3

Kromosom 3

14	15	16	17	18	19	20	21	22	23

Kromosom 4

3. Evaluasi berdasarkan fungsi fitness

   Fungsi fitness dari suatu kromosom akan dihitung agar dapat diketahui baik tidaknya suatu solusi berdasarkan nilai fitnessnya. Semakin besar nilai fitness, maka semakin baik juga solusi dari kromosom tersebut. Fungsi fitness dihitung dengan menggunakan persamaan sebagai berikut.

$$\frac{1}{(1 + Nilai\ Objektif)}$$

4. Melakukan seleksi

   Kromosom-kromosom diseleksi dengan berbagai macam teknik seleksi untuk medapatkan calon induk atau parents yang akan digunakan sebagai generasi berikutnya. Roulette wheel yaitu teknik memilih suatu parents berdasarkan kromosom-kromosom yang diwakilkan oleh presentase fitness dari kromosom-kromosom tersebut. Setiap kromosom memiliki luas bagian sesuai dengan presentase fitnessnya. Kromosom yang memiliki luas bagian lebih besar akan mendapatkan lebih banyak peluang untuk terpilih. Selain Roulette Wheel, ada suatu metode bernama Baker's SUS (Stochastic Universal Sampling) yang memperbanyak jumlah jarum pada Roulette Wheel. Kedua metode ini termasuk ke dalam Fitness Proportionate Selection (FPS).

Selain itu, ada metode seleksi elitism yang memilih calon induk berdasarkan nilai fitness tertinggi. Selanjutnya, metode Rank-Based Selection dimana seleksi dilakukan dengan perankingan yang memanfaatkan pendekatan selective pressure (probabilitas terpilih individu terbaik dibandingkan dengan rata-rata probabilitas terpilih semua individu). Binary Tournament, merupakan teknik seleksi memanfaatkan pendekatan sampling dengan mengambil sejumlah kromosom secara acak dari populasi. Dari sejumlah kromosom tersebut, akan dipilih satu kromosom sebagai parents berdasarkan cara atau prosedur tertentu.

5. Melakukan cross over

   Melibatkan dua parents yang didapat berdasarkan proses seleksi sebelumnya dengan cara melakukan kawin silang dalam satu generasi secara acak untuk membentuk kromosom-kromosom baru atau offspring.

0	**1**	**2**	**3**	**4**	**5**	6	7	8	9

Induk 1

14	15	16	17	18	19	**20**	**21**	**22**	**23**

Induk 2

0	**1**	**2**	**3**	**4**	**5**	**20**	**21**	**22**	**23**

Kromosom baru (offspring)

6. Melakukan mutasi

   Dilakukan penggeseran nilai ataupun menginversi nilai gen dari suatu kromosom, misal pada awalnya gen bernilai 1 menjadi 0 (biner), nilai acak sesuai interval (real).

14	15	16	17	18	19	20	21	**22**	23

Kromosom terpilih untuk dimutasi

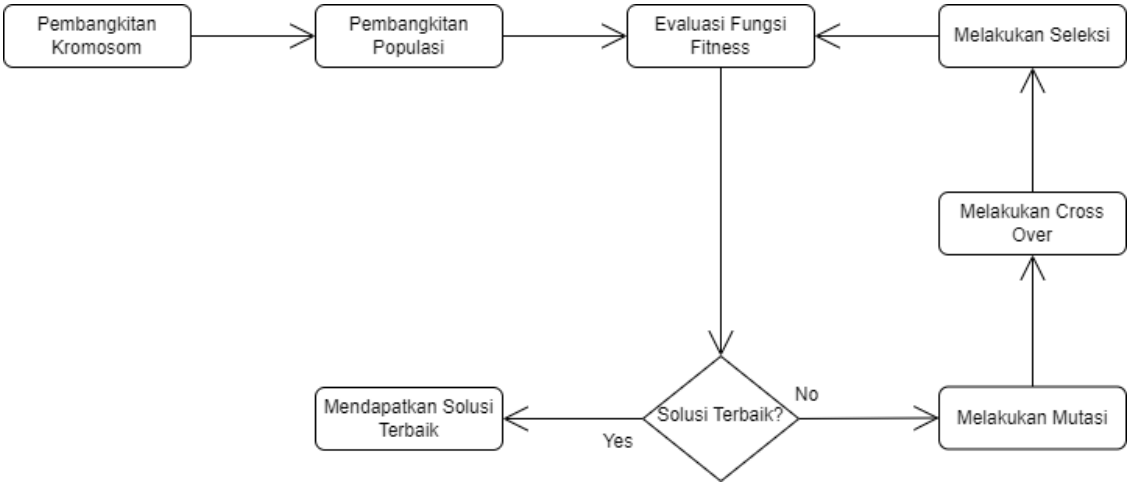
14	15	16	17	18	19	20	21	**5**	23

Kromosom setelah melalui proses mutasi

7. Penentuan solusi terbaik

   Pada langkah ini, keputusan akan diambil antara mendapatkan solusi terbaik atau belum. Jika belum mendapatkan solusi terbaik, maka akan kembali ke proses evaluasi fitness hingga ke proses mutasi sampai ada saatnya solusi terbaik dihasilkan.

Gambar 2.1 menunjukkan alur proses yang terjadi pada Algoritma Genetika.



Gambar 2.1 Alur Proses Algoritma Genetika

2.2.3 Friedman Test

Friedman Test adalah uji statistik non-parametrik yang digunakan untuk menganalisis data yang berasal dari desain eksperimen dengan lebih dari dua kondisi atau perlakuan yang bersifat berulang (repeated measures). Uji ini merupakan alternatif dari analisis varians satu arah (ANOVA) untuk data yang tidak memenuhi asumsi normalitas. Friedman Test digunakan untuk menentukan apakah ada perbedaan yang signifikan antara kelompok-kelompok yang dibandingkan[9].

Asumsi-asumsi Friedman Test:

1. Data Berulang (Repeated Measures): Subjek yang sama diukur lebih dari sekali di bawah kondisi yang berbeda.

2. Skala Pengukuran: Data harus dalam skala ordinal atau interval/rasio.
3. Randomisasi: Observasi yang diambil harus secara random.

Prosedur Friedman Test:

1. Rangking Data: Untuk setiap subjek, data diurutkan dan diberikan rangking dari yang terkecil hingga yang terbesar.
2. Menghitung Rata-rata Rangking: Menghitung rata-rata rangking untuk setiap kondisi.
3. Menghitung Statistik Uji: Statistik uji Friedman dihitung menggunakan formula:

   $x_r^2 = \frac{12}{n(k(k+1))} \left[\sum R_i^2\right] - 3n(k+1)$, di mana

   x_r^2 = nilai uji statistik friedman

   n = jumlah perulangan

   k = banyaknya perlakuan

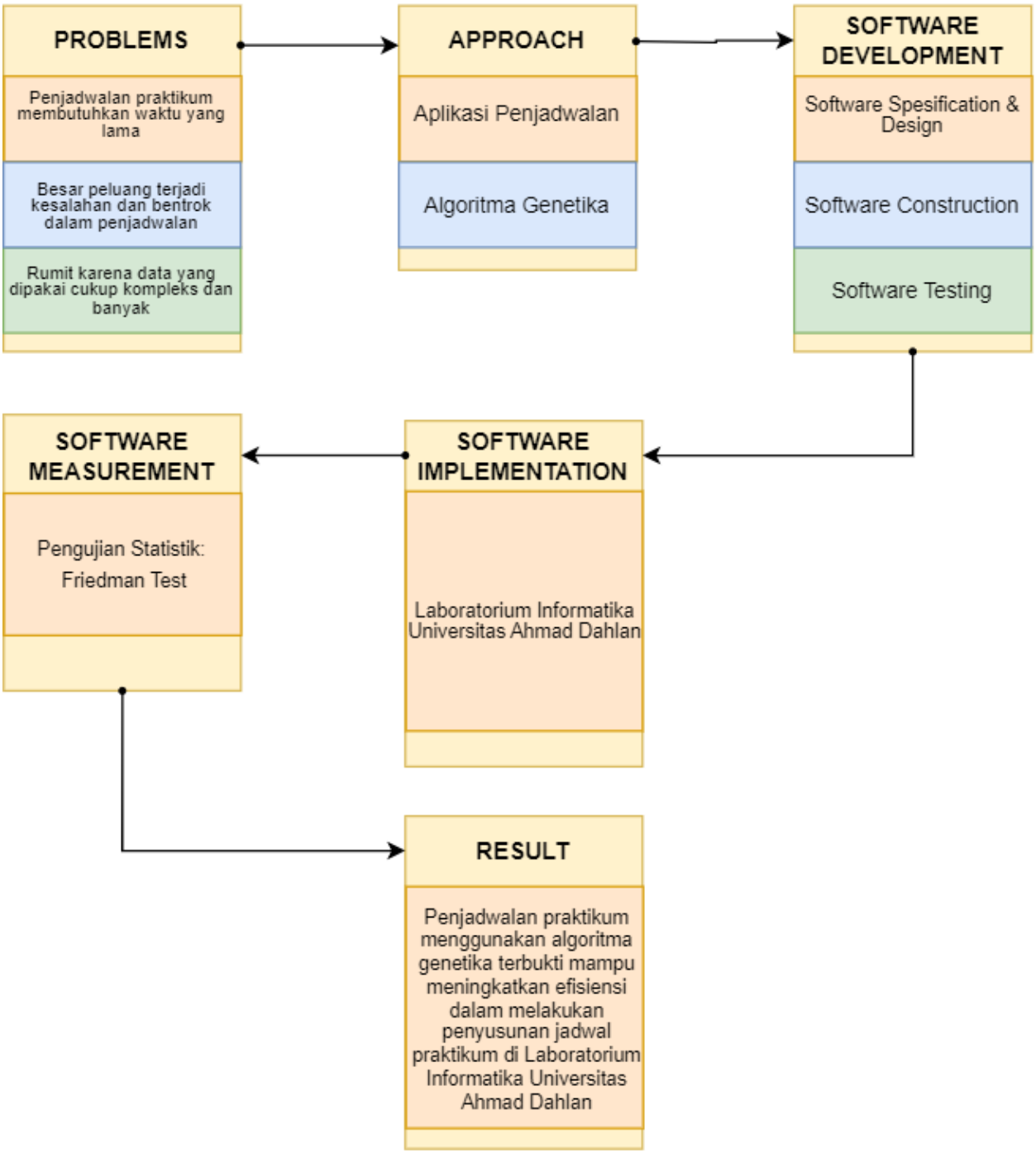
   $\sum R_i^2 =$ jumlah pemeringkatan

4. Menentukan Nilai p: Nilai statistik uji dibandingkan dengan distribusi chi-square untuk menentukan signifikansi.

BAB 3.
METODE PENELITIAN

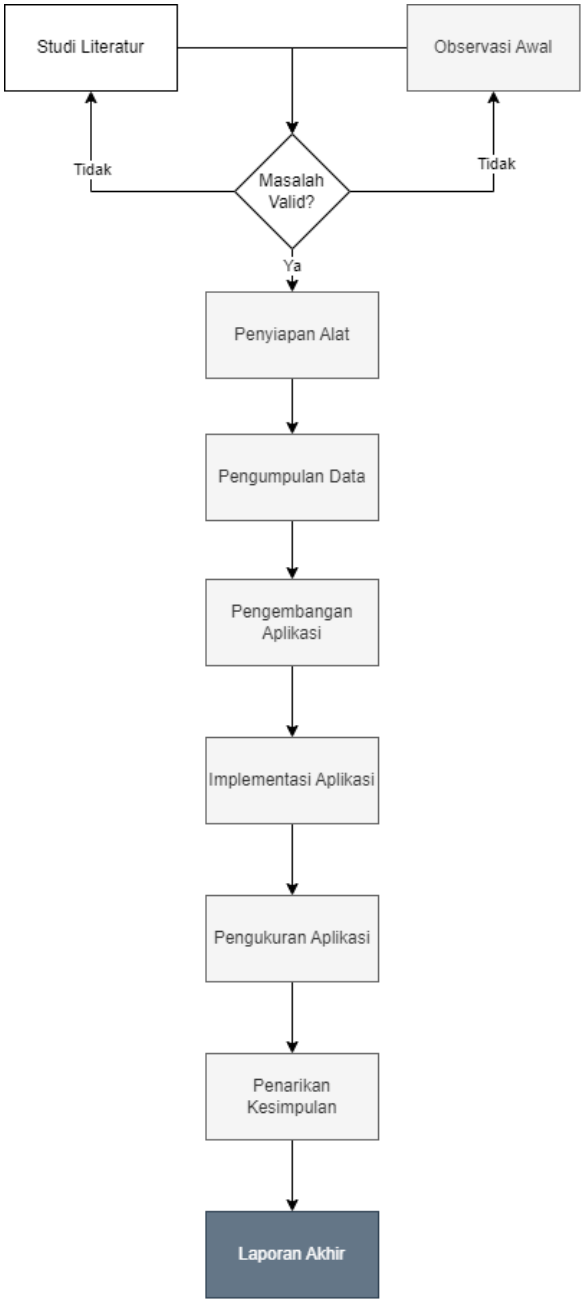
3.1 Kerangka Pemikiran Penelitian

Berdasarkan Gambar 3.1, dapat dijabarkan bahwa Problems, berisi masalah-masalah yang terjadi pada penelitian ini. Masalah-masalah tersebut terdiri dari penjadwalan praktikum membutuhkan waktu yang lama, besar peluang terjadi kesalahan dan bentrok dalam penjadwalan, dan Rumit karena data yang dipakai cukup kompleks dan banyak. Approach, yaitu teori pendekatan yang dimanfaatkan dalam memecahkan masalah yang ada pada bagian Problems. Pada penelitian ini menggunakan pendekatan penjadwalan aplikasi, dan Algoritma Genetika. Software Development, yaitu pengembangan software yang didasari oleh teori dan Approach yang telah dipilih sebelumnya. Software Spesification & Design, Software Construction, dan Software Testing merupakan tahap-tahap yang akan dilakukan pada bagian ini. Software Implementation, yaitu target dimana software akan diterapkan. Target penerapan di penelitian ini adalah Laboratorium Informatika Universitas Ahmad Dahlan. Software Measurement, merupakan tahap pengukuran seberapa berguna software ini melalui pengujian statistik friedman test. Result, merupakan kesimpulan yang didapat berdasarkan seluruh proses penelitian dan pengukuran yang telah dilakukan. Result pada penelitian ini adalah penjadwalan praktikum menggunakan algoritma genetika terbukti mampu meningkatkan efisiensi dalam melakukan penyusunan jadwal praktikum di Laboratorium Informatika Universitas Ahmad Dahlan.



Gambar 3.1 Kerangka Pemikiran Penelitian

3.2 Tahapan Penelitian



Gambar 3.2 Diagram Tahapan Penilitan

Pada tahapan penelitian, akan dijelaskan fase-fase yang akan ditempuh selama proses penelitian.

3.2.1 Observasi Awal

Berawal dari salah satu Dosen di Universitas Ahmad Dahlan memiliki ide penelitian dengan topik berkaitan dengan masalah penjadwalan praktikum di Laboratorium Informatika UAD. Dosen tersebut

menawarkan topik ini kepada mahasiswa bimbingannya, yang kemudian topik ini sepakat untuk diambil oleh penulis. Kemudian, penulis melakukan wawancara dengan salah satu Narasumber yang bertugas selama kurang lebih selama 15 tahun sebagai Laboran di Laboratorium Informatika UAD untuk mengetahui bagaimana proses penjadwalan praktikum yang terjadi di Lab tersebut. Berdasarkan informasi yang didapat dari Narasumber, diketahui bahwa penjadwalan praktikum masih dilakukan secara manual menggunakan media excel. Narasumber juga menjelaskan bahwa melakukan penjadwalan relative sulit tergantung dengan kondisi dosen dan jumlah slot yang didasari oleh jumlah mahasiswa dan mengalami beberapa kendala yang dialami selama melakukan penjadwalan praktikum di Lab tersebut antara lain, yaitu durasi dan waktu mulai antara pelaksanaan praktikum dan kuliah di kelas berbeda yang membuat beberapa dosen jadwalnya menjadi tanggung jika dimasukkan ke dalam jadwal praktikum pada jam-jam tertentu. Misal, jam 9 dimasukkan dosen A namun ia ada mengajar di kelas jam 9.25, jika dimasukkan dosen B, ada mengajar sampai jam 10, dan lain sebagainya. Selain itu, ada beberapa dosen yang tidak menerima jika terkena jadwal pagi, sore, sabtu, ataupun malam. Menurut Narasumber, jika ada aplikasi atau program penjadwalan praktikum, paling tidak proses penjadwalan bisa termudahkan beberapa persen daripada dilakukan secara manual sepenuhnya.

3.2.2 Studi Literatur

Studi literatur merupakan fase dimana mengumpulkan referensi-referensi yang dibutuhkan untuk mendukung penelitian ini. Aktivitas ini sangat penting karena bermanfaat untuk mengidentifikasi masalah dan melakukan perbandingan antara metode-metode yang pernah digunakan terkait penelitian. Referensi-referensi yang dikumpulkan berupa jurnal/artikel ilmiah, buku referensi dan lain sebagainya. Penelusuran referensi dalam bentuk jurnal memanfaatkan mesin pencari seperti Google Scholar dengan kata kunci antara lain yaitu: “penjadwalan praktikum”, “algoritma genetika”, dan lain sebagainya. Hasil penelusuran berupa berkas berformat PDF. Berkas ini disimpan dan dikelola menggunakan perangkat lunak bibliografi Mendeley.

3.2.3 Penyiapan Alat

Dibutuhkan beberapa peralatan untuk membantu agar penelitian ini dapat berjalan. Peralatan tersebut mencakup kebutuhan perangkat keras (hardware) dan juga perangkat lunak (software). Berikut ini rincian dari hardware dan software yang akan digunakan:

- Perangkat Keras (Hardware)

  Perangkat keras atau hardware yang akan digunakan dalam penelitian ini antara lain:

  a. Laptop Acer Nitro 5 AN515-54 dengan spesifikasi sebagai berikut:
     1. Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz
     2. Intel(R) UHD Graphics 630
     3. NVIDIA GeForce GTX 1650
     4. RAM 16 GB
  b. Keyboard
  c. Mouse

- Perangkat Lunak (Software)

  Perangkat lunak atau software yang akan digunakan dalam penelitian ini antara lain:

  a. Windows 11 64 bit
  b. Google Chrome
  c. Visual Studio Code
  d. Python
  e. HTML, CSS
  f. Bootstrap
  g. XAMPP

3.2.4 Pengumpulan Data

Pengumpulan data diperlukan agar mendapat data yang akan digunakan untuk penelitian. Penelitian ini menggunakan data-data dokumen jadwal praktikum dari Laboratorium Informatika UAD yang relevan dengan tujuan penelitan. Metode pengumpulan data ini menggunakan metode wawancara dan dokumen.

3.2.5 Pengembangan Aplikasi

Setelah memperoleh alat dan data penelitian yang dibutuhkan, selanjutnya akan dilakukan pengembangan Software Specification & Design, Software Construction (coding), dan Software Testing yang menggunakan pendekatan algoritma genetika agar dapat menghasilkan penjadwalan praktikum yang optimal dan efisien. Pembuatan program menggunakan bahasa pemrograman Python dengan mengikuti algoritma genetika serta memanfaatkan tools-tools dalam pembuatan front-end antara lain, yaitu HTML, CSS, dan Bootstrap.

3.2.6 Implementasi Aplikasi

Aplikasi yang telah dibuat kemudian akan diimplementasikan pada target obyek dimana aplikasi ini akan diterapkan. Target pada penelitian ini adalah Laboratorium Informatika UAD.

3.2.7 Pengukuran Aplikasi

Pengukuran aplikasi dilakukan guna mengetahui seberapa besar peningkatan yang dihasilkan aplikasi dalam mencapai tujuan penelitian ini. Pengukuran ini menggunakan pengujian statistik friedman test.

3.2.8 Penarikan Kesimpulan

Setelah mendapatkan hasil pengukuran akan dilakukan penarikan kesimpulan atas penelitian yang telah dilakukan. Kesimpulan dibuat berdasarkan hasil dari pengembangan, implementasi, dan juga pengukuran aplikasi.

BAB 4.
HASIL DAN PEMBAHASAN

4.1 Hasil Pengumpulan Data

Data yang berhasil dikumpulkan adalah data ketersediaan slot laboratorium, daftar praktikum, jadwal mengajar dosen, jam mengajar dosen, dan juga jam kerja laboratorium. Data ketersediaan slot laboratorium diambil berdasarkan hari dan jam dari laboratorium yang tersedia seperti yang disajikan oleh Tabel 4.1. Data daftar praktikum dan jam kerja laboratorium diambil dari sampel jadwal praktikum di Laboratorium Informatika pada satu semester seperti yang disajikan oleh Tabel 4.1. Jadwal mengajar dan jam mengajar dosen diambil dari website milik Simeru Universitas Ahmad Dahlan seperti yang disajikan oleh Tabel 4.2.

Tabel 4.1 Contoh Jadwal Praktikum Laboratorium Informatika

NO	WAKTU	SENIN	SELASA	RABU	KAMIS	JUMAT	SABTU
1	07.30 - 09.00	Matematika Diskret Nur Rochmah DPA, S.T., M.Kom.	APPL Drs., Tedy Setiadi, M.T.	APPL Faisal Fajri R., S.Si., M.Cs.	Matematika Diskret Lisna Zahrotun, S.T., M.Cs.		Matematika Diskret Dewi Soyusiowaty, S.T., M.T.
2	09.00 - 10.30	Matematika Diskret Miftahurrahma R., S.Kom., M.Eng.	APPL Mushlihudin, S.T., M.T.	APPL Faisal Fajri R., S.Si., M.Cs.	Matematika Diskret Lisna Zahrotun, S.T., M.Cs.	APPL Arfiani NK, S.T., M.Kom.	Matematika Diskret Dewi Soyusiowaty, S.T., M.T.
3	10.30 - 12.00	APPL	Matematika Diskret	APPL	Matematika Diskret		Kriptografi

		Ali Tarmuji., S.T., M.Cs.	Lisna Zahrotun, S.T., M.Cs.	Faisal Fajri R., S.Si., M.Cs.	Lisna Zahrotun, S.T., M.Cs.		Eko Aribowo,S.T., M.Kom.
4	12.00 - 13.30	APPL Arfiani NK, S.T., M.Kom.	Matematika Diskret Nur Rochmah DPA, S.T., M.Kom.	Kriptografi Nur Rochmah DPA, S.T., M.Kom.	APPL Arfiani NK, S.T., M.Kom.		Matematika Diskret Nur Rochmah DPA, S.T., M.Kom.
5	13.30 - 15.00						
6	15.00 - 16.30						
7	16.30 - 18.00						

Tabel 4.2 Contoh Jadwal Mengajar Dosen

Hari	Kode	Mata Kuliah	Kelas	SKS	Jam	Semester	Dosen	Ruang
Senin	211810531	Logika Informatika	A	3	1,2	I	Nur Rochmah Dyah PA, S.T., M.Kom.	4.1.5.57

	211810531	Logika Informatika	I	3	1,2	I	Herman Yuliansyah, S.T., M.Eng., Ph.D.	4.1.5.55
	211830341	Basis Data	C	4	1,2,3	III	Miftahurrahma Rosyda, S.Kom., M.Eng.	4.1.5.59
	211830641	Statistika Informatika	D	4	1,2,3	III	Ir., Sri Winiarti, S.T., M.Cs.	4.1.5.69
	211850831	Penambangan Data/Data Mining	C	3	1,2	V	Drs., Tedy Setiadi, M.T.	4.1.5.60
	211870320	Kewirausahaan	A	2	1,2	VII	Taufiq Ismail, S.T., M.Cs.	Daring (T. Informatika)
	211810720	Pancasila	C	2	3,4	I	Ilham Yuli Isdiyanto, S.H., M.H.	Daring (T. Informatika)

Data ketersediaan slot laboratorium (hari, kode jam, lab) yang telah didapat dari Tabel 4.1 akan direpresentasikan ke dalam bentuk nomor indeks 0-199. Nomor indeks ini yang nantinya akan digunakan untuk mewakili nilai-nilai dari setiap gen pada suatu kromosom. Representasi data ketersediaan slot laboratorium disajikan pada Tabel 4.3.

Tabel 4.3 Representasi Data Slot Laboratorium

No. Indeks	Hari, Kode Jam, Lab
0	['Senin', 0, 'Jarkom']
1	['Senin', 1, 'Jarkom']
2	['Senin', 2, 'Jarkom']
3	['Senin', 3, 'Jarkom']
4	['Senin', 4, 'Jarkom']
5	['Senin', 5, 'Jarkom']
6	['Senin', 6, 'Jarkom']
7	[Selasa', 0, 'Jarkom']
8	[Selasa', 1, 'Jarkom']
…	…
198	['Sabtu', 5, 'Riset']
199	['Sabtu', 6, 'Riset']

4.2 Parameter

Dalam Algoritma Genetika, terdapat parameter yang mengatur kinerja algoritma sehingga akan memengaruhi hasil pencarian solusi. Pada penelitian ini, algoritma genetika membutuhkan beberapa parameter, yaitu numOfChromosome, crossoverRate, mutationRate, maxGen, differentDay, dan teachingScheduleConflict. Makna tiap parameter adalah sebagai berikut dan ringkasannya disajikan pada tabel 4.3.

a. numOfChromosome, merupakan jumlah kromosom atau individu yang terdapat dalam populasi yang akan dibangkitkan. Jumlah ini memengaruhi seberapa besar variasi genetik yang dapat dieksplorasi dalam pencarian solusi oleh algoritma genetika. Semakin besar

numOfChromosome, semakin besar populasi yang dievaluasi, yang dapat meningkatkan kemungkinan menemukan solusi yang optimal atau mendekati optimal dalam ruang pencarian. Pada penelitian ini ditentukan numOfChromosome sebesar 30, karena menghasilkan solusi paling optimal berdasarkan eksperimen yang dilakukan.

b. crossoverRate, merupakan nilai yang akan menentukan probabilitas atau tingkat di mana crossover gen dari dua parents dilakukan untuk menghasilkan kromosom baru. Nilai crossoverRate biasanya dipilih secara acak antara 0 dan 1. Pada penelitian ini, ditentukan nilai crossoverRate sebesar 0.25, penentuan ini berdasarkan referensi yang didapat dari penelitian-penelitian yang telah dilakukan sebelumnya.

c. mutationRate, merupakan nilai yang akan menentukan probabilitas atau tingkat di mana perubahan acak atau mutasi genetik terjadi pada individu dalam populasi. Nilai mutationRate biasanya ditentukan secara acak antara 0 dan 1. Pada penelitian ini, ditentukan nilai mutationRate sebesar 0.1, penentuan ini berdasarkan referensi yang didapat dari penelitian-penelitian yang telah dilakukan sebelumnya.

d. maxGen, merupakan jumlah maksimum generasi atau iterasi yang akan dieksekusi oleh algoritma sebelum mencapai kriteria berhenti yang ditentukan sebelumnya. Ketika jumlah iterasi yang telah dijalankan mencapai nilai maxGen, algoritma akan berhenti. Pada penelitian ini, ditentukan nilai maxGen sebesar 15, penentuan ini berdasarkan eksperimen yang dilakukan hasil solusi paling optimal didapat pada maxGen sebesar 15. Parameter ini, digunakan sebagai kriteria penghentian karena dengan menggunakan kriteria penghentian berbasis iterasi dapat lebih stabil dibandingkan dengan penghentian berbasis nilai fitness [10].

e. differentDay, merupakan nilai penalti yang akan diberikan ketika ada keadaan di mana jadwal praktikum 2 slot memiliki hari praktikum yang berbeda. Nilai penalti ini akan memengaruhi nilai objektif dan juga fungsi fitness nantinya.

f. teachingScheduleConflict, merupakan nilai penalti yang akan diberikan ketika ada keadaan di mana jadwal praktikum bentrok dengan jadwal mengajar dosen. Nilai penalti ini akan memengaruhi nilai objektif dan juga fungsi fitness nantinya.

Tabel 4.4 Setting Parameter pada Algoritma Genetika

No	Parameter	Keterangan	Nilai
1	numofChromosome	Jumlah kromosom di dalam suatu populasi	30
2	crossoverRate	Nilai yang akan digunakan untuk melakukan crossover	0.25
3	mutationRate	Nilai yang akan digunakan untuk melakukan mutasi	0.1
4	maxGen	Jumlah maksimum iterasi dalam proses pencarian solusi	15
5	differentDay	Nilai pinalti untuk jadwal praktikum berbeda hari	20
6	teachingScheduleConflict	Nilai pinalti untuk jadwal praktikum bentrok dengan jadwal mengajar dosen	24

4.3 Implementasi Algoritma Genetika

4.3.1 Representasi Solusi

Kandidat solusi direpresentasikan dalam bentuk kromosom yang dibangkitkan secara acak. Proses optimasi nantinya akan menghasilkan solusi terbaik. Setiap kromosom terbentuk dari gen-gen sebanyak jumlah daftar mata praktikum. Setiap indeks gen mewakili kelas praktikum (mata praktikum, kelas, jumlah slot, dosen). Nilai setiap gen tersebut berupa indeks dari array data labSlots yang mewakili hari, kode jam, dan juga nama laboratorium. Proses implementasi pembangkitan kromosom dapat dilihat pada Listing 4.1 berikut.

```
1.  def initializingPopulation(self, labSlotIDs):
2.      population = [];
3.
4.      for _ in range(self.numOfChromosome):
5.          reqSlot = self.labTimeTables[0][2]
6.          randomSlotIndex = random.randint(0, len(labSlotIDs)-1)
7.          if reqSlot == 2 and randomSlotIndex+1 <= len(labSlotIDs)-1:
8.              residu = labSlotIDs[randomSlotIndex] - labSlotIDs[randomSlotIndex+1]
9.              while residu > 1:
10.                 randomSlotIndex = random.randint(0, len(labSlotIDs)-1)
11.             chromosome = [([labSlotIDs[randomSlotIndex], labSlotIDs[randomSlotIndex+1]])]
12.             labSlotIDs.pop(randomSlotIndex); labSlotIDs.pop(randomSlotIndex)
13.         else:
14.             chromosome = [([labSlotIDs[randomSlotIndex]])]
15.             labSlotIDs.pop(randomSlotIndex)
16.
17.         for labCourse in range(1, len(self.labTimeTables)):
18.             reqSlot = self.labTimeTables[labCourse][2]
19.             if reqSlot == 2:
20.                 if (len(chromosome[-1]) == 2):
21.                     if chromosome[-1][1] == len(labSlotIDs):
22.                         slots = [0, 1]
23.                     else:
```

```
24.                     slots = [(chromosome[-1][0] + 2) % (len(labSlotIDs)+2), (chromosome[-
    1][1] + 2) % (len(labSlotIDs)+2)]
25.                 elif (len(chromosome[-1]) == 1):
26.                     slots = [(chromosome[-1][0] + 1) % (len(labSlotIDs)+2), (chromosome[-1][1]
    + 2) % (len(labSlotIDs)+2)]
27.                 else:
28.                     print("persiapan butuh 3 slot")
29.                 chromosome.append(slots)
30.             else:
31.                 if (len(chromosome[-1]) == 2):
32.                     slot = [(chromosome[-1][1] + 1) % (len(labSlotIDs)+2)]
33.                 elif (len(chromosome[-1]) == 1):
34.                     slot = [(chromosome[-1][0] + 1) % (len(labSlotIDs)+2)]
35.                 chromosome.append(slot)
```

Listing 4.1 Kode Program Pembangkitan Kromosom

Listing 4.1 merupakan fungsi untuk menginisialisasi labslot sesuai dengan jumlah slot masing-masing mata praktikum. Berikut adalah penjelasan umum berdasarkan nomor baris program.

1. Baris 1: Mendefinisikan fungsi **initializingPopulation** yang mengambil argumen **labSlotIDs**.
2. Baris 2: Membuat list kosong **population** yang akan menampung kromosom-kromosom.
3. Baris 4: Melakukan iterasi sebanyak **numOfChromosome**.
4. Baris 5-15: Mencari nilai acak antara 0 sampai dengan panjangnya **labSlotIDs** dengan memperhatikan **reqSlot** sebagai nilai indeks ke-0 dari masing-masing kromosom.

5. Baris 17-35: Melanjutkan inisialisasi nilai indeks selanjutnya secara urut sampai dengan panjang **labTimeTables** dengan memperhatikan **reqSlot**.

Output pembangkitan kromosom dapat dilihat di bawah ini.

```
Kromosom:
[[165, 166], [167, 168], [169, 170], [171, 172], [173, 174], [175, 176], [177, 178], [179, 180], [181, 182], [183, 184], [185, 186], [187, 188], [189, 190], [191, 192], [193, 194], [195, 196], [197, 198], [0, 1], [2, 3], [4, 5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50], [51], [52], [53], [54], [55], [56], [57], [58], [59], [60], [61], [62], [63], [64], [65], [66], [67], [68], [69], [70], [71], [72], [73], [74], [75], [76], [77], [78], [79], [80], [81], [82], [83], [84], [85], [86], [87], [88], [89], [90], [91], [92], [93], [94], [95], [96], [97], [98], [99], [100], [101], [102], [103], [104], [105], [106]]
```

Berdasarkan output di atas, nilai dari setiap indeks merupakan representasi dari hari, kode jam, dan laboratorium yang tersedia dan nantinya akan digunakan. Berikut merupakan contoh data yang diwakilkan oleh nilai-nilai di atas.

```
Data Kromosom:
[[['Senin', 5, 'Riset'], ['Senin', 6, 'Riset']], [['Selasa', 0, 'Riset'], ['Selasa', 1, 'Riset']], [['Selasa', 2, 'Riset'], ['Selasa', 3, 'Riset']], [['Selasa', 4, 'Riset'], ['Selasa', 5, 'Riset']], [['Selasa', 6, 'Riset'], ['Rabu', 0, 'Riset']], [['Rabu', 1, 'Riset'], ['Rabu', 2, 'Riset']], [['Rabu', 3, 'Riset'], ['Rabu', 4, 'Riset']], [['Rabu', 5, 'Riset'], ['Rabu', 6, 'Riset']], [['Kamis', 0, 'Riset'], ['Kamis', 1, 'Riset']], [['Kamis', 2, 'Riset'], ['Kamis',
```

3, 'Riset']], [['Kamis', 4, 'Riset'], ['Kamis', 5, 'Riset']], [['Kamis', 6, 'Riset'], ['Jumat', 0, 'Riset']], [['Jumat', 1, 'Riset'], ['Jumat', 4, 'Riset']], [['Jumat', 5, 'Riset'], ['Jumat', 6, 'Riset']], [['Sabtu', 0, 'Riset'], ['Sabtu', 1, 'Riset']], [['Sabtu', 2, 'Riset'], ['Sabtu', 3, 'Riset']], [['Sabtu', 4, 'Riset'], ['Sabtu', 5, 'Riset']], [['Senin', 0, 'Jarkom'], ['Senin', 1, 'Jarkom']], [['Senin', 2, 'Jarkom'], ['Senin', 3, 'Jarkom']], [['Senin', 4, 'Jarkom'], ['Senin', 5, 'Jarkom']], ['Senin', 6, 'Jarkom'], ['Selasa', 0, 'Jarkom'], ['Selasa', 1, 'Jarkom'], ['Selasa', 2, 'Jarkom'], ['Selasa', 3, 'Jarkom'], ['Selasa', 4, 'Jarkom'], ['Selasa', 5, 'Jarkom'], ['Selasa', 6, 'Jarkom'], ['Rabu', 0, 'Jarkom'], ['Rabu', 1, 'Jarkom'], ['Rabu', 2, 'Jarkom'], ['Rabu', 3, 'Jarkom'], ['Rabu', 4, 'Jarkom'], ['Rabu', 5, 'Jarkom'], ['Rabu', 6, 'Jarkom'], ['Kamis', 0, 'Jarkom'], ['Kamis', 1, 'Jarkom'], ['Kamis', 2, 'Jarkom'], ['Kamis', 3, 'Jarkom'], ['Kamis', 4, 'Jarkom'], ['Kamis', 5, 'Jarkom'], ['Kamis', 6, 'Jarkom'], ['Jumat', 0, 'Jarkom'], ['Jumat', 1, 'Jarkom'], ['Jumat', 4, 'Jarkom'], ['Jumat', 5, 'Jarkom'], ['Jumat', 6, 'Jarkom'], ['Sabtu', 0, 'Jarkom'], ['Sabtu', 1, 'Jarkom'], ['Sabtu', 2, 'Jarkom'], ['Sabtu', 3, 'Jarkom'], ['Sabtu', 4, 'Jarkom'], ['Sabtu', 5, 'Jarkom'], ['Sabtu', 6, 'Jarkom'], ['Senin', 0, 'Basdat'], ['Senin', 1, 'Basdat'], ['Senin', 2, 'Basdat'], ['Senin', 3, 'Basdat'], ['Senin', 4, 'Basdat'], ['Senin', 5, 'Basdat'], ['Senin', 6, 'Basdat'], ['Selasa', 0, 'Basdat'], ['Selasa', 1, 'Basdat'], ['Selasa', 2, 'Basdat'], ['Selasa', 3, 'Basdat'], ['Selasa', 4, 'Basdat'], ['Selasa', 5, 'Basdat'], ['Selasa', 6, 'Basdat'], ['Rabu', 0, 'Basdat'], ['Rabu', 1, 'Basdat'], ['Rabu', 2, 'Basdat'], ['Rabu', 3, 'Basdat'], ['Rabu', 4, 'Basdat'], ['Rabu', 5, 'Basdat'], ['Rabu', 6, 'Basdat'], ['Kamis', 0, 'Basdat'], ['Kamis', 1, 'Basdat'], ['Kamis', 2, 'Basdat'], ['Kamis', 3, 'Basdat'], ['Kamis', 4, 'Basdat'], ['Kamis', 5, 'Basdat'], ['Kamis', 6, 'Basdat'], ['Jumat', 0, 'Basdat'], ['Jumat', 1, 'Basdat'], ['Jumat', 4, 'Basdat'], ['Jumat', 5, 'Basdat'], ['Jumat', 6, 'Basdat'], ['Sabtu', 0, 'Basdat'], ['Sabtu', 1, 'Basdat'], ['Sabtu', 2, 'Basdat'], ['Sabtu', 3, 'Basdat'], ['Sabtu', 4, 'Basdat'], ['Sabtu', 5, 'Basdat'], ['Sabtu', 6, 'Basdat'], ['Senin', 0, 'Komdas'], ['Senin', 1, 'Komdas'], ['Senin', 2, 'Komdas'], ['Senin', 3, 'Komdas'], ['Senin', 4, 'Komdas'], ['Senin', 5, 'Komdas'], ['Senin', 6, 'Komdas'], ['Selasa', 0, 'Komdas'], ['Selasa', 1, 'Komdas'], ['Selasa', 2,

```
'Komdas'], ['Selasa', 3, 'Komdas'], ['Selasa', 4, 'Komdas'], ['Selasa', 5, 'Komdas'], ['Selasa', 6, 'Komdas'], ['Rabu', 0, 'Komdas'], ['Rabu', 1, 'Komdas'], ['Rabu', 2, 'Komdas'], ['Rabu', 3, 'Komdas'], ['Rabu', 4, 'Komdas'], ['Rabu', 5, 'Komdas'], ['Rabu', 6, 'Komdas'], ['Kamis', 0, 'Komdas'], ['Kamis', 1, 'Komdas'], ['Kamis', 2, 'Komdas'], ['Kamis', 3, 'Komdas'], ['Kamis', 4, 'Komdas'], ['Kamis', 5, 'Komdas']]
```

4.3.2 Fungsi Objektif

Kromosom-kromosom yang telah dibangkitkan di dalam populasi kemudian akan dihitung nilai objektifnya. Nilai objektif dihitung berdasarkan jumlah nilai pinalti dari setiap kromosom. Nilai pinalti sendiri didapatkan dari setiap gen yang memiliki 2 kondisi, yaitu ada jadwal praktikum bentrok dengan jadwal mengajar dosen dan juga ada jadwal praktikum 2 slot tetapi berbeda hari. Nilai objektif ini nantinya akan digunakan untuk menghitung fungsi fitness. Fungsi objektif diimplementasikan pada kode program berikut.

```
1.  def getSatuMataPraktikum(self, genID):
2.      for labTimeTableID in range(len(self.labTimeTables)):
3.          if genID == labTimeTableID:
4.              return self.labTimeTables[labTimeTableID]
5.
6.  def getTeachingTimeByIDTime(self, lecturerTimes, teachingIDTimes):
7.      teachingTimeInHours = []
8.      for teachingIDTime in teachingIDTimes:
9.          teachingTimeInHours.append(lecturerTimes[teachingIDTime])
10.     return ([teachingTimeInHours[0][0], teachingTimeInHours[-1][-1]])
```

```
11.
12. def isConflict(self, teachingTimeLowerUpperBound,
    slotPraktikumTimeLowerUpperBound):
13.     techingTimeLowerBound = teachingTimeLowerUpperBound[0]
14.     teachingTimeUpperBound = teachingTimeLowerUpperBound[1]
15.     slotPraktikumTimeLowerBound = slotPraktikumTimeLowerUpperBound[0]
16.     slotPraktikumTimeUpperBound = slotPraktikumTimeLowerUpperBound[1]
17.     if slotPraktikumTimeLowerBound >= techingTimeLowerBound and
    slotPraktikumTimeLowerBound <= teachingTimeUpperBound:
18.         return 20
19.     if slotPraktikumTimeUpperBound >= techingTimeLowerBound and
    slotPraktikumTimeUpperBound <= teachingTimeUpperBound:
20.         return 20
21.
22. def teachingAndSlotTimeIsConflict(self, teachingTimeIDs, jadwalSlot):
23.     labTimers = getTimetableData()['labTimers']
24.     ret = 0
25.     lecturerTimers = getTimetableData()['lecturerTimers']
26.
27.     if len(jadwalSlot) == 1:
28.         slotTimerBound = {'lowerBound': labTimers[jadwalSlot[0][1]][0], 'upperBound':
    labTimers[jadwalSlot[0][1]][1]}
29.     if len(jadwalSlot) > 1:
```

```
30.         slotTimerBound = {'lowerBound': labTimers[jadwalSlot[0][1]][0], 'upperBound':
      labTimers[jadwalSlot[1][1]][1]}
31.     combinedAllTeachingTimeIDs = []
32.     for teachingTimeID in teachingTimeIDs:
33.         for i in lecturerTimers[teachingTimeID]:
34.             combinedAllTeachingTimeIDs.append(i)
35.     teachingTimerBound = {'lowerBound': combinedAllTeachingTimeIDs[0], 'upperBound':
      combinedAllTeachingTimeIDs[-1]}
36.
37.     if slotTimerBound['lowerBound'] >= teachingTimerBound['lowerBound'] and
      slotTimerBound['lowerBound'] <= teachingTimerBound['upperBound']:
38.         ret = 1
39.     if slotTimerBound['upperBound'] > teachingTimerBound['lowerBound'] and
      slotTimerBound['upperBound'] <= teachingTimerBound['upperBound']:
40.         ret = 1
41.     return ret
42.
43. def getTeachingScheduleByLecturer(self, teachingSchedules, satuMataPraktikum,
      jadwalSlot):
44.     lecturerName = satuMataPraktikum[3]
45.     for teachingSchedule in teachingSchedules:
46.         lecturerTeacherName = teachingSchedule[0]
47.         if lecturerName == lecturerTeacherName:
```

```
48.             teachingDay = teachingSchedule[1]
49.             teachingIDTimes = teachingSchedule[4]
50.             slotInDay = jadwalSlot[0][0]
51.             if slotInDay == teachingDay and
    self.teachingAndSlotTimeIsConflict(teachingIDTimes, jadwalSlot):
52.                 return [self.penaltyValues['teachingScheduleConflict'], teachingSchedule]
53.             else:
54.                 return 0
55.
56. def isDifferentSlotDay(self, jadwalSlot):
57.     return jadwalSlot[0][0] != jadwalSlot[1][0]
58.
59. def calcPenalties(self, chromosome):
60.     penalties = []; differentDaysPenalty = []; conflictWithTeachingClassesPenalty = [];
    penalty = []; sameslot = []
61.     ret = {'penalties':0, 'differentDays':None, 'conflictTeachingClass':None, 'sameSlot': None}
62.
63.     for genID in range(len(chromosome)):
64.         if len(self.getLabSlotInfo(self.labSlots, chromosome[genID])) == 2 and
    self.isDifferentSlotDay(self.getLabSlotInfo(self.labSlots, chromosome[genID])):
65.             penalti = self.penaltyValues['differentDay']
66.             differentDaysPenalty.append([self.getSatuMataPraktikum(genID),
    self.getLabSlotInfo(self.labSlots, chromosome[genID])])
```

```
67.         else:
68.             penalti = self.getTeachingScheduleByLecturer(self.lecturerTeachingSchedules,
        self.getSatuMataPraktikum(genID), self.getLabSlotInfo(self.labSlots,
        chromosome[genID]))
69.         if isinstance(penalti, list):
70.             conflictWithTeachingClassesPenalty.append([penalti[1],
        self.getSatuMataPraktikum(genID), self.getLabSlotInfo(self.labSlots,
        chromosome[genID])])
71.             penalti = penalti[0]
72.         penalties.append(penalti)
```

Listing 4.2 Kode Program Fungsi Objektif

Listing 4.2 merupakan kode program untuk menghitung fungsi objektif berdasarkan jumlah pinalti dari setiap kromosom. Berikut adalah penjelasan umum berdasarkan baris kode program.

1. Baris 1-4: Fungsi **getSatuMataPraktikum** mengambil satu mata praktikum tertentu berdasarkan **genID**.
2. Baris 6-10: Fungsi **getTeachingTimeByIDTime** mengambil waktu mengajar pertama dan terakhir dosen dalam jam.
3. Baris 12-20: Fungsi **isConflict** memeriksa konflik antara waktu mengajar dosen dan waktu slot praktikum yang diberikan.
4. Baris 22-41: Fungsi **teachingAndSlotTimeIsConflict** memeriksa apakah ada konflik antara waktu mengajar dosen dan waktu slot praktikum yang diberikan.
5. Baris 43-54: Fungsi **getTeachingScheduleByLecturer** mengambil jadwal mengajar, jadwal praktikum untuk satu mata praktikum tertentu, dan slot praktikum yang diberikan, lalu

memeriksa apakah ada konflik dengan jadwal mengajar dosen dan mengembalikan nilai pinalti jika ada.

6. Baris 56-57: Fungsi **isDifferentSlotDay** memeriksa apakah dua slot praktikum berada di hari yang berbeda.
7. Baris 59-72: Fungsi **calcPenalties** menghitung semua pinalti untuk suatu kromosom tertentu. Ini mencakup penalti untuk slot praktikum yang berada di hari yang berbeda dan konflik dengan jadwal mengajar dosen.

Output fungsi objektif sebagai berikut.

```
Nilai objektif:
176
```

4.3.3 Fungsi Fitness

Fungsi fitness ditentukan dari seberapa besar nilai dari fungsi objektif dengan persamaan sebagai berikut:

$$\frac{1}{(1+Nilai\ Objektif)}.$$

Oleh karena itu, semakin besar nilai objektif maka akan semakin kecil nilai fitness, dan semakin kecil nilai objektif maka akan semakin besar nilai fitnessnya. Dalam penelitian ini, akan mengejar nilai objektif sekecil mungkin sehingga akan mendapatkan nilai fitness sebesar mungkin karena semakin besar nilai fitness maka akan semakin baik solusinya. Berikut merupakan kode program implementasi fungsi fitness.

```
1. def calcFitnessValue(self, objectiveValue):
2.     return 1 / (1 + objectiveValue)
```

Listing 4.3 Kode Program Fungsi Fitness

Dengan Listing 4.3, maka akan didapatkan nilai fitness berdasarkan nilai objektif dari masing masing kromosom dalam suatu populasi. Berikut adalah output setelah adanya implementasi fungsi fitness pada Listing 4.3.

```
Nilai objektif setiap kromosom:
[132, 156, 204, 136, 204, 204, 180, 132, 180, 132, 324, 156, 132, 132, 204, 132, 180, 160, 132,
252, 132, 132, 228, 156, 156, 156, 228, 228, 324, 156, 204, 60, 180, 156, 132, 204, 156]

Nilai objektif terkecil: 60  Index:  31

Nilai fitness setiap kromosom:
[0.007518796992481203,  0.006369426751592357,  0.004878048780487805,
0.007299270072992700,  0.004878048780487805,  0.004878048780487805,
0.005524861878453038,  0.007518796992481203,  0.005524861878453038,
0.007518796992481203,  0.003076923076923077,  0.006369426751592357,
0.007518796992481203,  0.007518796992481203,  0.004878048780487805,
0.007518796992481203,  0.005524861878453038,  0.006211180124223602,
0.007518796992481203,  0.003952569169960474,  0.007518796992481203,
0.007518796992481203,  0.004366812227074236,  0.006369426751592357,
0.006369426751592357,  0.006369426751592357,  0.004366812227074236,
0.004366812227074236,  0.003076923076923077,  0.006369426751592357,
0.004878048780487805,  0.016393442622950820,  0.005524861878453038,
```

```
0.006369426751592357,  0.007518796992481203,  0.004878048780487805,
0.006369426751592357]

Nilai fitness terbesar: 0.016393442622950820  Index:  31
```

Dapat dilihat dari output di atas, terbukti bahwa index nilai objektif terkecil sama dengan index nilai fitness terbesar sehingga semakin kecil nilai objektif maka akan semakin besar nilai fitness. Dalam hal ini, nilai fitness terbesar akan mendapatkan kesempatan lebih besar untuk terpilih ke dalam seleksi yang akan dilakukan pada langkah selanjutnya.

4.3.4 Pembaruan Populasi

Pembaruan populasi adalah tahap kritis dalam algoritma genetika di mana generasi baru individu-individu dihasilkan berdasarkan kinerja individu-individu pada generasi sebelumnya. Proses ini melibatkan seleksi, crossover, dan mutasi.

4.3.4.1 Seleksi

Tahap pertama dalam pembaruan populasi adalah diperbarui dengan proses seleksi berdasarkan nilai fitness dari setiap kromosom yang telah diperoleh di proses sebelumnya. Dalam penelitian ini, seleksi yang dilakukan yaitu menggunakan metode Roulette Wheel Selection, dimana nilai fitness akan dilibatkan dalam proses seleksi ini. Nilai fitness yang lebih besar memiliki persentase lebih besar juga untuk terpilih. Perhitungan probabilitas tiap nilai fitness kromosom dapat diketahui sebagai berikut: $\frac{nilai\ fitness\ tiap\ kromosom}{total\ nilai\ fitness\ keseluruhan\ kromosom}$. Kode program proses perhitungan probabilitas tiap nilai fitness dari kromosom adalah sebagai berikut.

```
1.  def fitnessValueSummation(self, population):
2.      fitnessValues = []
3.      for chromosome in population:
4.          fitnessValues.append(chromosome['fitnessValue'])
5.      return sum(fitnessValues)
6.
7.  def rouletteWheelSelection(self, population):
8.      probCummulative = 0; probCummulatives = []
9.      sumOfFitnessValue = self.fitnessValueSummation(population)
10.     for chromosome in population:
11.         probability = chromosome['fitnessValue'] / sumOfFitnessValue
12.         probCummulative = probCummulative + probability
13.         probCummulatives.append(probCummulative)
```

Listing 4.4 Kode Program Roulette Wheel Selection

Listing 4.4 merupakan kode program untuk menghitung fungsi objektif berdasarkan jumlah pinalti dari setiap kromosom. Berikut adalah penjelasan umum berdasarkan baris kode program.

1. Baris 1-5: Fungsi **FitnessValueSummation** menjumlahkan nilai fungsi fitness tiap kromosom yang ada pada populasi.
2. Baris 7-13: Melakukan perhitungan probabilitas kumulatif tiap fitness value kromosom yang ada pada populasi.

Berikut adalah output dari kode program Listing 4.4.

```
Probabilitas Kumulatif:
```

```
[0.04967096159428665, 0.09037874214900277, 0.1310865227037189, 0.16557145782331917,
0.18921396792714992, 0.22369890304675022, 0.25361125008419355, 0.2800216638099362,
0.3145065989295365, 0.3552143794842526, 0.38162479320999526, 0.4115371402474386,
0.44602207536703886, 0.4759344224044822, 0.5058467694419255, 0.5465545499966417,
0.5810394851162419, 0.6217472656709581, 0.6481576793967008, 0.682642614516301,
0.7233503950710172, 0.7497608087967598, 0.7904685893514759, 0.831176369906192,
0.8575867836319346, 0.8971060158492867, 0.9270183628867301, 0.9436772392368139,
0.9735895862742573, 1.0]
```

Setelah mendapatkan probabilitas dari tiap kromosom, maka akan dilakukan proses pemilihan kandidat kromosom yang akan diganti. Pemilihan ini dilakukan dengan cara membangkitkan nilai acak (0 dan 1) dan membandingkannya dengan nilai probabilitas kumulatif ke-i. Kode program pemilihan kandidat kromosom yang akan diganti adalah sebagai berikut:

```
1.  selectedCandidateChromosomes = []
2.      while len(selectedCandidateChromosomes) == 0:
3.          for i in range(len(probCummulatives)):
4.              randomValue = random.uniform(0,1)
5.              if randomValue > probCummulatives[i] and randomValue <=
    probCummulatives[i+1]:
6.                  selectedCandidateChromosomes.append({
7.                      'index':i,
8.                      'fitnessValue':population[i+1]['fitnessValue'],
9.                      'objectiveValue':population[i+1]['objectiveValue'],
```

```
10.         'differentDays':population[i+1]['differentDays'],
11.         'sameSlot':population[i+1]['sameSlot'],
12.         'chromosome':population[i+1]['chromosome'],
13.         'conflictTeachingClass':population[i+1]['conflictTeachingClass']})
```

Listing 4.5 Kode Program Pemilihan Kandidat Kromosom

Berikut adalah penjelasan umum Listing 4.5 berdasarkan baris kode program.

1. Baris 1: Membuat sebuah list kosong yang akan digunakan untuk menyimpan kromosom yang terpilih.
2. Baris 2: Menjalankan perulangan sampai list s**electedCandidateChromosomes** terisi setidaknya 1.
3. Baris 3: Mengulangi iterasi sepanjang jumlah **probCummulatives**
4. Baris 4-12: Untuk setiap iterasi, dilakukan pembangkitan nilai acak antara 0 dan 1 (**randomValue**) dan dilakukan pengecekan untuk menentukan kromosom mana yang akan dipilih berdasarkan probabilitas kumulatif. Jika nilai acak berada dalam rentang probabilitas kumulatif tertentu, kromosom tersebut akan ditambahkan ke dalam list selectedCandidateChromosomes.

Berikut adalah output dari Listing 4.5.

```
[{'index': 11, 'fitnessValue': 0.004366812227074236, 'objectiveValue': 228, 'differentDays': [[['Alpro', 4, 2, 'Dr. Ardiansyah, S.T., M.Cs.'], [['Selasa', 6, 'Komdas'], ['Rabu', 0, 'Komdas']]], [['Alpro', 11, 2, 'Drs., Tedy Setiadi, M.T.'], [['Kamis', 6, 'Komdas'], ['Jumat', 0, 'Komdas']]], [['KDJK', 6, 2, 'Nuril Anwar, S.T., M.Kom'], [['Sabtu', 6, 'Komdas'], ['Senin', 0, 'Mulmed']]]], 'sameSlot': [], 'chromosome': [[87, 88], [89, 90], [91, 92], [93, 94], [95, 96], [97, 98], [99, 100], [101, 102], [103, 104], [105, 106], [107, 108], [109, 110], [111, 112], [113, 114], [115, 116], [117, 118], [119, 120],
```

[121, 122], [123, 124], [125, 126], [127], [128], [129], [130], [131], [132], [133], [134], [135], [136], [137], [138], [139], [140], [141], [142], [143], [144], [145], [146], [147], [148], [149], [150], [151], [152], [153], [154], [155], [156], [157], [158], [159], [160], [161], [162], [163], [164], [165], [166], [167], [168], [169], [170], [171], [172], [173], [174], [175], [176], [177], [178], [179], [180], [181], [182], [183], [184], [185], [186], [187], [188], [189], [190], [191], [192], [193], [194], [195], [196], [197], [198], [199], [0], [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27]], 'conflictTeachingClass': [[['Dr. Ardiansyah, S.T., M.Cs.', 'Selasa', 'RPL', 3, [9, 10, 11]], ['Alpro', 3, 2, 'Dr. Ardiansyah, S.T., M.Cs.'], [['Selasa', 4, 'Komdas'], ['Selasa', 5, 'Komdas']], [91, 92], 24], [['Anna Hendri Soleliza Jones, S.Kom., M.Cs.', 'Rabu', 'KCB', 4, [4, 5]], ['KCB', 10, 1, 'Anna Hendri Soleliza Jones, S.Kom., M.Cs.'], [['Rabu', 2, 'Mulmed']], [136], 24], [['Anna Hendri Soleliza Jones, S.Kom., M.Cs.', 'Rabu', 'KCB', 4, [4, 5]], ['KCB', 11, 1, 'Anna Hendri Soleliza Jones, S.Kom., M.Cs.'], [['Rabu', 3, 'Mulmed']], [137], 24], [['Dr., H. Imam Riadi, M.Kom.', 'Jumat', 'KI', 2, [2, 3]], ['KI', 5, 1, 'Dr., H. Imam Riadi, M.Kom.'], [['Jumat', 0, 'Mulmed']], [148], 24], [['Supriyanto, S.T., M.T.', 'Senin', 'IMK', 2, [8, 9]], ['VisDat', 3, 1, 'Supriyanto, S.T., M.T.'], [['Senin', 4, 'Riset']], [164], 24], [['Supriyanto, S.T., M.T.', 'Senin', 'IMK', 2, [8, 9]], ['VisDat', 4, 1, 'Supriyanto, S.T., M.T.'], [['Senin', 5, 'Riset']], [165], 24], [['Adhi Prahara, S.Si., M.Cs.', 'Senin', 'Deep Learning', 2, [10, 11]], ['Penglihatan Komputer', 2, 1, 'Adhi Prahara, S.Si., M.Cs.'], [['Senin', 6, 'Jarkom']], [6], 24]]}]

Setelah itu, kandidat kromosom yang terpilih akan diganti dengan kromosom selanjutnya. Kode program pergantian kromosom yang telah terpilih adalah sebagai berikut.

```
1. for candidateNewChromosome in candidateNewChromosomes:
```

```
2.    population[candidateNewChromosome['index']]['fitnessValue'] =
      candidateNewChromosome['fitnessValue']
3.    population[candidateNewChromosome['index']]['objectiveValue'] =
      candidateNewChromosome['objectiveValue']
4.    population[candidateNewChromosome['index']]['differentDays'] =
      candidateNewChromosome['differentDays']
5.    population[candidateNewChromosome['index']]['conflictTeachingClass'] =
      candidateNewChromosome['conflictTeachingClass']
6.    population[candidateNewChromosome['index']]['sameSlot'] =
      candidateNewChromosome['sameSlot']
7.    population[candidateNewChromosome['index']]['chromosome'] =
      candidateNewChromosome['chromosome']
```

Listing 4.6 Kode Program Pergantian Kromosom

Berikut adalah penjelasan umum Listing 4.6 berdasarkan baris kode program.

1. Baris 1: Melakukan perulangan melalui list **candidateNewChromosomes** yang berisi kromosom terpilih.
2. Baris 2-7: Memperbarui atau mengganti kromosom yang ada dalam populasi dengan kromosom yang baru dihasilkan.

4.3.4.2 Crossover

Setelah proses seleksi dilakukan, tahap pembaruan populasi selanjutnya adalah crossover. Pada proses ini, kromosom anak atau yang bisa disebut dengan offspring akan terbentuk dari hasil perkawinan silang antara pasangan induk atau parents yang terpilih. Pasangan induk akan dipilih secara acak berdasarkan crossover rate yang telah ditentukan pada parameter dengan kode program sebagai berikut.

```
def generateRandomValues(self):
    rets = []
    for i in range(self.numOfChromosome):
        if random.uniform(0,1) < self.cr:
            rets.append(i)
    return rets
```

Listing 4.7 Kode Program Kemungkinan Kromosom Terpilih

```
randomIndexValues = self.generateRandomValues()
while len(randomIndexValues) <= 1:
    randomIndexValues = self.generateRandomValues()
selectedChromosomesToCrossover = []
for i in randomIndexValues:
selectedChromosomesToCrossover.append({'chromosomes':population[i]['chromosome'],
'index':i})
parentCandidatesIndex = []
for i in selectedChromosomesToCrossover:
    for j in selectedChromosomesToCrossover:
        if i['index'] != j['index']:
            parentCandidatesIndex.append([i['index'],j['index']])
sortedParentIndexes = []
for parentIndex in parentCandidatesIndex:
    parentIndex.sort()
    sortedParentIndexes.append(parentIndex)
```

```
16. finalParentIndexes = []
17. for sortedParentIndex in sortedParentIndexes:
18.     if sortedParentIndex not in finalParentIndexes:
19.         finalParentIndexes.append(sortedParentIndex)
```

Listing 4.8 Kode Program Pemilihan Kromosom untuk Crossover

Dengan Listing 4.7, didapatkan indeks-indeks random yang akan dicrossover. Berikut adalah penjelasan umum Listing 4.7 berdasarkan baris kode program.

1. Baris 1: Mendeklarasi fungsi **generateRandomValues.**
2. Baris 2: Membuat list kosong **rets**.
3. Baris 3-5: Melakukan perulangan sebanyak jumlah kromosom, dan melakukan pengecekan jika nilai acak lebih kecil dari nilai crossover rate, maka indeks kromosom akan ditampung ke dalam list **rets.**

Dengan Listing 4.8, telah dilakukan pemilihan beberapa induk kromoson yang akan dicrossover. Berikut adalah penjelasan umum Listing 4.8 berdasarkan baris kode program.

1. Baris 1: Membuat variabel **randomIndexValues** yang berisi nilai-nilai acak yang dihasilkan dari fungsi **generateRandomValues()** yang merupakan fungsi untuk menghasilkan indeks-indeks acak dari suatu populasi kromosom.
2. Baris 2-3: Melakukan perulangan while untuk memastikan bahwa jumlah nilai dalam **randomIndexValues** tidak kurang dari atau sama dengan 1.
3. Baris 4: Membuat list kosong **selectedChromosomesToCrossover** yang akan digunakan untuk menyimpan kromosom-kromosom yang dipilih untuk crossover.

4. Baris 5-6: Melakukan perulangan untuk setiap nilai i dalam **randomIndexValues** dan menambahkan kromosom dengan indeks yang sesuai ke dalam **selectedChromosomesToCrossover**.
5. Baris 7-11: Membuat **parentCandidatesIndex** yang berisi semua kombinasi unik indeks pasangan kromosom yang dipilih untuk crossover.
6. Baris 12-15: Mengurutkan indeks-indeks dalam **parentCandidatesIndex** dan menyimpannya dalam **sortedParentIndexes** untuk memastikan bahwa pasangan kromosom yang sama tidak dihitung dua kali.
7. Baris 16-19: Membuat **finalParentIndexes** yang berisi pasangan indeks kromosom yang unik, untuk memastikan bahwa setiap pasangan kromosom hanya dipertimbangkan satu kali dalam proses crossover.

Output dari Listing 4.8 adalah sebagai berikut.

```
Induk parents yang terpilih untuk dicrossover:
[[0, 4], [0, 15], [0, 17], [0, 20], [0, 21], [0, 22], [0, 25], [4, 15], [4, 17], [4, 20], [4, 21], [4, 22], [4, 25], [15, 17], [15, 20], [15, 21], [15, 22], [15, 25], [17, 20], [17, 21], [17, 22], [17, 25], [20, 21], [20, 22], [20, 25], [21, 22], [21, 25], [22, 25]]
```

Pasangan induk dipilih secara acak dengan memperhatikan tidak ada pasangan induk yang sama terjadi berulang. Setelah mendapatkan pasangan-pasangan induk untuk dilakukan crossover, selanjutnya adalah menentukan titik potong secara acak. Pasangan induk dan titik potong sudah diperoleh maka proses crossover sudah dapat dilakukan. Kode program implementasi crossover adalah sebagai berikut:

```
1.  for parentsIndex in finalParentIndexes:
2.      cutPointIndex = random.randint(0, numOfDimension-2)
3.      offset = self.crossOver(cutPointIndex, population, parentsIndex)
```

Listing 4.9 Kode Program Pembuatan Titik Potong

```
1.  def crossOver(self, cutPointIndex, population, parentsIndex):
2.      offsets = []
3.      parent1 = population[parentsIndex[0]]['chromosome']
4.      parent2 = population[parentsIndex[1]]['chromosome']
5.      numOfDimension = len(self.labTimeTables)
6.      if cutPointIndex == numOfDimension-1:
7.          for i in range(numOfDimension):
8.              if i < numOfDimension-1:
9.                  offsets.append(parent2[i])
10.             else:
11.                 offsets.append(parent1[cutPointIndex])
12.     else:
13.         for i in range(numOfDimension):
14.             if i <= cutPointIndex:
15.                 offsets.append(parent1[i])
16.             else:
17.                 offsets.append(parent2[i])
```

```
18.      objectiveValues = self.calcPenalties(offsets)
19. return {
20.          'fitnessValue':self.calcFitnessValue(objectiveValues['penalties']),
21.          'objectiveValue':objectiveValues['penalties'],
22.          'differentDays':objectiveValues['differentDays'],
23.          'conflictTeachingClass':objectiveValues['conflictTeachingClass'],
24.          'sameSlot':objectiveValues['sameSlot'],
25.          'chromosome':offsets
26.       }
```

Listing 4.10 Kode Program Crossover

Dengan Listing 4.9, telah didapatkan nilai random sebagai nilai titik potong untuk proses crossover. Berikut adalah penjelasan umum Listing 4.9 berdasarkan baris kode program.

1. Baris 1: Melakukan iterasi melalui list **finalParentIndexes** yang berisi indeks-indeks pasangan induk.
2. Baris 2: Menghasilkan indeks acak untuk titik potong (cut point) antara 0 dan **numOfDimension-2**. Ini menentukan di mana kromosom akan dipotong saat melakukan operasi crossover.
3. Baris 3: Memanggil fungsi **crossover()** dengan parameter **cutPointIndex**, **population**, dan **parentsIndex**.

Dengan Listing 4.10, telah dilakukan proses crossover antar beberapa induk kromosom dan titik potong yang telah dicari sebelumnya. Berikut adalah penjelasan umum Listing 4.10 berdasarkan baris kode program.

1. Baris 1: Fungsi **crossOver** didefinisikan dengan parameter **cutPointIndex**, **population**, dan **parentsIndex**.
2. Baris 2: Membuat list **offsets** untuk menyimpan kromosom hasil crossover.
3. Baris 3-4: Mendeklarasikan dua orang tua yang akan dicrossover.
4. Baris 5: mendeklarasikan **numOfDimension** bernilai sama dengan panjang **self.labTimeTables**.
5. Baris 6-17: Fungsi melakukan crossover berdasarkan **cutPointIndex** yang diberikan. Jika **cutPointIndex** adalah indeks terakhir dari kromosom, maka dilakukan operasi crossover sederhana dimana elemen-elemen pertama diambil dari orang tua kedua dan elemen terakhir diambil dari orang tua pertama. Jika **cutPointIndex** bukan indeks terakhir, maka dilakukan operasi crossover yang lebih kompleks dimana elemen-elemen sebelum indeks dipertahankan dari orang tua pertama, dan elemen setelah indeks dipertahankan dari orang tua kedua.
6. Baris 18: Setelah crossover selesai, dilakukan perhitungan nilai objektif (**objectiveValues**) berdasarkan kromosom hasil crossover menggunakan fungsi **calcPenalties**.
7. Baris 19-26: Hasil dari operasi crossover, nilai fitness, nilai objektif, dan kromosom hasil crossover dikembalikan dalam bentuk dictionary.

Output dari Listing 4.10 (salah satu pasangan induk kromosom yang terlibat dalam proses crossover) adalah sebagai berikut.

```
parent 1:
[[128, 129], [130, 131], [132, 133], [134, 135], [136, 137], [138, 139], [140, 141], [142, 143],
[144, 145], [146, 147], [148, 149], [150, 151], [152, 153], [154, 155], [156, 157], [158, 159],
[160, 161], [162, 163], [164, 165], [166, 167], [168, 169], [170, 171], [172, 173], [174, 175],
```

[176, 177], [178, 179], [180, 181], [182, 183], [184, 185], [186, 187], [188, 189], [190, 191], [192, 193], [194], [195], [196], [197], [198], [199], [200], [201], [202], [203], [204], [0], [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50], [51], [52], [53], [54], [55], [56], [57], [58], [59], [60], [61], [62], [63], [64], [65], [66], [67], [68], [69], [70], [71], [72], [73], [74], [75], [76], [77], [78], [79], [80], [81], [82], [83], [84], [85], [86], [87], [88], [89], [90], [91], [92], [93], [94]]

parent 2:

[[118, 119], [120, 121], [122, 123], [124, 125], [126, 127], [128, 129], [130, 131], [132, 133], [134, 135], [136, 137], [138, 139], [140, 141], [142, 143], [144, 145], [146, 147], [148, 149], [150, 151], [152, 153], [154, 155], [156, 157], [158, 159], [160, 161], [162, 163], [164, 165], [166, 167], [168, 169], [170, 171], [172, 173], [174, 175], [176, 177], [178, 179], [180, 181], [182, 183], [184], [185], [186], [187], [188], [189], [190], [191], [192], [193], [194], [195], [196], [197], [198], [199], [200], [201], [202], [203], [204], [0], [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50], [51], [52], [53], [54], [55], [56], [57], [58], [59], [60], [61], [62], [63], [64], [65], [66], [67], [68], [69], [70], [71], [72], [73], [74], [75], [76], [77], [78], [79], [80], [81], [82], [83], [84]]

hasil crossover:

[[128, 129], [130, 131], [132, 133], [134, 135], [136, 137], [138, 139], [140, 141], [142, 143], [144, 145], [146, 147], [148, 149], [150, 151], [152, 153], [154, 155], [156, 157], [158, 159], [160, 161], [162, 163], [164, 165], [166, 167], [168, 169], [170, 171], [172, 173], [174, 175],

[176, 177], [178, 179], [180, 181], [182, 183], [184, 185], [186, 187], [188, 189], [190, 191], [192, 193], [194], [195], [196], [197], [198], [199], [200], [201], [202], [203], [204], [0], [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50], [51], [52], [53], [54], [55], [56], [57], [58], [59], [60], [61], [62], [63], [64], [65], [66], [67], [68], [69], [70], [71], [72], [73], [74], [75], [76], [77], [78], [69], [70], [71], [72], [73], [74], [75], [76], [77], [78], [79], [80], [81], [82], [83], [84]]

Kromosom baru yang telah didapat tetap akan dihitung nilai fitnessnya.

4.3.4.3 Mutasi

Tahap pembaruan populasi selanjutnya adalah Mutasi. Pada tahap ini, akan memilih indeks kromosom secara acak dan juga index gen secara acak dari indeks kromosom yang telah terpilih tadi. Indeks gen yang terpilih nantinya akan dimutasi dan diganti dengan indeks labslot yang belum terpilih pada kromosom ini secara acak kembali. Mutasi akan dilakukan sebanyak n dengan persamaan sebagai berikut.

$$n = mutationRate * (jumlah\ kromosom * jumlah\ kelas\ praktikum)$$

Kode program implementasi mutasi adalah sebagai berikut.

```
1. numOfMutation = round(self.mr * (self.numOfChromosome * numOfDimension))
2. numOfIndex = len(population) * numOfDimension
3. selectedChromosomeIndexes = []
4. for i in range(numOfMutation):
```

```
5.      selectedChromosomeIndex = random.randint(0,numOfIndex-1)
6.      selectedChromosomeIndexes.append(selectedChromosomeIndex)
7.
8.  selectedChromosomeIndexes = list(set(selectedChromosomeIndexes))
9.
10. for indeks in selectedChromosomeIndexes:
11.     chromosomeindex = math.floor(indeks / numOfDimension)
12.     if indeks % numOfDimension == 0 and indeks != 0:
13.         genindex = 120
14.         chromosomeindex -=1
15.     else:
16.         genindex = indeks % numOfDimension
17.         if genindex != 0 and chromosomeindex != 0:
18.             genindex -= 1
19.     mutatedChromosome = population[chromosomeindex]['chromosome']
20.     selectedValue = mutatedChromosome[genindex]
21.     total_indeks = set(range(self.numOfLabSlot))
22.
23.     used_indeks = set()
24.     for item in mutatedChromosome:
25.         used_indeks.update(item)
26.
27.     unused_indeks = total_indeks - used_indeks
```

```
28.
29.     unused_labslot = sorted(list(unused_indeks))
30.     selectedIndex = random.randint(0, len(unused_labslot)-1)
31.
32.     if len(selectedValue) == 1:
33.         selectedValue[0] = unused_labslot[selectedIndex]
34.         unused_labslot.pop(selectedIndex)
35.     if len(selectedValue) == 2 and selectedIndex+1 <= len(unused_labslot)-1:
36.         residu = unused_labslot[selectedIndex] - unused_labslot[selectedIndex+1]
37.         while residu < -1:
38.             selectedIndex = random.randint(0, len(unused_labslot)-1)
39.             if selectedIndex+1 <= len(unused_labslot)-1:
40.                 residu = unused_labslot[selectedIndex] - unused_labslot[selectedIndex+1]
41.         selectedValue[0] = unused_labslot[selectedIndex]
42.         selectedValue[1] = unused_labslot[selectedIndex+1]
43.         unused_labslot.pop(selectedIndex); unused_labslot.pop(selectedIndex)
44.     objectiveValues = self.calcPenalties(mutatedChromosome)
45.     population[chromosomeindex]['fitnessValue'] =
    self.calcFitnessValue(objectiveValues['penalties'])
46.     population[chromosomeindex]['objectiveValue'] = objectiveValues['penalties']
47.     population[chromosomeindex]['differentDays'] = objectiveValues['differentDays']
48.     population[chromosomeindex]['conflictTeachingClass'] =
    objectiveValues['conflictTeachingClass']
```

```
49.  population[chromosomeindex]['sameSlot'] = objectiveValues['sameSlot']
50.  population[chromosomeindex]['chromosome'] = mutatedChromosome
```

Listing 4.11 Kode Program Mutasi

Dengan Listing 4.11, telah dilakukan proses mutasi kepada beberapa gen indeks terpilih dari kromosom yang terpilih secara acak. Berikut adalah penjelasan umum tentang 4.11 berdasarkan baris program.

1. Baris 1: Menghitung jumlah mutasi berdasarkan nilai self.mr (mutation rate), jumlah kromosom, dan jumlah dimensi.
2. Baris 2: Menghitung jumlah indeks berdasarkan panjang populasi dan jumlah dimensi.
3. Baris 3-6: Memilih indeks kromosom secara acak sebanyak **numOfMutation**.
4. Baris 8: Menghilangkan duplikat indeks yang dipilih.
5. Baris 10-44: Melakukan proses mutasi pada kromosom yang dipilih.
6. Baris 11-19: Mengatur indeks gen yang akan dimutasi.
7. Baris 23-31: Mencari indeks laboratorium yang tidak digunakan.
8. Baris 32-43: Memilih nilai baru untuk gen yang akan dimutasi dan mengupdate kromosom dengan nilai baru tersebut.
9. Baris 44-50: Menghitung nilai fitness dan nilai objektif baru untuk kromosom yang dimutasi, dan mengupdate populasi dengan informasi baru dari kromosom yang dimutasi.

Output dari indeks gen yang terpilih adalah sebagai berikut.

```
Indeks gen yang terpilih:
[4102, 2057, 9, 15, 21, 4121, 2073, 4137, 2096, 49, 2101, 54, 4151, 4156, 2114, 2119, 73, 88, 89, 2138,
2142, 2143, 95, 102, 2154, 108, 117, 123, 131, 2182, 135, 134, 147, 2201, 2213, 176, 181, 2235, 192,
199, 201, 2273, 2279, 238, 2298, 2306, 259, 2319, 2320, 2321, 2324, 2340, 294, 2342, 2345, 2347, 2350,
```

2353, 326, 329, 2382, 339, 2396, 2397, 2405, 2407, 364, 2417, 374, 2424, 386, 2440, 393, 405, 407, 2456, 2466, 2474, 2488, 448, 450, 453, 457, 2508, 2510, 2512, 2514, 475, 2537, 2549, 2553, 2560, 2561, 519, 531, 2582, 2588, 551, 2601, 555, 2608, 561, 2609, 2611, 565, 2614, 2615, 566, 2619, 572, 2621, 574, 581, 2631, 2637, 2638, 2644, 2649, 601, 602, 606, 613, 2663, 2690, 644, 2698, 667, 2721, 2723, 675, 2731, 2737, 694, 2742, 2744, 704, 712, 721, 2774, 2776, 2778, 2785, 2793, 754, 766, 769, 2825, 2827, 782, 783, 784, 785, 794, 2843, 2849, 804, 2857, 809, 814, 816, 834, 861, 2909, 863, 2913, 2915, 2919, 2920, 2923, 875, 877, 886, 2934, 2943, 899, 903, 2962, 2970, 2985, 2987, 2989, 942, 3010, 3012, 3022, 974, 3032, 3034, 3038, 991, 3041, 994, 1000, 1003, 1006, 1007, 3056, 1009, 3063, 3075, 3080, 3086, 1042, 1052, 1053, 3104, 1073, 3131, 3137, 3142, 3148, 1101, 1107, 3157, 1110, 3168, 3171, 1125, 3175, 1128, 1129, 1133, 3181, 1134, 1165, 3215, 3218, 3221, 1176, 1181, 3244, 3248, 3255, 3273, 3276, 1229, 3277, 1231, 3279, 3280, 3283, 1239, 1240, 3288, 3293, 3295, 1248, 1249, 1251, 1253, 3311, 1267, 1285, 1293, 3358, 1310, 3359, 3362, 1315, 3368, 3372, 3375, 1336, 3387, 1347, 1348, 1350, 1356, 3410, 1372, 1376, 3435, 1395, 1411, 1412, 3471, 1425, 1430, 1434, 3485, 1440, 3492, 3493, 1445, 3500, 3502, 1455, 1462, 1464, 3527, 1481, 1485, 1489, 3539, 3540, 1508, 1517, 3568, 3569, 3574, 1526, 3583, 3584, 1547, 1554, 1558, 1563, 3617, 1569, 1577, 1595, 1598, 3671, 3682, 1640, 1643, 1644, 1651, 1658, 3713, 1680, 3734, 1687, 1698, 3747, 1705, 3764, 1720, 1721, 1726, 3774, 3775, 1731, 3788, 1747, 1750, 3800, 3804, 3806, 3807, 1778, 1781, 3836, 3842, 1797, 3859, 3875, 1836, 1843, 1848, 3897, 1851, 1852, 3904, 1860, 1863, 1867, 3916, 1872, 1875, 1881, 1897, 3946, 3951, 1908, 3964, 3966, 1922, 3973, 1926, 3974, 3982, 1935, 1937, 3985, 3989, 3990, 1945, 3995, 1949, 3999, 4003, 1957, 1962, 1963, 1965, 4015, 1973, 4024, 1979, 1984, 4036, 1988, 4046, 4061, 4066, 4070, 2027, 4080, 2032, 4088]

Output dari salah satu gen indeks yang terpilih sebagai gen indeks yang dimutasi adalah sebagai berikut.

Sebelum mutasi:

[[146, 147], [148, 149], [150, 151], [152, 153], [154, 155], [156, 157], [158, 159], [160, 161], [162, 163], [164, 165], [166, 167], [168, 169], [170, 171], [172, 173], [174, 175], [176, 177], [178, 179], [180, 181], [182, 183], [184, 185], [186, 187], [188, 189], [190, 191], [192, 193], [194, 195], [196, 197], [198, 199], [200, 201], [202, 203], [0, 1], [2, 3], [4, 5], [6, 7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [63], [64], [65], [66], [67], [68], [69], [70], [71], [72], [73], [74], [75], [76], [77], [78], [79], [80], [81], [82], [83], [84], [85], [86], [87], [88], [89], [90], [91], [92], [93], [94], [95], [96], [97], [98], [99], [100], [101], [102], [103], [104], [105], [106], [107], [108], [109], [110], [111], [112], [113], [114], [115], [116], [117], [118], [119], [120], [121], [122], [123], [124], [125], [126], [127], [128], [129], [130], [131], [132], [133], [134], [135]]

Setelah mutasi:

[[146, 147], [148, 149], [150, 151], [152, 153], [154, 155], [156, 157], [158, 159], [160, 161], [162, 163], [164, 165], [166, 167], [168, 169], [170, 171], [172, 173], [174, 175], [176, 177], [178, 179], [180, 181], [182, 183], [184, 185], [186, 187], [188, 189], [190, 191], [192, 193], [194, 195], [196, 197], [198, 199], [200, 201], [202, 203], [0, 1], [2, 3], [4, 5], [6, 7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [63], [64], [65], [66], [67], [68], [69], [50], [71], [72], [73], [74], [75], [76], [77], [78], [79], [80], [81], [82], [83], [84], [85], [86], [87], [88], [89], [90], [91], [92], [93], [94], [95], [96], [97], [98], [99], [100], [101], [102], [103], [104], [105], [106], [107], [108], [109], [110], [111], [112], [113], [114], [115], [116], [117], [118], [119], [120], [121], [122], [123], [124], [125], [126], [127], [128], [129], [130], [131], [132], [133], [134], [135]]

Setelah semua proses pembaruan populasi telah dilakukan, maka selanjutnya akan dilakukan pengurutan kromosom berdasarkan nilai fitness dari tiap masing-masing kromosom di dalam populasi. Proses pengurutan kromosom diimplementasikan dalam kode program berikut.

```
1.  def sortPopulationByFitnessValue(self, population, column_name):
2.      for i in range(len(population)):
3.          for j in range(i+1, len(population)):
4.              if population[i][column_name] < population[j][column_name]:
5.                  temp = population[j]
6.                  population[j] = population[i]
7.                  population[i] = temp
8.      return population
9.  population = self.sortPopulationByFitnessValue(population, 'fitnessValue')
```

Listing 4.12 Kode Program Pengurutan Kromosom

Dengan Listing 4.12, kromosom dalam populasi telah berhasil diurutkan berdasarkan nilai fitness yang tebesar ke nilai fitness terkecil dari masing-masing kromosom. Berikut adalah penjelasan umum tentang 4.12 berdasarkan baris program.

1. Baris 1: didefinisikan sebuah fungsi **sortPopulationByFitnessValue**
2. Baris 2-3: Merupakan nested loop yang digunakan untuk membandingkan setiap kromosom dalam populasi.
3. Baris 4: Melakukan perbandingan nilai fitness antara dua kromosom yang sedang dibandingkan berdasarkan **fitnessValue**. Jika nilai fitness kromosom pada posisi i lebih kecil dari nilai fitness kromosom pada posisi j, maka dilakukan pertukaran posisi.

4. Baris 5-7: Merupakan proses pertukaran posisi antara dua kromosom yang dibandingkan jika diperlukan.
5. Baris 8: Setelah semua kromosom dibandingkan dan diurutkan, populasi yang telah diurutkan akan dikembalikan.

Output dari Listing 4.12 adalah sebagai berikut.

```
sebelum pengurutan:
[0.0044444444444444444, 0.005076142131979695, 0.0027397260273972603,
0.0021691973969631237, 0.004524886877828055, 0.004081632653061225,
0.005076142131979695, 0.002398081534772182, 0.003194888178913738,
0.0034129692832764505, 0.004524886877828055, 0.0034129692832764505,
0.003663003663003663, 0.009900990099009901, 0.0037174721189591076,
0.004081632653061225, 0.0037174721189591076, 0.0037174721189591076,
0.0055248618784530384, 0.005780346820809248, 0.004524886877828055,
0.004878048780487805, 0.003663003663003663, 0.005780346820809248,
0.005780346820809248, 0.004081632653061225, 0.006535947712418301,
0.003194888178913738, 0.002680965147453083, 0.002680965147453083,
0.0034129692832764505, 0.006711409395973154, 0.0031545741324921135,
0.0037174721189591076, 0.005780346820809248]
setelah pengurutan:
[0.009900990099009901, 0.006711409395973154, 0.006535947712418301,
0.005780346820809248, 0.005780346820809248, 0.005780346820809248,
0.005780346820809248, 0.0055248618784530384, 0.005076142131979695,
0.005076142131979695, 0.004878048780487805, 0.004524886877828055,
```

```
0.004524886877828055,  0.004524886877828055,  0.0044444444444444444,
0.004081632653061225,  0.004081632653061225,  0.004081632653061225,
0.0037174721189591076, 0.0037174721189591076, 0.0037174721189591076,
0.0037174721189591076, 0.003663003663003663,  0.003663003663003663,
0.0034129692832764505, 0.0034129692832764505, 0.0034129692832764505,
0.003194888178913738,  0.003194888178913738,  0.0031545741324921135,
0.0027397260273972603, 0.002680965147453083,  0.002680965147453083,
0.002398081534772182, 0.0021691973969631237]
```

Setelah proses pengurutan kromosom selesai, maka akan dilakukan proses pemangkasan populasi. Hal ini dilakukan, karena setelah melalui proses pembaruan populasi tahap crossover, telah lahir kromosom baru dimana akan menyebabkan jumlah kromosom bertambah. Populasi akan dipangkas hingga jumlah kromosom bernilai sama dengan **numOfChromosome.** Proses pemangkasan populasi diimplementasikan dalam kode program berikut.

```
1. def slicingPopulation(self, population):
2.         ret = []; i = 0
3.         for chromosomes in population:
4.             if i <= self.numOfChromosome-1:
5.                 ret.append(chromosomes)
6.             i+=1
7.         return ret
```

Listing 4.13 Kode Program Pemangkasan Populasi

Dengan Listing 4.13, populasi telah berhasil dipangkas hingga tersisa sama dengan nilai **numOfChromosome.** Kromosom yang tersisa adalah kromosom yang mempunyai nilai fitness yang lebih besar dibanding dengan kromosom yang tersingkirkan berdasarkan proses pengurutan kromosom yang telah dilakukan sebelumnya. Berikut adalah penjelasan umum tentang 4.13 berdasarkan baris program.

1. Baris 1: Deklarasi metode **slicingPopulation**.
2. Baris 2: Inisialisasi variabel **ret** sebagai sebuah list kosong untuk menampung kromosom yang akan dikembalikan dan inisialisasi variabel **i** dengan nilai 0 untuk penghitungan.
3. Baris 3-6: Perulangan **for** untuk setiap **chromosomes** dalam **population**. Jika **i** kurang dari atau sama dengan jumlah kromosom maksimum dikurangi 1 (**self.numOfChromosome - 1**), maka **chromosomes** akan ditambahkan ke dalam list **ret**. Setiap iterasi, nilai **i** akan ditambah 1.
4. Baris 7: Mengembalikan list **ret** yang berisi kromosom-kromosom yang telah dipilih.

Output Listing 4.13 adalah sebagai berikut.

```
Jumlah Kromosom Setelah Crossover dan Belum Dipangkas:  38
Jumlah Kromosom Setelah Crossover dan Sudah Dipangkas:  30
```

4.3.5 Solusi Terbaik

Solusi terbaik merupakan kromosom yang memiliki nilai fitness paling tinggi yang didapatkan dari iterasi yang telah dilakukan. Pada penelitian ini, telah terjadi iterasi sebanyak 15 kali, di mana setiap iterasi dilakukan pembaruan dan evaluasi informasi yang dimiliki tiap kromosom yang ada dalam populasi. Solusi terbaik diimplementasikan dalam kode program berikut.

```
1.  tmpBestChromosome = {
```

```
2.              'fitnessValue':0,
3.              'objectiveValue':0,
4.              'differentDays':None,
5.              'conflictTeachingClass':None,
6.              'sameSlot':None,
7.              'chromosome':population[random.randint(0, self.numOfChromosome-
    1)]['chromosome']}
8.
9.  for generation in range(self.maxGeneration):
10.     population = self.sortPopulationByFitnessValue(population, 'fitnessValue')
11.     population = self.slicingPopulation(population)
12.     bestID = 0
13.     bestChromosome = population[bestID]
14.     if bestChromosome['fitnessValue'] > tmpBestChromosome['fitnessValue']:
15.         tmpBestChromosome = copy.deepcopy(bestChromosome)
```

Listing 4.14 Kode Program Solusi Terbaik

Dengan Listing 4.14, didapatkan solusi terbaik berdasarkan nilai fitness tertinggi dari iterasi terakhir. Berikut adalah penjelasan umum tentang Listing 4.14 berdasarkan baris program.

1. Baris 1-7: Mendefinisikan dictionary untuk menampung kromosom terbaik.
2. Baris 9: Melakukan iterasi sebanyak **maxGeneration.**
3. Baris 10-11: Melakukan pengurutan dan pemangkasan dalam populasi.
4. Baris 12-15: Melakukan pencarian kromosom terbaik berdasarkan **fitnessValue**.

Output Listing 4.14 adalah sebagai berikut.

Generation- 0

0

Generation- 1

0.009900990099009901

Generation- 2

0.009900990099009901

Generation- 3

0.009900990099009901

Generation- 4

0.009900990099009901

Generation- 5

0.009900990099009901

Generation- 6

0.009900990099009901

Generation- 7

0.009900990099009901

Generation- 8

0.009900990099009901

Generation- 9

0.009900990099009901

Generation- 10

0.009900990099009901

Generation- 11

0.009900990099009901

Generation- 12

0.009900990099009901

Generation- 13

0.011235955056179775

Generation- 14

0.011235955056179775

FINALE

Solusi Terbaik: 0.011235955056179775

Kromosom: [[64, 65], [124, 125], [196, 197], [73, 74], [14, 15], [50, 51], [144, 145], [98, 99], [171, 172], [88, 89], [187, 188], [39, 40], [111, 112], [162, 163], [21, 22], [23, 24], [160, 161], [118, 119], [81, 82], [7, 8], [75], [195], [106], [2], [177], [110], [76], [38], [37], [120], [90], [102], [123], [126], [11], [63], [6], [80], [27], [77], [19], [72], [3], [94], [56], [104], [13], [26], [136], [139], [108], [57], [183], [49], [138], [131], [70], [71], [170], [151], [79], [117], [185], [25], [55], [142], [46], [20], [168], [17], [35], [66], [31], [154], [69], [159], [189], [150], [113], [156], [87], [67], [12], [109], [122], [134], [29], [148], [32], [133], [141], [174], [52], [146], [60], [180], [42], [4], [115], [153], [0], [58], [198], [191], [47], [10], [190], [186], [41], [193], [5], [178], [105], [44], [91], [61], [157], [16], [167], [86], [34]]

Hasil Jadwal:

[{'labslot': [64, 65], 'praktikum': ['Alpro', 1, 2, 'Rusydi Umar, M.T., Ph.D.'], 'slot': [['Kamis', 3, 'Basdat'], ['Kamis', 4, 'Basdat']]}, {'labslot': [124, 125], 'praktikum': ['Alpro', 2, 2, 'Rusydi Umar, M.T., Ph.D.'], 'slot': [['Senin', 4, 'Mulmed'], ['Senin', 5, 'Mulmed']]}, {'labslot': [196, 197], 'praktikum': ['Alpro', 3, 2, 'Dr. Ardiansyah, S.T., M.Cs.'], 'slot': [['Sabtu', 3, 'Riset'], ['Sabtu', 4,

'Riset']]}, …, {'labslot': [34], 'praktikum': ['Kripto', 2, 1, 'Eko Aribowo, S.T., M.Kom.'], 'slot': [['Sabtu', 1, 'Jarkom']]}]

Berdasarkan output di atas, dapat dilihat bahwa praktikum **['Alpro', 1, 2, 'Rusydi Umar, M.T., Ph.D.']** mendapat jadwal slot laboratorium **[['Kamis', 3, 'Basdat'], ['Kamis', 4, 'Basdat']]** yang diwakilkan dengan indeks **[64, 65]**, yang artinya praktikum tersebut dijadwalkan pada hari Kamis pukul 12.00 – 15.00 di lab Basis Data. Penjelasan tersebut berlaku untuk mata praktikum seterusnya sesuai dengan hasil solusi yang telah didapatkan. Iterasi dilakukan sebanyak 15 kali dan terdapat nilai fitness di setiap iterasinya. Solusi terbaik diambil dari nilai fitness terbaik dari iterasi terakhir.

4.3.6 Hasil Eksperimen

Eksperimen pada penelitian ini dilakukan untuk menguji perubahan nilai fitness dengan mengatur jumlah kromosom yang berbeda dalam satu populasi. Pengujian akan dilakukan sebanyak dua kali dengan menggunakan data daftar praktikum yang berbeda. Dengan demikian, akan dapat dilihat perbedaan jika jumlah kelas praktikum berbeda pada 2 data daftar praktikum yang digunakan dalam eksperimen ini. Eksperimen dilakukan dengan iterasi yang sesuai dengan Tabel 4.3.

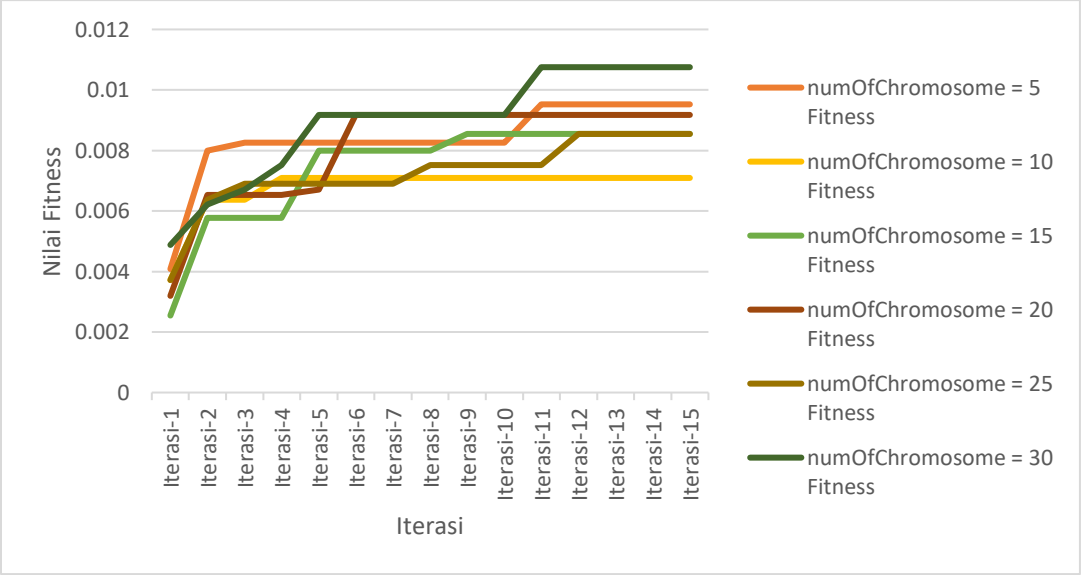
4.3.6.1 Eksperimen Data Daftar Praktikum 1

Data yang digunakan pada eksperimen ini, memiliki kelas praktikum sebanyak 139 kelas praktikum atau bisa juga disebut dalam satu kromosom memiliki 139 gen. Berikut hasil eksperimen pengujian dapat dilihat pada Tabel 4.4 dan Gambar 4.1.

Tabel 4.5 Hasil Eksekusi Data Daftar Praktikum 1

	numOfChromosome = 5		numOfChromosome = 10		numOfChromosome = 15		numOfChromosome = 20		numOfChromosome = 25		numOfChromosome = 30	
	Objektif	Fitness	Objektif	Fitness	Objektif	Fitness	Objektif	Fitness	Objektif	Fitness	Objektif	Fitness

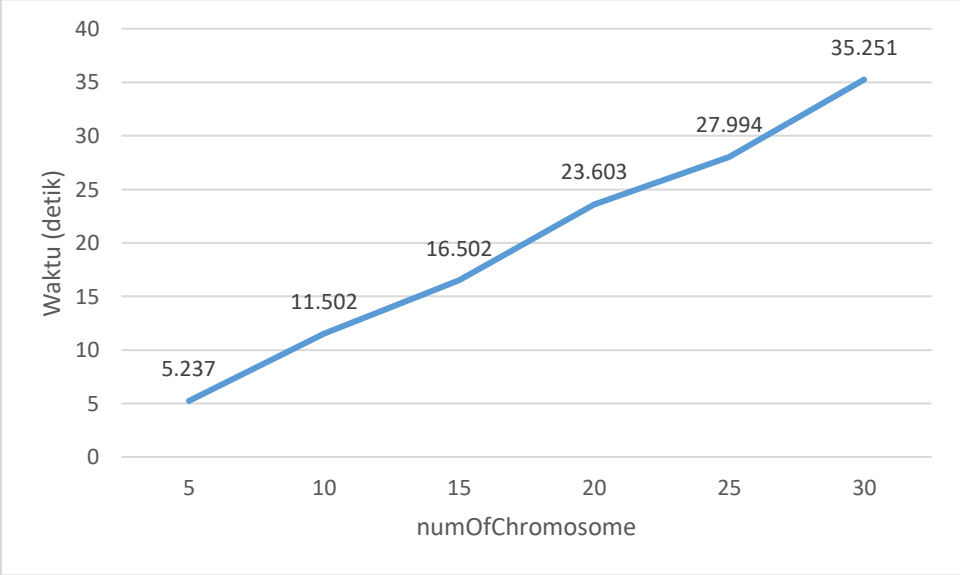
Iter-1	244	0.0041	268	0.0037	392	0.0025	312	0.0032	268	0.0037	204	0.0049
Iter-2	124	0.0080	156	0.0064	172	0.0058	152	0.0065	156	0.0064	160	0.0062
Iter-3	120	0.0083	156	0.0064	172	0.0058	152	0.0065	144	0.0069	148	0.0067
Iter-4	120	0.0083	140	0.0071	172	0.0058	152	0.0065	144	0.0069	132	0.0075
Iter-5	120	0.0083	140	0.0071	124	0.0080	148	0.0067	144	0.0069	108	0.0092
Iter-6	120	0.0083	140	0.0071	124	0.0080	108	0.0092	144	0.0069	108	0.0092
Iter-7	120	0.0083	140	0.0071	124	0.0080	108	0.0092	144	0.0069	108	0.0092
Iter-8	120	0.0083	140	0.0071	124	0.0080	108	0.0092	132	0.0075	108	0.0092
Iter-9	120	0.0083	140	0.0071	116	0.0085	108	0.0092	132	0.0075	108	0.0092
Iter-10	120	0.0083	140	0.0071	116	0.0085	108	0.0092	132	0.0075	108	0.0092
Iter-11	104	0.0095	140	0.0071	116	0.0085	108	0.0092	132	0.0075	92	0.0108
Iter-12	104	0.0095	140	0.0071	116	0.0085	108	0.0092	116	0.0085	92	0.0108
Iter-13	104	0.0095	140	0.0071	116	0.0085	108	0.0092	116	0.0085	92	0.0108
Iter-14	104	0.0095	140	0.0071	116	0.0085	108	0.0092	116	0.0085	92	0.0108
Iter-15	104	0.0095	140	0.0071	116	0.0085	108	0.0092	116	0.0085	92	0.0108
Waktu (detik)	5.237		11.502		16.502		23.603		27.994		35.251	



Gambar 4.1 Grafik Eksperimen Data Daftar Praktikum 1

Berdasarkan Gambar 4.1, dapat dilihat perubahan nilai fitness secara signifikan terjadi mulai dari iterasi kedua, dan selanjutnya grafik nilai fitness dari tiap populasi mengalami kenaikan maupun tidak di setiap iterasinya dan yang pasti tidak ada grafik yang mengalami

penurunan. Dengan demikian, iterasi maksimum dapat diterapkan pada penelitian ini. Dilihat dari perbandingan grafik di tiap populasi dengan jumlah kromosom yang berbeda, populasi dengan jumlah kromosom terbanyak (30 kromosom) memiliki nilai fitness tertinggi, yaitu 0.0108. Pada Tabel 4.4, dapat dilihat juga semakin banyak jumlah kromosom dalam populasi, maka waktu eksekusi juga semakin tinggi.



Gambar 4.2 Grafik Waktu Eksekusi Data Daftar Praktikum 1

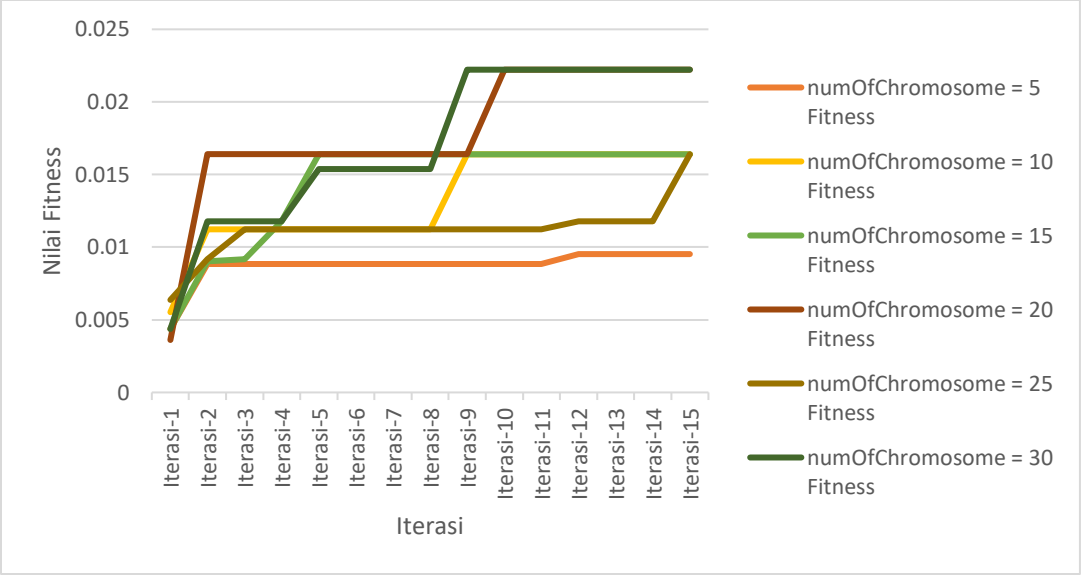
Grafik pada Gambar 4.2 menunjukan perbandingan waktu eksekusi dari tiap populasi yang memiliki jumlah kromosom yang berbeda-beda. Populasi dengan 5 kromosom mengeksekusi program tercepat dengan waktu 5,237 detik. Grafik mengalami kenaikan berdasarkan jumlah kromosom yang semakin bertambah juga. Dengan demikian, dapat dikatakan bahwa semakin banyak jumlah kromosom maka akan semakin lama juga waktu eksekusi program.

4.3.6.2 Eksperimen Data Daftar Praktikum 2

Data yang digunakan pada eksperimen yang kedua ini, memiliki kelas praktikum sebanyak 121 kelas praktikum atau bisa juga disebut dalam satu kromosom memiliki 121 gen. Berikut hasil eksperimen pengujian dapat dilihat pada Tabel 4.5 dan Gambar 4.3.

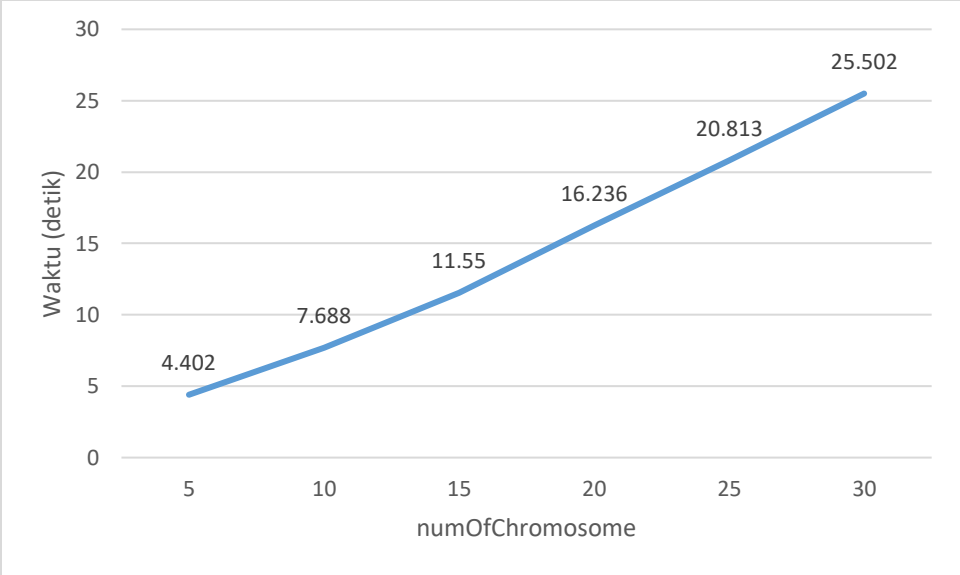
Tabel 4.6 Hasil Eksekusi Data Daftar Praktikum 2

	numOfChromosome = 5		numOfChromosome = 10		numOfChromosome = 15		numOfChromosome = 20		numOfChromosome = 25		numOfChromosome = 30	
	Objektif	Fitness	Objektif	Fitness	Objektif	Fitness	Objektif	Fitness	Objektif	Fitness	Objektif	Fitness
Iter-1	228	0.0044	180	0.0055	228	0.0044	276	0.0036	156	0.0064	228	0.0044
Iter-2	112	0.0088	88	0.0112	110	0.009	60	0.0164	108	0.0092	84	0.0118
Iter-3	112	0.0088	88	0.0112	108	0.0092	60	0.0164	88	0.0112	84	0.0118
Iter-4	112	0.0088	88	0.0112	84	0.0118	60	0.0164	88	0.0112	84	0.0118
Iter-5	112	0.0088	88	0.0112	60	0.0164	60	0.0164	88	0.0112	64	0.0154
Iter-6	112	0.0088	88	0.0112	60	0.0164	60	0.0164	88	0.0112	64	0.0154
Iter-7	112	0.0088	88	0.0112	60	0.0164	60	0.0164	88	0.0112	64	0.0154
Iter-8	112	0.0088	88	0.0112	60	0.0164	60	0.0164	88	0.0112	64	0.0154
Iter-9	112	0.0088	60	0.0164	60	0.0164	60	0.0164	88	0.0112	44	0.0222
Iter-10	112	0.0088	60	0.0164	60	0.0164	44	0.0222	88	0.0112	44	0.0222
Iter-11	112	0.0088	60	0.0164	60	0.0164	44	0.0222	88	0.0112	44	0.0222
Iter-12	104	0.0095	60	0.0164	60	0.0164	44	0.0222	84	0.0118	44	0.0222
Iter-13	104	0.0095	60	0.0164	60	0.0164	44	0.0222	84	0.0118	44	0.0222
Iter-14	104	0.0095	60	0.0164	60	0.0164	44	0.0222	84	0.0118	44	0.0222
Iter-15	104	0.0095	60	0.0164	60	0.0164	44	0.0222	60	0.0164	44	0.0222
Waktu (detik)	4.402		7.688		11.550		16.236		20.813		25.502	



Gambar 4.3 Grafik Eksperimen Data Daftar Praktikum 2

Berdasarkan Gambar 4.3, sama seperti eksperimen sebelumnya dapat dilihat perubahan nilai fitness secara signifikan terjadi mulai dari iterasi kedua, dan selanjutnya grafik nilai fitness dari tiap populasi mengalami kenaikan maupun tidak di setiap iterasinya dan yang pasti tidak ada grafik yang mengalami penurunan. Dengan demikian, iterasi maksimum dapat diterapkan pada penelitian ini. Dilihat dari perbandingan grafik di tiap populasi dengan jumlah kromosom yang berbeda, yang memiliki nilai fitness tertinggi adalah populasi dengan jumlah kromosom sebanyak 20 dan 30, yaitu sebesar 0.02222222. Pada Tabel 4.5, dapat dilihat juga semakin banyak jumlah kromosom dalam populasi, maka waktu eksekusi juga semakin tinggi.



Gambar 4.4 Grafik Waktu Eksekusi Data Daftar Praktikum 2

Sama seperti hasil eksperimen sebelumnya, Gambar 4.4 menunjukkan bahwa waktu eksekusi program mengalami kenaikan mengikuti dengan naiknya jumlah kromosom dalam populasi.

Berdasarkan hasil kedua eksperimen yang telah dilakukan dengan menggunakan daftar praktikum dengan jumlah kelas praktikum atau gen yang berbeda, dapat disimpulkan bahwa semakin sedikit kelas praktikum atau gen yang digunakan maka akan semakin tinggi nilai fitnessnya. Hal ini disebabkan karena semakin sedikit kelas praktikum maka peluang terjadinya bentrok antara jadwal praktikum dengan jadwal mengajar dosen akan semakin kecil yang akan memengaruhi nilai pinalti sebagaimana digunakan sebagai nilai objektif. Waktu eksekusi pada eksperimen kedua juga lebih cepat dibandingkan dengan eksperimen yang pertama.

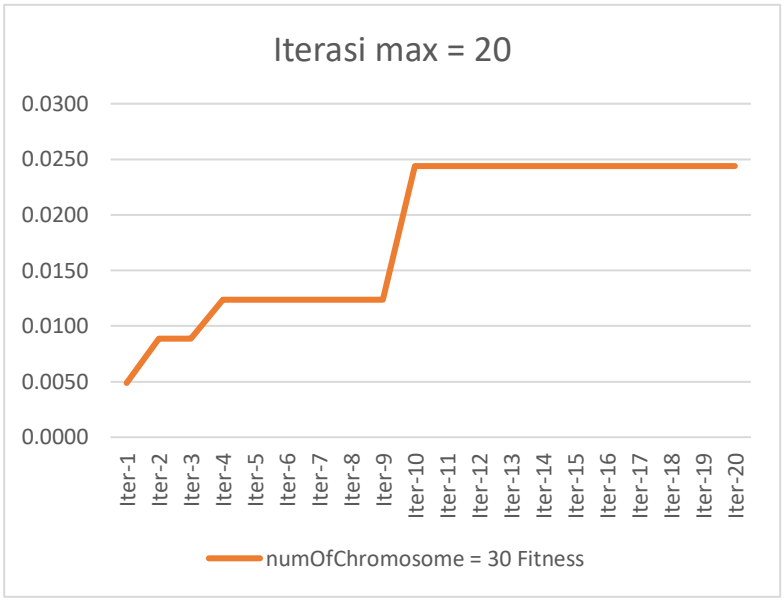
4.3.6.3 Eksperimen Peningkatan Iterasi

Eksperimen ini dilakukan dengan meningkatkan jumlah iterasi menjadi 20, 25, 30 yang bertujuan untuk memastikan iterasi mana yang paling optimal dan efisien untuk digunakan.

Eksperimen ini menggunakan dataset daftar praktikum 2. Berikut ini adalah hasil eksperimen peningkatan iterasi.

Tabel 4.7 Eksperimen Iterasi Maksimal 20

	numOfChromosome = 30	
	Objekt if	**Fitness**
Iter-1	204	0.0049
Iter-2	112	0.0088
Iter-3	112	0.0088
Iter-4	80	0.0123
Iter-5	80	0.0123
Iter-6	80	0.0123
Iter-7	80	0.0123
Iter-8	80	0.0123
Iter-9	80	0.0123
Iter-10	40	0.0244
Iter-11	40	0.0244
Iter-12	40	0.0244
Iter-13	40	0.0244
Iter-14	40	0.0244
Iter-15	40	0.0244
Iter-16	40	0.0244
Iter-17	40	0.0244
Iter-18	40	0.0244
Iter-19	40	0.0244
Iter-20	40	0.0244
Waktu (detik)	34.223	

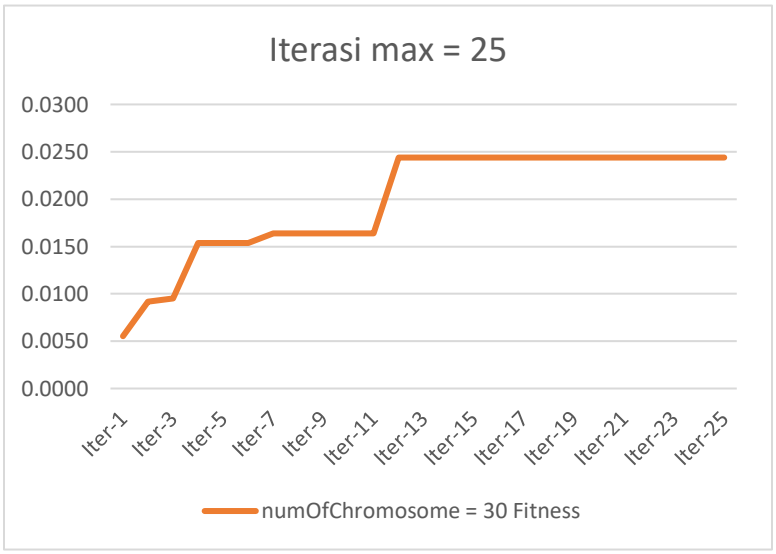


Gambar 4.5 Grafik Iterasi Maksimal 20

Tabel 4.8 Eksperimen Iterasi Maksimal 25

	numOfChromosome = 30	
	Objekt if	**Fitness**
Iter-1	180	0.0055
Iter-2	108	0.0092
Iter-3	104	0.0095
Iter-4	64	0.0154

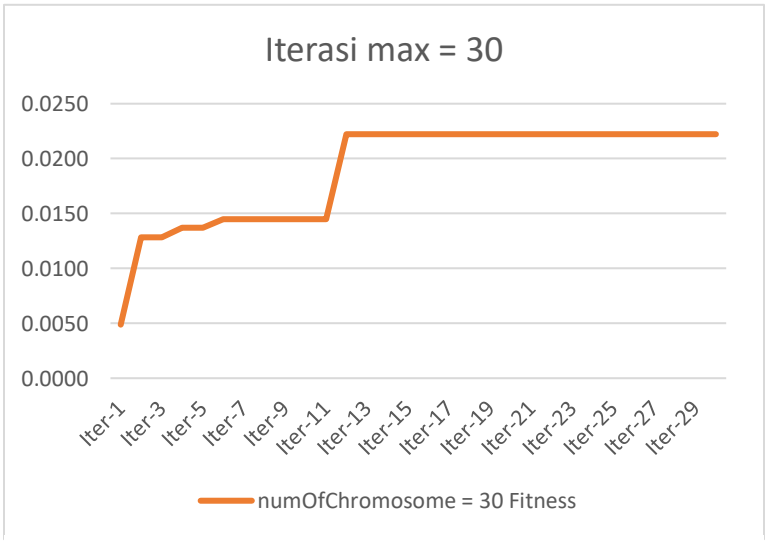
Iter-5	64	0.0154
Iter-6	64	0.0154
Iter-7	60	0.0164
Iter-8	60	0.0164
Iter-9	60	0.0164
Iter-10	60	0.0164
Iter-11	60	0.0164
Iter-12	40	0.0244
Iter-13	40	0.0244
Iter-14	40	0.0244
Iter-15	40	0.0244
Iter-16	40	0.0244
Iter-17	40	0.0244
Iter-18	40	0.0244
Iter-19	40	0.0244
Iter-20	40	0.0244
Iter-21	40	0.0244
Iter-22	40	0.0244
Iter-23	40	0.0244
Iter-24	40	0.0244
Iter-25	40	0.0244
Waktu (detik)	36.573	



Gambar 4.6 Grafik Iterasi Maksimal 25

Tabel 4.9 Eksperimen Iterasi Maksimal 30

	numOfChromosome = 30	
	Objekt if	**Fitness**
Iter-1	204	0.0049
Iter-2	88	0.0128
Iter-3	88	0.0128
Iter-4	72	0.0137
Iter-5	72	0.0137
Iter-6	68	0.0145
Iter-7	68	0.0145
Iter-8	68	0.0145
Iter-9	68	0.0145
Iter-10	68	0.0145
Iter-11	68	0.0145
Iter-12	44	0.0222



Gambar 4.7 Grafik Iterasi Maksimal 30

Iter-13	44	0.0222
Iter-14	44	0.0222
Iter-15	44	0.0222
Iter-16	44	0.0222
Iter-17	44	0.0222
Iter-18	44	0.0222
Iter-19	44	0.0222
Iter-20	44	0.0222
Iter-21	44	0.0222
Iter-22	44	0.0222
Iter-23	44	0.0222
Iter-24	44	0.0222
Iter-25	44	0.0222
Iter-26	44	0.0222
Iter-27	44	0.0222
Iter-28	44	0.0222
Iter-29	44	0.0222
Iter-30	44	0.0222
Waktu (detik)	48.845	

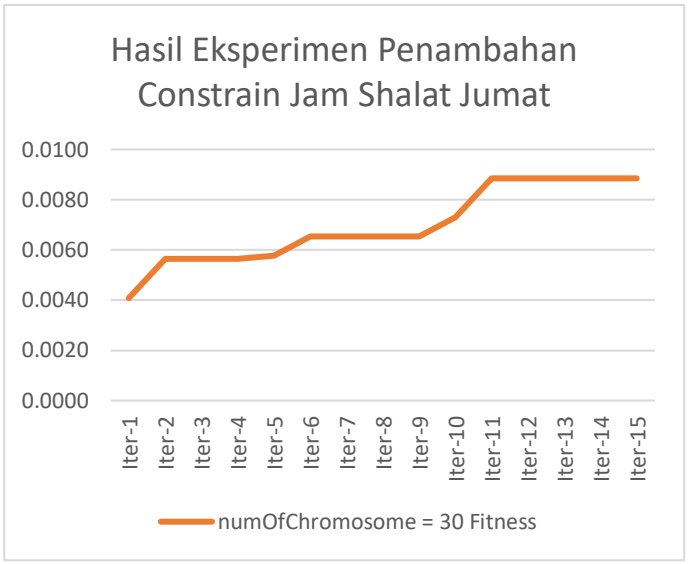
Berdasarkan Hasil eksperimen di atas, dapat dilihat bahwa, perubahan nilai fitness tidak terjadi setelah melalui iterasi ke-15.Dengan demikian, dapat disimpulkan bahwa penambahan iterasi menjadi 20, 25, maupun 30 tidak diperlukan dan cukup sampai iterasi ke-15 saja, mengetahui dengan menggunakan 15 iterasi waktu eksekusi program juga lebih cepat.

4.3.6.4 Eksperiman Penambahan Constrain Jam Shalat Jumat

Eksperimen ini dilakukan dengan menghapus slot laboratorium yang bertabrakan dengan waktu shalat jumat. Eksperimen ini menggunakan dataset daftar praktikum 1. Berikut ini adalah hasil eksperimennya.

Tabel 4.10 Eksperimen Constrain Jam Shalat Jumat

	numOfChromosome = 30	
	Objektif	**Fitness**
Iter-1	244	0.0041
Iter-2	176	0.0056
Iter-3	176	0.0056
Iter-4	176	0.0056
Iter-5	172	0.0058
Iter-6	152	0.0065
Iter-7	152	0.0065
Iter-8	152	0.0065
Iter-9	152	0.0065
Iter-10	136	0.0073
Iter-11	112	0.0088
Iter-12	112	0.0088
Iter-13	112	0.0088
Iter-14	112	0.0088
Iter-15	112	0.0088



Gambar 4.8 Grafik Penambahan Constrain

Berdasarkan hasil eksperimen di atas, dapat disimpulkan bahwa dengan menghapus slot laboratorium yang bentrok dengan jadwal shalat jumat memengaruhi nilai fitness. Hal ini disebabkan, dengan adanya penghapusan slot laboratorium, maka akan berkurang ketersediaan slot laboratorium yang akan digunakan sehingga akan relative mengurangi nilai fungsi fitness.

4.4 Penerapan Algoritma Genetika Pada Software

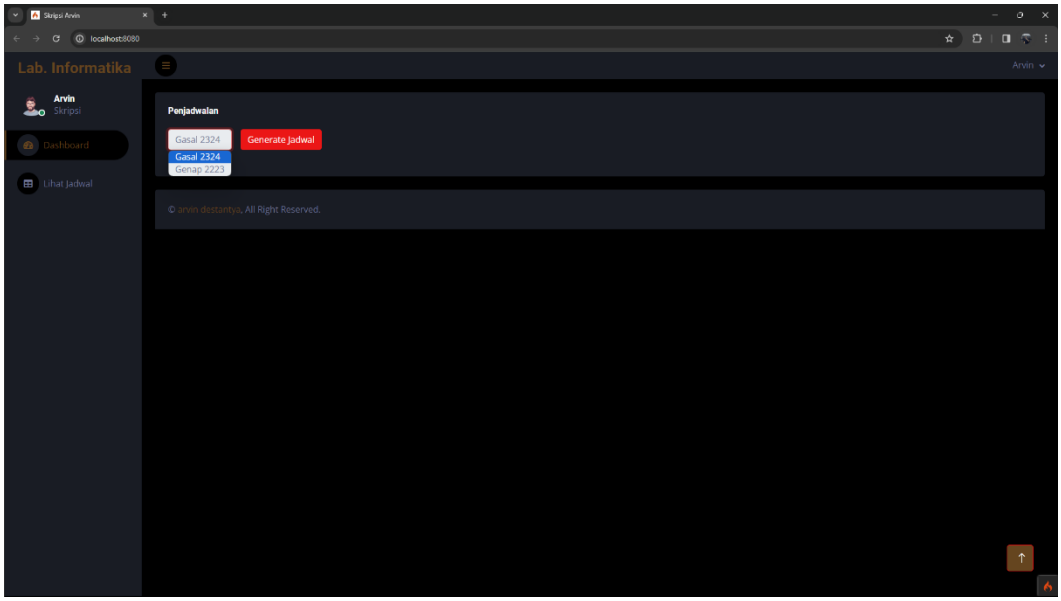
4.4.1 Hasil Spesifikasi Fungsional

Pada penelitian ini, program algoritma genetika yang telah dieksekusi akan dikeluarkan hasilnya melalui sebuah aplikasi berbasis website. Aplikasi berbasis website ini akan menyediakan fitur dimana user dapat memilih dataset praktikum yang akan digunakan dan menjalankannya agar menghasilkan jadwal praktikum berbentuk tabel di dalam website. Implementasi aplikasi dibagi menjadi 2 bagian.

1. Aplikasi website

   a. Memilih dataset dan generate

      User dapat memilih dataset yang akan digunakan dan menjalankan program algoritma genetika dengan klik “generate jadwal”.



Gambar 4.9 Halaman Dashboard Website

   b. Hasil Jadwal Praktikum

      Setelah user melakukan generate jadwal, pilihan dataset yang dipilih oleh user akan dikirimkan ke API untuk menjalankan program algoritma genetika. Setelah

program algoritma genetika selesai dijalankan, API akan mengirimkan data jadwal praktikum ke website.

Gambar 4.10 Halaman Jadwal Praktikum

Gambar 4.11 Halaman Jadwal Praktikum 2

2. API

Dataset yang dipilih oleh pengguna akan dialirkan melalui API untuk kemudian diolah dengan menggunakan algoritma genetika. API yang digunakan dalam website ini dibangun dengan menggunakan bantuan framework dari python, yaitu Flask.

```
1. from flask import Flask, jsonify
```

```
2.  from algengenap import *
3.  from algengasal import *
4.  import mysql.connector
5.  import json
6.
7.  app = Flask(__name__)
8.  start_time = None
9.  parameterss = {
10.     'numOfChromosome': 30,
11.     'crossoverRate': 0.25,
12.     'mutationRate': 0.1,
13.     'maxGen': 15,
14.     'penalties':{
15.         'differentDay':20,
16.         'teachingScheduleConflict':24}
17. }
18. @app.route('/api/build/<int:generate>/', methods=['GET'])
19. def build(generate):
20.     if generate == 1 :
21.         a = GALabTimeTables_1(parameterss)
22.         b = a.generateLabTimeTables()
23.         c = a.dosen()
24.
25.         delete_data_to_mysql()
26.         altab_data_to_mysql()
27.
```

```
28.     i = 0
29.     for i in range(139):
30.         labslot = json.dumps(b[i]['labslot'])
31.         praktikum = json.dumps(b[i]['praktikum'])
32.         slot_json = json.dumps(b[i]['slot'])
33.         jam_json = json.dumps(b[i]['jam'])
34.         penalti = b[i]['penalti']
35.
36.         # Data yang akan disisipkan ke MySQL
37.         data_to_insert = {
38.             'kolom1': labslot,
39.             'kolom2': praktikum,
40.             'kolom3': slot_json,
41.             'kolom4': jam_json,
42.             'kolom5': penalti
43.         }
44.         insert_data_to_mysql(data_to_insert)
45.
46.     j = 0
47.     for j in range(164):
48.         d_hari = c[j]['hari']
49.         d_matkul = c[j]['matkul']
50.         d_nama = c[j]['nama']
51.         d_jam = json.dumps(c[j]['jam'])
52.
53.         # Data yang akan disisipkan ke MySQL
```

```
54.             dosen_to_insert = {
55.                 'kolom1': d_hari,
56.                 'kolom2': d_matkul,
57.                 'kolom3': d_nama,
58.                 'kolom4': d_jam,
59.             }
60.             insert_dosen_data_to_mysql(dosen_to_insert)
61.
62.         data_set = {
63.             'tes' : b,
64.             'jadwaldosen' : c
65.         }
66.         return jsonify(data_set)
67.
68.     elif generate == 2 :
69.         a = GALabTimeTables_2(91enalty91rs)
70.         b = a.generateLabTimeTables()
71.         c = a.dosen()
72.
73.         delete_data_to_mysql()
74.         altab_data_to_mysql()
75.
76.         i = 0
77.         for i in range(121):
78.             labslot = json.dumps(b[i]['labslot'])
79.             praktikum = json.dumps(b[i]['praktikum'])
```

```
80.         slot_json = json.dumps(b[i]['slot'])
81.         jam_json = json.dumps(b[i]['jam'])
82.         92enalty = b[i]['penalti']
83.
84.         # Data yang akan disisipkan ke MySQL
85.         data_to_insert = {
86.             'kolom1': labslot,
87.             'kolom2': praktikum,
88.             'kolom3': slot_json,
89.             'kolom4': jam_json,
90.             'kolom5': 92enalty
91.         }
92.         insert_data_to_mysql(data_to_insert)
93.
94.     j = 0
95.     for j in range(164):
96.         d_hari = c[j]['hari']
97.         d_matkul = c[j]['matkul']
98.         d_nama = c[j]['nama']
99.         d_jam = json.dumps(c[j]['jam'])
100.
101.            # Data yang akan disisipkan ke MySQL
102.            dosen_to_insert = {
103.                'kolom1': d_hari,
104.                'kolom2': d_matkul,
105.                'kolom3': d_nama,
```

```
106.                 'kolom4': d_jam,
107.             }
108.             insert_dosen_data_to_mysql(dosen_to_insert)
109.
110.         data_set = {
111.             'tes' : b,
112.             'jadwaldosen' : c
113.         }
114.         return jsonify(data_set)
115.
116. @app.route('/api/end_jadwal/<int:jadwal>/', methods=['GET'])
117. def end_jadwal(jadwal):
118.     return get_jadwal()
119.
120. if __name__ == '__main__':
121.     # app.run(debug=True)
122.     app.run(host='0.0.0.0', port=105)
```

Listing 4.15 Kode Program API

Listing 4.15 merupakan kode program API untuk website yang mengimplementasikan algoritma genetika. Pilihan dataset yang dipilih oleh user akan dikirimkan ke API untuk dieksekusi oleh algoritma genetika.

Setelah proses algoritma genetika selesai dan menghasilkan solusi terbaik, data jadwal praktikum akan dikirim dan ditampung ke dalam database. Data jadwal praktikum yang telah ditampung ke database ini kemudian akan dikirimkan kembali ke website melalui API dalam format JSON.

Setelah data jadwal praktikum dikirimkan oleh API, dan akan diterima oleh website dengan menggunakan Javascript. Selanjutnya, data jadwal praktikum akan ditampilkan kepada pengguna sebagai hasil penjadwalan praktikum menggunakan algoritma genetika.

4.4.2 Hasil Pengujian Black-Box

Pengujian black box akan dilakukan untuk mengevaluasi fungsionalitas dan kinerja aplikasi dari perspektif pengguna, dengan fokus pada input dan output yang dihasilkan. Hal ini bertujuan untuk memastikan bahwa aplikasi dapat beroperasi sesuai dengan harapan dan memenuhi kebutuhan pengguna dengan baik.

Tabel 4.11 Pengujian Black-Box

No	Skenario Pengujian	Hasil yang diharapkan	Hasil pengujian	Kesimpulan
Halaman Dashboard				
1	1. Masuk ke halaman dashboard 2. Memilih menu dataset ‘Gasal 2324’ 3. Klik tombol ‘Generate Jadwal’	Aplikasi akan menerima pilihan dataset dan akan menjalankan program algoritma.	Aplikasi berhasil menerima pilihan dataset dan akan menjalankan program algoritma.	Sesuai
2	1. Masuk ke halaman dashboard 2. Memilih menu dataset ‘Genap 2223’ 3. Klik tombol ‘Generate Jadwal’	Aplikasi akan menerima pilihan dataset dan akan menjalankan program algoritma.	Aplikasi berhasil menerima pilihan dataset dan akan menjalankan program algoritma.	Sesuai
3	1. Masuk ke halaman dashboard 2. Klik menu sidebar ‘Lihat Jadwal’	Aplikasi akan beralih ke halaman lihat jadwal	Aplikasi berhasil beralih ke halaman lihat jadwal	Sesuai
Halaman Lihat Jadwal				
7	1. Masuk ke halaman lihat jadwal 2. Pilih menu ‘Tampilkan semua lab’ 3. Klik tombol ‘Lihat Jadwal’	Aplikasi akan menampilkan jadwal praktikum pada semua laboratorium	Aplikasi berhasil menampilkan jadwal praktikum pada semua	Sesuai

		dalam bentuk tabel	laboratorium dalam bentuk tabel	
8	1. Masuk ke halaman lihat jadwal 2. Pilih menu 'Jaringan Komputer' 3. Klik tombol 'Lihat Jadwal'	Aplikasi akan menampilkan jadwal praktikum pada laboratorium jaringan komputer dalam bentuk tabel	Aplikasi berhasil menampilkan jadwal praktikum pada laboratorium jaringan komputer dalam bentuk tabel	Sesuai
9	1. Masuk ke halaman lihat jadwal 2. Pilih menu 'Basis Data' 3. Klik tombol 'Lihat Jadwal'	Aplikasi akan menampilkan jadwal praktikum pada laboratorium basis data dalam bentuk tabel	Aplikasi berhasil menampilkan jadwal praktikum pada laboratorium basis data dalam bentuk tabel	Sesuai
10	1. Masuk ke halaman lihat jadwal 2. Pilih menu 'Komputasi Dasar' 3. Klik tombol 'Lihat Jadwal'	Aplikasi akan menampilkan jadwal praktikum pada laboratorium komputasi dasar dalam bentuk tabel	Aplikasi berhasil menampilkan jadwal praktikum pada laboratorium komputasi dasar dalam bentuk tabel	Sesuai
11	1. Masuk ke halaman lihat jadwal 2. Pilih menu 'Multimedia' 3. Klik tombol 'Lihat Jadwal'	Aplikasi akan menampilkan jadwal praktikum pada laboratorium multimedia dalam bentuk tabel	Aplikasi berhasil menampilkan jadwal praktikum pada laboratorium multimedia dalam bentuk tabel	Sesuai
12	1. Masuk ke halaman lihat jadwal 2. Pilih menu 'Riset' 3. Klik tombol 'Lihat Jadwal'	Aplikasi akan menampilkan jadwal praktikum pada laboratorium riset dalam bentuk tabel	Aplikasi berhasil menampilkan jadwal praktikum pada laboratorium	Sesuai

			riset dalam bentuk tabel	
13	1. Masuk ke halaman lihat jadwal 2. Klik menu sidebar 'dashboard'	Aplikasi akan beralih ke halaman dashboard	Aplikasi berhasil beralih ke halaman dashboard	Sesuai

4.5 Hasil Uji Statistik

Pada penelitian ini, uji statistik akan dilakukan dengan menggunakan uji Friedman sebagai metode statistik utama untuk menganalisis data yang telah dikumpulkan. Metode ini dipilih karena kemampuannya dalam membandingkan pengaruh beberapa perlakuan atau kondisi terhadap variabel yang diamati, sehingga dapat memberikan pemahaman yang mendalam tentang hubungan antara faktor-faktor yang diteliti, dengan tujuan untuk menentukan apakah terdapat perbedaan signifikan antara kelompok perlakuan yang berbeda dalam eksperimen yang telah dilakukan.

Sampel data yang digunakan adalah data yang diperoleh dari hasil eksperimen yang telah dilakukan sebelumnya, dalam hal ini mengambil kolom yang berisi nilai fitness pada Tabel 4.4 dan Tabel 4.5. Berikut adalah sampel data yang akan digunakan.

Tabel 4.12 Sampel Data Uji Statistik 1

	numOfChromosome = 5	**numOfChromosome = 10**	**numOfChromosome = 15**	**numOfChromosome = 20**	**numOfChromosome = 25**	**numOfChromosome = 30**
	Fitness	**Fitness**	**Fitness**	**Fitness**	**Fitness**	**Fitness**
Iter-1	0.0041	0.0037	0.0025	0.0032	0.0037	0.0049
Iter-2	0.0080	0.0064	0.0058	0.0065	0.0064	0.0062
Iter-3	0.0083	0.0064	0.0058	0.0065	0.0069	0.0067
Iter-4	0.0083	0.0071	0.0058	0.0065	0.0069	0.0075
Iter-5	0.0083	0.0071	0.0080	0.0067	0.0069	0.0092
Iter-6	0.0083	0.0071	0.0080	0.0092	0.0069	0.0092
Iter-7	0.0083	0.0071	0.0080	0.0092	0.0069	0.0092
Iter-8	0.0083	0.0071	0.0080	0.0092	0.0075	0.0092
Iter-9	0.0083	0.0071	0.0085	0.0092	0.0075	0.0092
Iter-10	0.0083	0.0071	0.0085	0.0092	0.0075	0.0092

Iter-11	0.0095	0.0071	0.0085	0.0092	0.0075	0.0108
Iter-12	0.0095	0.0071	0.0085	0.0092	0.0085	0.0108
Iter-13	0.0095	0.0071	0.0085	0.0092	0.0085	0.0108
Iter-14	0.0095	0.0071	0.0085	0.0092	0.0085	0.0108
Iter-15	0.0095	0.0071	0.0085	0.0092	0.0085	0.0108

Tabel 4.13 Sampel Data Uji Statistik 2

	numOfChromosome = 5	**numOfChromosome = 10**	**numOfChromosome = 15**	**numOfChromosome = 20**	**numOfChromosome = 25**	**numOfChromosome = 30**
	Fitness	**Fitness**	**Fitness**	**Fitness**	**Fitness**	**Fitness**
Iter-1	0.0044	0.0055	0.0044	0.0036	0.0064	0.0044
Iter-2	0.0088	0.0112	0.0090	0.0164	0.0092	0.0118
Iter-3	0.0088	0.0112	0.0092	0.0164	0.0112	0.0118
Iter-4	0.0088	0.0112	0.0118	0.0164	0.0112	0.0118
Iter-5	0.0088	0.0112	0.0164	0.0164	0.0112	0.0154
Iter-6	0.0088	0.0112	0.0164	0.0164	0.0112	0.0154
Iter-7	0.0088	0.0112	0.0164	0.0164	0.0112	0.0154
Iter-8	0.0088	0.0112	0.0164	0.0164	0.0112	0.0154
Iter-9	0.0088	0.0164	0.0164	0.0164	0.0112	0.0222
Iter-10	0.0088	0.0164	0.0164	0.0222	0.0112	0.0222
Iter-11	0.0088	0.0164	0.0164	0.0222	0.0112	0.0222
Iter-12	0.0095	0.0164	0.0164	0.0222	0.0118	0.0222
Iter-13	0.0095	0.0164	0.0164	0.0222	0.0118	0.0222
Iter-14	0.0095	0.0164	0.0164	0.0222	0.0118	0.0222
Iter-15	0.0095	0.0164	0.0164	0.0222	0.0164	0.0222

Setelah mengumpulkan data-data, akan menentukan hipotesis awal (H_0) dan tingkat signifikansi (α) di mana pada penelitian ini, H_0 adalah tidak terdapat perbedaan yang signifikan antar perlakuan eksperimen yang telah dilakukan dan tingkat signifikansi 5%. Langkah selanjutnya adalah melakukan pemeringkatan nilai data dari 1 (peringkat terendah) hingga 6 (peringkat tertinggi) dan menjumlahkannya. Berikut adalah hasil pemeringkatannya.

Tabel 4.14 Hasil Pemeringkatan Data 1

	numOfChromosome = 5		numOfChromosome = 10		numOfChromosome = 15		numOfChromosome = 20		numOfChromosome = 25		numOfChromosome = 30	
	Fitness	Peringkat	Fitness	Peringkat	Fitness	Peringkat	Fitness	Peringkat	Fitness	Peringkat	Fitness	Peringkat
Iter-1	0.0041	5	0.0037	3.5	0.0025	1	0.0032	2	0.0037	3.5	0.0049	6
Iter-2	0.0080	6	0.0064	2.5	0.0058	1	0.0065	4	0.0064	2.5	0.0062	5
Iter-3	0.0083	6	0.0064	2	0.0058	1	0.0065	3	0.0069	5	0.0067	4
Iter-4	0.0083	6	0.0071	4	0.0058	1	0.0065	2	0.0069	3	0.0075	5
Iter-5	0.0083	5	0.0071	3	0.0080	4	0.0067	1	0.0069	2	0.0092	6
Iter-6	0.0083	4	0.0071	2	0.0080	3	0.0092	5.5	0.0069	1	0.0092	5.5
Iter-7	0.0083	4	0.0071	2	0.0080	3	0.0092	5.5	0.0069	1	0.0092	5.5
Iter-8	0.0083	4	0.0071	1	0.0080	3	0.0092	5.5	0.0075	2	0.0092	5.5
Iter-9	0.0083	3	0.0071	1	0.0085	4	0.0092	5.5	0.0075	2	0.0092	5.5
Iter-10	0.0083	3	0.0071	1	0.0085	4	0.0092	5.5	0.0075	2	0.0092	5.5
Iter-11	0.0095	5	0.0071	1	0.0085	3	0.0092	4	0.0075	2	0.0108	6
Iter-12	0.0095	5	0.0071	1	0.0085	2.5	0.0092	4	0.0085	2.5	0.0108	6
Iter-13	0.0095	5	0.0071	1	0.0085	2.5	0.0092	4	0.0085	2.5	0.0108	6
Iter-14	0.0095	5	0.0071	1	0.0085	2.5	0.0092	4	0.0085	2.5	0.0108	6
Iter-15	0.0095	5	0.0071	1	0.0085	2.5	0.0092	4	0.0085	2.5	0.0108	6
Total		71		27		38		59.5		36		83.5
Rerata		5		1.8		2.5		4		2.4		5.6

Tabel 4.15 Hasil Pemeringkatan Data 2

	numOfChromosome = 5		numOfChromosome = 10		numOfChromosome = 15		numOfChromosome = 20		numOfChromosome = 25		numOfChromosome = 30	
	Fitness	Peringkat	Fitness	Peringkat	Fitness	Peringkat	Fitness	Peringkat	Fitness	Peringkat	Fitness	Peringkat
Iter-1	0.0044	3	0.0055	5	0.0044	3	0.0036	1	0.0064	6	0.0044	3
Iter-2	0.0088	1	0.0112	4	0.0090	2	0.0164	6	0.0092	3	0.0118	5
Iter-3	0.0088	1	0.0112	3.5	0.0092	2	0.0164	6	0.0112	3.5	0.0118	5
Iter-4	0.0088	1	0.0112	2.5	0.0118	5	0.0164	6	0.0112	2.5	0.0118	5
Iter-5	0.0088	1	0.0112	2.5	0.0164	5.5	0.0164	5.5	0.0112	2.5	0.0154	4
Iter-6	0.0088	1	0.0112	2.5	0.0164	5.5	0.0164	5.5	0.0112	2.5	0.0154	4
Iter-7	0.0088	1	0.0112	2.5	0.0164	5.5	0.0164	5.5	0.0112	2.5	0.0154	4
Iter-8	0.0088	1	0.0112	2.5	0.0164	5.5	0.0164	5.5	0.0112	2.5	0.0154	4
Iter-9	0.0088	1	0.0164	4	0.0164	4	0.0164	4	0.0112	2	0.0222	6
Iter-10	0.0088	1	0.0164	3.5	0.0164	3.5	0.0222	5.5	0.0112	2	0.0222	5.5
Iter-11	0.0088	1	0.0164	3.5	0.0164	3.5	0.0222	5.5	0.0112	2	0.0222	5.5

Iter-12	0.0095	1	0.0164	3.5	0.0164	3.5	0.0222	5.5	0.0118	2	0.0222	5.5
Iter-13	0.0095	1	0.0164	3.5	0.0164	3.5	0.0222	5.5	0.0118	2	0.0222	5.5
Iter-14	0.0095	1	0.0164	3.5	0.0164	3.5	0.0222	5.5	0.0118	2	0.0222	5.5
Iter-15	0.0095	1	0.0164	3	0.0164	3	0.0222	5.5	0.0164	3	0.0222	5.5
Total		17		50		58		78		40		72.5
Rerata		1		3.3		3.9		5		2.7		4.8

Telah didapatkan hasil pemeringkatan dan penjumlahan nilai peringkat data dari masing-masing tabel data uji. Langkah selanjutnya adalah menghitung nilai uji statistik dengan menggunakan rumus friedman sebagai berikut.

$x_r^2 = \frac{12}{n(k(k+1))} \left[\sum R_i^2\right] - 3n(k+1)$, di mana

x_r^2 = nilai uji statistik friedman

n = jumlah perulangan

k = banyaknya perlakuan

$\sum R_i^2 =$ jumlah pemeringkatan

Berdasarkan Tabel 4.9, maka hasil perhitungan nilai uji statistik menggunakan rumus friedman adalah sebagai berikut.

$$x_r^2 = \frac{12}{15(6(6+1))} \left[(71^2) + (27^2) + (38^2) + (59{,}5^2) + (36^2) + (83.5^2)\right] - 3 \times 15(6+1)$$

$$x_r^2 = \frac{12}{15(42)}\ 19022.5 - 315$$

$$x_r^2 = \frac{228270}{630} - 315$$

$$x_r^2 = 47{,}333$$

Dengan demikian, telah didapatkan hasil uji friedman dari data Tabel 4.7, yaitu memiliki nilai sebesar 47,333. Selanjutnya, akan dilanjutkan perhitungan nilai uji statistik friedman berdasarkan tabel 4.10.

$$x_r^2 = \frac{12}{15(6(6+1))}\left[(17^2)+(50^2)+(58^2)+(78^2)+(40^2)+(72.5^2)\right] - 3 \times 15(6+1)$$

$$x_r^2 = \frac{12}{15(42)}\ 19093.25 - 315$$

$$x_r^2 = \frac{229119}{630} - 315$$

$$x_r^2 = 48{,}68$$

Setelah mendapatkan nilai uji statistik friedman dari masing-masing sampel data, selanjutnya akan menentukan derajat kebebasan (df), di mana $df = k - 1$. Jadi, $df = 5$. Kemudian, akan menggunakan tabel distribusi chi-square untuk menentukan nilai p dari nilai uji statistik yang telah dihitung.

Tabel 4.16 Chi-Square

df	Taraf Signifikansi (α)		
	10%	5%	1%
1	2.706	3.481	6.635
2	3.605	5.591	9.21
3	6.251	7.815	11.341
4	7.779	9.488	13.277
5	9.236	11.07	15.086

Berdasarkan derajat kebebasan ($df = 5$.) dan tingkat signifikansi (α = 5%) yang telah ditentukan sebelumnya, maka didapatkan nilai p = 11,07 yang dilihat dari Tabel 4.11. Dikarenakan

nilai uji statistik friedman dari kedua sampel data (x_r^2= 47,33 dan x_r^2 = 48,68) lebih besar dibandingkan dengan nilai p (11,07), maka H_0 ditolak. Dengan H_0 ditolak, dapat disimpulkan bahwa terdapat perbedaan yang signifikan antar perlakuan eksperimen yang telah dilakukan.

BAB 5.
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Penelitian ini bertujuan untuk mengoptimalkan mekanisme penjadwalan praktikum di Laboratorium Prodi S1 Informatika UAD. Untuk mencapai tujuan tersebut, penelitian ini telah mengembangkan sebuah aplikasi penjadwalan praktikum berbasis web dengan menggunakan algoritma genetika. Berdasarkan hasil pengembangan dan analisis eksperimen, dapat disimpulkan bahwa algoritma genetika berhasil meningkatkan efisiensi penjadwalan praktikum yang sebelumnya dilakukan secara konvensional menjadi otomatis. Hasil yang diperoleh menunjukkan bahwa:

1. Semakin sedikit data kelas praktikum yang digunakan, akan mendapatkan solusi dengan nilai objektif yang kecil dan nilai fitness yang besar. Artinya, semakin besar nilai fitness akan membuat solusi lebih optimal.
2. Jumlah kromosom memengaruhi besarnya nilai fitness dan juga waktu eksekusi program. Semakin besar jumlah kromosom, akan semakin lama waktu eksekusi program, begitu juga sebaliknya.

Meskipun algoritma genetika telah berhasil diaplikasikan dalam penelitian ini dan memberikan hasil yang optimal, tetapi muncul pertanyaan mengenai hasil yang mungkin didapat jika menggunakan algoritme metaheuristik lainnya, seperti algoritme ACO, Firefly, PSO, dan sebagainya. Hal ini penting mengingat algoritma-algoritma tersebut telah menunjukkan kinerja yang baik dalam berbagai bidang lain.

Selain itu, berdasarkan kesimpulan yang ditarik, pengelola laboratorium S1 Informatika UAD dapat mempertimbangkan penerapan sistem dan aplikasi penjadwalan ini untuk menyusun jadwal praktikum di masa mendatang.

5.2 Saran

Adapun saran yang diajukan untuk penelitian selanjutnya adalah sebagai berikut:

1. Pada penelitan selanjutnya, perlu dilakukan dengan menggunakan algoritma optimasi lain yang relatif baru agar dapat mengetahui lebih dalam bagaimana kinerja algoritma metaheuristik pada penjadwalan praktikum.
2. Aplikasi penjadwalan praktikum yang dikembangkan dalam penelitian ini masih terbatas dalam cakupannya dan hanya cocok untuk keperluan praktikum di Prodi S1 Informatika UAD. Untuk memperluas jangkauan penggunaan, penelitian selanjutnya perlu fokus pada peningkatan skalabilitas aplikasi agar dapat digunakan di berbagai laboratorium antar Program Studi di lingkungan UAD.

DAFTAR PUSTAKA

[1] O. Sayudias, L. Fujiyanti, and M. Setya Pratama, “Aplikasi Sistem Informasi Penjadwalan Laboratorium (Studi Kasus Laboratorium TRPL),” in *Prosiding Seminar Nasional Inovasi Teknologi Terapan*, 2022, pp. 381–386.

[2] B. V. Christioko, S. Asmiaatun, and Susanto, “Penjadwalan Kegiatan Praktikum Menggunakan Algoritma Genetika (Studi Kasus: Jurusan Teknologi Informasi Universitas Semarang),” *Journal Voice of Informatics*, vol. 11, no. 1, pp. 25–34, 2022.

[3] M. Syawal, P. L. Lokapitasari, and A. Rachman Manga, “Implementasi Algoritma Genetika Untuk Penjadwalan Laboratotium Fakultas Ilmu Komputer Universitas Muslim Indonesia,” *Indonesian Journal of Data and Science (IJODAS)*, vol. 2, no. 1, pp. 29–37, 2021.

[4] I. M. B. Adnyana, “Implementasi Algoritma Genetika untuk Penjadwalan Asisten Dosen di STIKOM Bali,” *Jurnal Sistem dan Informatika*, vol. 12, no. 2, pp. 166–173, 2018.

[5] M. Alda, “Aplikasi Penjadwalan Laboratorium Berbasis Android Pada SMK Bina Satria,” *Komputika : Jurnal Sistem Komputer*, vol. 11, no. 2, pp. 147–156, Feb. 2022, doi: 10.34010/komputika.v11i2.6011.

[6] Y. N. Firmansyah, N. Kasan, and M. Lestandy, “Sistem Informasi Manajemen Laboratorium Teknik Elektro UMM Berbasis CodeIgniter Menggunakan Algoritma Genetika,” in *Seminar Nasional Teknologi dan Rekayasa (SENTRA)*, Malang, 2020, pp. 2527–6050.

[7] E. Herjanto, *Manajemen Operasi (Edisi 3)*, 3rd ed. Jakarta: PT Grasindo, 2007.

[8] Suyanto, *Evolutionary Computation: Komputasi Berbasis "Evolusi" dan "Genetika."* Bandung: Informatika, Bandung, Indonesia, 2008.

[9] E. Na'imah and S. Rohmaniah, "Analisis Data Produksi Ikan Konsumsi Menggunakan Uji Friedman," *UJMC (Unisda Journal of Mathematics and Computer Science)*, vol. 6, no. 01, Jun. 2020, doi: https://doi.org/10.52166/ujmc.v6i01.2384.

[10] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft comput*, vol. 9, pp. 3–12, Oct. 2005, doi: 10.1007/s00500-003-0328-5.