BAB II. TINJAUAN PUSTAKA

2.1. Sistem Informasi

Sistem adalah kumpulan prosedur atau tahapan yang terdiri atas komponen atau elemen yang saling terhubung untuk mencapai suatu tujuan tertentu (Ramadhansyah et al., 2024). Sementara itu, informasi adalah data yang telah diproses hingga menjadi lebih berguna dan bermanfaat bagi penggunanya. Informasi juga dapat berupa hasil pengolahan data yang dihasilkan dari berbagai sumber yang dapat menambah pengetahuan bagi individu (Permana & Ariana, 2022).

Sistem informasi (SI) adalah sistem yang dibentuk untuk mengumpulkan, menyimpan, mengelola, dan mendistribusikan informasi. Suatu kombinasi dari teknologi informasi dan aktivitas manusia yang menggunakan teknologi tersebut untuk mendukung operasi, manajemen, dan pengambilan keputusan dalam suatu organisasi (Sari, 2023). Sejak permulaan peradaban, manusia telah bergantung pada sistem informasi untuk berkomunikasi, baik menggunakan verbal maupun instrument perangkat keras, perintah, prosedur pemrosesan informasi, saluran komunikasi, serta sumber daya data yang disimpan (Nur Arifah et al., 2023).

Sistem informasi diterapkan di berbagai bidang seperti di kantor, bandara, pasar swalayan, sekolah, dan rumah. Dalam dunia bisnis yang dinamis, sistem informasi menjadi aset penting yang mendukung inovasi, pengembangan produk atau jasa, dan strategi yang lebih baik. Hal ini bertujuan untuk meningkatkan efisiensi operasi perusahaan atau organisasi agar tetap kompetitif dan relevan di pasar global yang kompleks (Nur Arifah et al., 2023).

2.2. Komponen Sistem Informasi

Beberapa komponen utama dalam suatu sistem informasi yang memiliki fungsi dan tugasnya masing-masing serta saling berkaitan satu dengan yang lainnya. Keterkaitan komponen tersebut membuat sistem informasi dapat mencapai tujuan yang ingin dicapai oleh pengembang dan pengguna sistem (Nur Arifah et al., 2023). Berikut komponen-komponen yang terdapat dalam sistem informasi:

1. **Masukan (*Input*)**

   Masukan (*Input*) ini mengacu pada semua data dan informasi yang masuk ke dalam sistem. Data ini dapat berasal dari berbagai sumber internal organisasi maupun eksternal seperti internet atau sumber lainnya. Masukan diperlukan untuk memulai proses pengelolaan informasi dalam sistem. Dalam konteks aplikasi, masukan sering kali berupa input pengguna berupa teks, gambar, suara, atau bentuk data lainnya yang relevan (Nur Arifah et al., 2023).

2. **Keluaran (*Output*)**

   Keluaran (*Output*) merupakan hasil dari pengelolaan data dan informasi yang telah dimasukkan sebelumnya. Keluaran ini berfungsi sebagai hasil akhir yang disajikan kepada pengguna sistem informasi. Informasi yang dihasilkan dapat berupa laporan, grafik, visualisasi data, atau bentuk lain yang memenuhi kebutuhan pengguna. Tujuan utama keluaran adalah menyediakan informasi yang relevan, akurat, dan dapat dipahami, oleh pengguna untuk mendukung pengambilan keputusan atau kegiatan lainnya (Nur Arifah et al., 2023).

3. **Perangkat Keras (*Hardware*)**

   Perangkat keras (*Hardware*) merupakan komponen fisik yang membentuk infrastruktur dari sistem informasi. Ini mencakup semua perangkat fisik seperti komputer *server*, komputer klien, jaringan komputer (*hub, switch, router*), serta perangkat *mobile* seperti tablet dan *smartphone*. Perangkat keras ini mendukung eksekusi operasi sistem informasi secara fisik, termasuk pengelolaan data, penyimpanan, dan transfer informasi antar perangkat (Nur Arifah et al., 2023).

4. **Perangkat Lunak (*Software*)**

   Perangkat lunak (*Software*) mencakup semua program komputer yang mendukung operasi sistem informasi. Ini termasuk sistem operasi yang menjalankan perangkat keras, aplikasi bisnis seperti aplikasi pengolah kata, *spreadsheet*, *database management system* (DBMS), serta aplikasi khusus lainnya yang dibuat untuk tujuan tertentu dalam sistem informasi. Perangkat

lunak ini berperan penting dalam pengolahan data, analisis, dan penyajian informasi kepada pengguna (Nur Arifah et al., 2023).

5. **Pengguna (*Brainware*)**
   Pengguna (*Brainware*) merupakan komponen manusia yang berinteraksi langsung dengan sistem informasi. Mereka terbagi menjadi pengguna akhir yang menggunakan informasi yang dihasilkan oleh sistem untuk kegiatan operasional atau pengambilan keputusan, serta pengembangan aplikasi yang terlibat dalam pengembangan, pemeliharaan, dan peningkatan sistem informasi. Pengguna mencakup berbagai peran seperti operator, analisis sistem, pemrograman aplikasi, administrator basis data, hingga web master (Nur Arifah et al., 2023).

6. **Basis Data (*Database*)**
   Basis data (*Database*) adalah komponen yang menyimpan, mengelola, dan menyajikan data secara terkomputerisasi. Basis data terdiri dari satu atau lebih tabel yang berhubungan melalui relasi, memungkinkan penyimpanan data dalam format yang terstruktur dan terorganisasi. Basis data menyediakan fasilitas untuk operasi dasar seperti penyisipan (*insert*), pengeditan (*edit*), pembaruan (*update*), dan penghapusan (*delete*) data, serta mengelola integritas data untuk memastikan keakuratan dan konsistensi data (Nur Arifah et al., 2023).

7. **Prosedur dan Kontrol**
   Prosedur mencakup aturan dan prosedur yang harus diikuti dalam pengoperasian sistem informasi untuk mencapai tujuan yang diinginkan. Kontrol berkaitan dengan perlindungan terhadap data dan informasi dari berbagai ancaman seperti keamanan *cyber*, kegagalan perangkat keras, atau bencana alam. Prosedur dan kontrol ini menjamin kehandalan dan keamanan sistem informasi serta menjaga kualitas layanan yang diberikan kepada pengguna (Nur Arifah et al., 2023).

Dengan demikian, integrasi yang baik antara komponen-komponen ini memungkinkan sistem informasi untuk beroperasi secara efektif dalam memproses, menyimpan, mengelola, dan menyajikan informasi yang

diperlukan untuk mencapai tujuan bisnis atau organisasi dengan efisiensi dan akurasi yang tinggi (Nur Arifah et al., 2023).

2.3. Penjadwalan

Penjadwalan merupakan serangkaian mekanisme dalam sistem operasi yang berhubungan dengan pengaturan urutan kerja proses yang sering dilakukan pada sebuah sistem komputer. Tujuan utama dari penjadwalan adalah untuk membantu memahami dan mengoptimalkan pelaksanaan dari kegiatan, seperti menentukan proses mana yang harus berjalan terlebih dahulu, kapan waktu pelaksanaannya, dan berapa lama durasi yang diperlukan untuk menyelesaikan proses tersebut (Putri, 2021).

Jadwal matakuliah adalah salah satu komponen penting dalam pelaksanaan kegiatan belajar mengajar di perguruan tinggi (Marcellino et al., 2022) . Penjadwalan matakuliah merupakan salah satu bentuk permasalahan penjadwalan yang bersifat umum namun juga kompleks, karena bertujuan untuk mengatur jadwal pertemuan berdasarkan ketersediaan berbagai sumber daya, seperti dosen, matakuliah, ruanga kelas, dan waktu. Kesuksesan kegiatan dosen dan mahasiswa dalam belajar sangat bergantung pada jadwal yang telah disusun, sehingga penyusunan jadwal harus dilakukan dengan teliti dan tepat pada awal semester (Marcellino et al., 2022).

Penjadwalan yang efektif harus memperhatikan dua jenis batasan, yaitu Batasan pokok (*hard constraints*) dan Batasan tambahan (*soft constraints*).

1. **Batasan Pokok (*hard constraints*)**
   Batasan pokok (*hard constraints*) adalah syarat-syarat yang harus dipenuhi untuk menghasilkan jadwal yang valid. Pelanggaran terhadap Batasan ini tidak diperbolehkan karena akan menyebabkan konflik atau ketidakefisienan dalam penjadwalan. Batasan pokok (*hard constraints*) dapat mencakup:
   - Tidak boleh ada bentrokan dosen pada waktu yang sama untuk dua kegiatan yang berbeda.
   - Tidak boleh ada bentrokan ruangan pada waktu yang sama untuk dua kegiatan yang berbeda

- Kapasitas ruangan yang digunakan harus sesuai dengan kapasitas yang dibutuhkan oleh kelas.
- Ketersediaan dosen pada waktu yang dijadwalkan.
- Jumlah waktu dan SKS yang sinkron.
- Tidak boleh melanggar jam istirahat.

Pelanggaran terhadap Batasan pokok (*hard constraints*) akan diberikan penalti atau hukuman. Nilai penalti biasanya berkisar antara 0 hingga 1 untuk setiap pelanggaran. Semakin kecil jumlah pelanggaran, semakin baik kualitas jadwal yang dihasilkan (Saud et al., 2017).

2. Batasan Tambahan (*soft constraints*)

Batasan tambahan (*soft constraints*) adalah aturan-aturan istimewa yang ditambahkan untuk mengoptimalkan jadwal berdasarkan kondisi tertentu, meskipun tidak mutlak harus dipenuhi untuk menghasilkan jadwal yang valid. Batasan tambahan (*soft constraints*) dapat mencakup:

- Preferensi dosen untuk mengajar pada waktu tertentu.
- Menghindari penjadwalan matakuliah yang berbeda pada jam yang berurutan untuk menghindari kelelahan dosen dan mahasiswa.
- Mengatur distribusi waktu matakuliah agar tersebar merata dalam kurun waktu satu minggu.
- Menjadwalkan matakuliah di hari tertentu.
- Menggabungkan satu matakuliah untuk beberapa program studi.

Batasan tambahan (*soft constraints*) diperlukan untuk meningkatkan kenyamanan dan efisiensi, meskipun penjadwalan masih bisa dilakukan tanpa memenuhinya. Namun, jika aturan pokok tidak cukup untuk menghasilkan jadwal yang layak pakai, batasan tambahan (*soft constraints*) dapat diintegrasikan untuk meningkatkan kualitas penjadwalan (Saud et al., 2017).

2.4. Algoritma Genetika

Algoritma Genetika (GA) merupakan salah satu metode pencarian yang terinspirasi dari prinsip-prinsip seleksi alam dan genetika. Pendekatan ini terbukti efektif dalam menyelesaikan berbagai permasalahan

optimasi yang kompleks dan sulit diselesaikan dengan metode konvensional. Konsep Algoritma Genetika diperkenalkan oleh John Holland pada tahun 1975 di Universitas Michigan. Holland menjelaskan bahwa, setiap masalah adaptasi alami ataupun buatan dapat dijelaskan melalui terminology genetika (Mone & Simarmata, 2021).

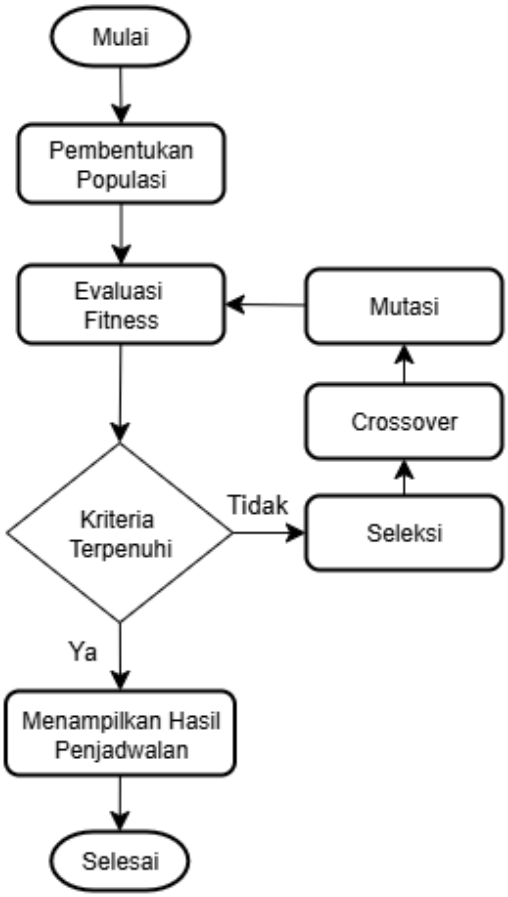
Algoritma Genetika merupakan sebuah metode yang mencoba menerapkan prinsip-prinsip evolusi alam untuk menyelesaikan masalah-masalah tertentu (*problem solving*). Pendekatan yang digunakan oleh Algoritma Genetika adalah dengan cara mengombinasikan secara acak sejumlah solusi terbaik dari suatu populasi untuk membentuk generasi baru yang diharapkan memiliki kualitas solusi yang lebih baik. Tujuan dari mekanisme tersebut untuk meningkatkan nilai kecocokan (*fitness value*) dari setiap solusi yang dihasilkan. Generasi baru ini akan menunjukkan perbaikan-perbaikan pada populasi awalnya. Dengan melakukan proses secara berulang, Algoritma Genetika diharapkan dapat menstimulasikan proses evolusi hingga diperoleh solusi yang optimal (Asri, Al Munir, 2020).

Algoritma Genetika bekerja dengan populasi yang terdiri dari solusi potensi (kromosom). Populasi awal dibentuk secara acak, dan setiap generasi berikutnya dihasilkan melalui evolusi kromosom menggunakan fungsi *fitness* yang mengukur kualitas solusi yang diwakilinya. Berikut beberapa definisi penting dalam algoritma genetika (Asri, Al Munir, 2020), yaitu:

1. Genotype (Gen) merupakan satuan dasar yang membawa informasi tertentu dan dapat direpresentasikan dalam bentuk nilai *biner, integer, float*, ataupun karakter.
2. Allel adalah nilai spesifik yang terkandung di dalam suatu gen.
3. Kromosom adalah sekelompok gen yang tersusun membentuk satu kesatuan yang merepresentasikan satu solusi potensial terhadap permasalahan yang dihadapi.
4. Seleksi adalah proses dalam pemilihan kromosom dengan nilai *finess* tertinggi untuk dipertahankan atau dikembangkan ke generasi berikutnya.
5. *Crossover* adalah tahap penggabungan dua kromosom untuk menghasilkan kromosom baru dengan kombinasi gen dari keduanya.

6. Mutasi adalah proses yang dilakukan dengan cara mengubah sebagian kecil gen dalam kromosom secara acak untuk menjaga keragaman populasi.
7. Nilai *fitness* digunakan untuk mengukur seberapa baik suatu individu atau solusi dalam menyelesaikan masalah.
8. Individu adalah suatu nilai atau keadaan yang mewakili salah satu solusi potensial.
9. Populasi merupakan kumpulan individu yang akan diproses bersama dalam satu siklus evolusi.
10. Generasi adalah siklus perulangan dalam algoritma genetika yang terdiri dari seleksi, crossover, mutasi, dan evaluasi nilai *fitness* hingga kriteria penghentian terpenuhi.

Tingkat konvergensi Algoritma Genetika sangat bergantung pada kekuatan tekanan seleksi. Jika tekanan seleksi terlalu rendah, proses pencarian solusi akan berjalan lambat sehingga algoritma memerlukan waktu yang lama untuk menemukan solusi optimal. Sebaliknya, jika tekanan seleksi terlalu tinggi, algoritma dapat menyatu terlalu cepat ke solusi sub optimal.



Gambar 2. 1 Diagram alir proses algoritma genetika

Pada Gambar 2.1, dapat dijelaskan sebagai berikut:

- Pembentukan Populasi
  Proses yang digunakan untuk membangkitkan populasi awal secara acak sehingga menghasilkan solusi awal.
- Evaluasi *Fitness*
  Proses yang digunakan untuk mengevaluasi setiap individu dalam populasi dengan menghitung nilai *fitness* masing-masing kromosom dan mengevaluasinya hingga kriteria terpenuhi.
- Seleksi
  Proses untuk memilih individu-individu mana saja yang akan digunakan dalam *crossover*.
- *Crossover*
  Proses *crossover* untuk meningkatkan keanekaragaman dalam populasi

- Mutasi
  Mutasi adalah proses mengubah nilai dari satu atau beberapa gen dalam satu kromosom.
- Kriteria Terpenuhi
  Kriteria terpenuhi merupakan syarat-syarat yang digunakan untuk menghentikan proses Algoritma Genetika.
- Menampilkan Hasil Penjadwalan
  Hasil akhir dari solusi optimum yang diperoleh dari Algoritma Genetika.

2.5. Komponen-Komponen Algoritma Genetika

2.5.1 Teknik Pengkodean

Teknik pengkodean adalah teknik untuk mempresentasikan populasi awal sebagai calon solusi dari suatu permasalahan ke dalam bentuk kromosom, yang menjadi elemen kunci dalam perumusan masalah (Wiratna et al., 2023).

2.5.2 Prosedur Inisialisasi (*generate* populasi awal)

Tahap inisialisasi dilakukan dengan menghasilkan sejumlah individu secara acak sesuai dengan batasan solusi. Ukuran populasi dan jenis operator genetika yang digunakan bergantung pada karakteristik permasalahan (Wiratna et al., 2023). Dalam konteks penjadwalan, setiap kromosom merepresentasikan satu jadwal perkuliahan yang memuat informasi seperti program studi, dosen, matakuliah, ruang, haru, dan waktu. Parameter seperti ukuran populasi, jumlah iterasi, dan nilai probabilitas operator ditentukan di awal sebelum proses dimulai (Imam Saepul Azmi & Ina Najiyah, 2023).

2.5.3 Fungsi Evaluasi

Fungsi evaluasi digunakan untuk menilai kinerja setiap individu dalam populasi. Individu dengan nilai *fitness* yang lebih tinggi dianggap mamiliki solusi yang lebih baik dan akan lebih mungkin dipertahankan untuk generasi selanjutnya (Wiratna et al., 2023). Nilai fitness dihitung berdasarkan tingkat pelanggaran terhadap kendala (*constraint*) yang ada. Semakin tinggi nilai *fitness* dalam individu, maka akan semakin tinggi peluang individu tersebut dalam bertahan

hidup. Nilai *fitness* suatu individu dapat dihitung menggunakan rumus fitness (Ulkarim et al., 2020), yaitu sebagai berikut:

$$fitness = \frac{1}{1 + Jumlah\ Constraint\ Dilanggar}\ x\ 100\%$$

2.5.4 Seleksi

Seleksi adalah proses yang digunakan untuk menentukan kandidat solusi terbaik berdasarkan nilai *fitness* dari setiap kromosom. Proses seleksi ini sangat penting karena menentukan kualitas individu yang akan menghasilkan keturunan baru (Wiratna et al., 2023). Proses seleksi dilakukan dengan memilih dua kromosom induk (*parent*) dari populasi berdasarkan nilai *fitness* masing-masing. Semakin tinggi nilai *fitness* suatu kromosom, maka semakin besar pula peluangnya untuk terpilih dalam proses reproduksi.

2.5.5 Operator Genetika

Ada dua operator utama dalam Algoritma Genetika:

1. ***Crossover* (Persilangan)**

   Crossover (persilangan) merupakan tahapan dalam Algoritma Genetika yang berfungsi untuk menggabungkan dua kromosom induk (*parent*) untuk membentuk kromosom baru (*offspring*). Melalui proses ini, diperoleh kromosom yang mengarah pada solusi yang lebih baik hasil dari persilangan dua kromosom tersebut (Wiratna et al., 2023). Beberapa jenis *crossover*, yaitu:

   a. ***Crossover* 1 Titik**

      Pada metode *crossover* 1 titik, sebuah *string* dibagi menjadi dua bagian. Salah satu bagian dari kromosom pertama kemudian ditukar dengan bagian yang sesuai dari kromosom kedua, yang juga dibagi dengan cara yang sama. Proses ini disebut operator *crossover* satu titik (*one-point crossover*). Berikut adalah contoh dari *crossover* 1 titik:

Tabel 2. 1 Contoh Crossover 1 Titik

Kromosom Orangtua 1	**11001**011
Kromosom Orangtua 2	11011**111**
Keturunan	**11001111**

Sumber: (Assagaf et al., 2018)

b. *Crossover* 2 Titik

Proses ini dilakukan dengan menentukan dua titik *crossover* pada kromosom. Kromosom hasil keturunan dibentuk dengan menyalin bagian awal kromosom induk pertama hingga titik *crossover* pertama. Bagian antara titik *crossover* pertama dan kedua disalin dari orang tua kedua, dan sisanya disalin kembali dari orang tua pertama. Berikut adalah contoh dari *crossover* 2 titik:

Tabel 2. 2 Contoh Crossover 2 Titik

Kromosom Orangtua 1	**11**0010**11**
Kromosom Orangtua 2	11**011**11
Keturunan	**11011111**

Sumber: (Assagaf et al., 2018)

c. *Crossover* Seragam

Dalam *crossover* seragam, bit-bit dari kedua orang tua disalin secara acak untuk membentuk kromosom keturunan. Ini berarti setiap bit dari kromosom keturunan bisa berasal dari salah satu dari dua orang tua, dipilih secara acak. Berikut contoh *crossover* seragam:

Tabel 2. 3 Contoh Crossover Seragam

Kromosom Orangtua 1	**110**01011
Kromosom Orangtua 2	110**111**11
Keturunan	**11011111**

Sumber: (Assagaf et al., 2018)

2. ***Mutation*** **(Mutasi)**

Proses mutasi dilakukan dengan mengubah satu atau beberapa gen dalam sebuah kromosom. Tahapan ini berfungsi untuk menggantikan gen yang mungkin hilang selama proses seleksi serta memungkinkan munculnya kembali gen-gen yang sebelumnya tidak terbentuk pada saat inisialisasi populasi (Wiratna et al., 2023).

Operasi *crossover* dilakukan pada kromosom dengan tujuan untuk menghasilkan kromosom baru yang berpotensi menjadi solusi pada generasi mendatang dengan tingkat *fitness* yang lebih baik. Untuk menentukan panjang sebuah kromosom, digunakan rumus sebagai berikut (Toyib Hidayat et al., 2023):

$$N_{gen} = (H \times R \times W)$$

Keterangan:

N_{gen} = Jumlah gen dalam satu kromosom
H = Jumlah hari dalam perkuliahan
R = Jumlah ruangan dalam perkuliahan
W = Jumlah waktu dalam perkuliahan

Operator mutasi adalah operasi yang berhubungan dengan satu kromosom tertentu. Beberapa metode operasi mutasi yang diterapkan dalam Algoritma Genetika tergantung pada jenis pengkodean terhadap fenotipe, antara lain:

a. **Mutasi dalam Pengkodean Biner**

Merupakan operasi yang relatif sederhana dalam proses Algoritma Genetika. Proses yang dilakukan adalah dengan membalik nilai bit pada posisi tertentu yang dipilih secara acak (menggunakan skema tertentu) pada kromosom, yang disebut dengan *invers bit* (Asri, Al Munir, 2020).

Tabel 2. 4 Contoh Mutasi dalam Pengkodean Biner

Kromosom sebelum mutasi	10010**1**11
Kromosom setelah mutasi	10010**0**11

Sumber:(Asri, Al Munir, 2020)

b. Mutasi dalam Pengkodean Permutasi

Berbeda dengan pengkodean biner yang memungkinkan perubahan nilai bit, pada pengkodean permutasi, mutasi tidak dapat dilakukan dengan cara yang sama kerena urutan elemen harus tetap konsisten. Salah satu metode yang dapat dilakukan adalah dengan memilih dua porsi (*locus*) pada kromosom dan menukar nilai di antara keduanya (Asri, Al Munir, 2020).

Tabel 2. 5 Contoh Mutasi dalam Pengkodean Permutasi

Kromosom sebelum mutasi	12**3**4658**7**9
Kromosom sesudah mutasi	12**7**4658**3**9

Sumber: (Asri, Al Munir, 2020)

c. Mutasi dalam Pengkodean Nilai

Mutasi pada pengkodean nilai hampir mirip dengan yang dilakukan pada pengkodean biner, tetapi yang diubah bukanlah nilai bit. Penerapannya bergantung pada jenis nilai yang digunakan. Sebagai contoh, untuk nilai *real*, proses mutasi dapat dilakukan seperti pada pengkodean permutasi, dengan saling menukar nilai dua gen pada kromosom (Asri, Al Munir, 2020).

d. Mutasi dalam Pengkodean Pohon

Pada pengkodean pohon, mutasi dapat dilakukan dengan cara mengganti operator matematika seperti (+, -, *, /) atau mengganti nilai yang terdapat pada simpul (*vertex*) tertentu. Alternatif lainnya adalah dengan menukar dua simpul pada pohon,

baik berupa operator maupun nilai yang dikandungnya (Asri, Al Munir, 2020).

Tidak semua gen dalam kromosom mengalami mutasi, karena proses ini dikendalikan oleh suatu nilai probabilitas tertentu yang disebut dengan *mutation rate* (probabilitas mutasi) dan dinotasikan sebagai Pm (Asri, Al Munir, 2020).

Dengan komponen-komponen ini, Algoritma Genetika bekerja melalui siklus iteratif untuk menentukan solusi yang optimal atau mendekati optimal terhadap masalah yang diberikan.

2.6. Kajian Teori Algoritma Genetika

Berdasarkan penelitian-penelitian yang telah dilakukan sebelumnya mengenai penggunaan Algoritma Genetika maka dapat dirangkum sebagai berikut:

Tabel 2. 6 Kajian Teori

No.	Peneliti	Judul Penelitian	Hasil Penelitian
1.	(Toyib Hidayat et al., 2023)	Implementasi Sistem Penjadwalan Mata Kuliah Menggunakan Metode Algoritma Genetika Berbasis Web	Menunjukkan Algoritma Genetika dapat meningkatkan efisiensi dan mengatasi kendala penjadwalan di lingkungan akademik
2.	(Pambudi et al., 2021)	Perancangan Sistem Penjadwalan Perkuliahan Berbasis Website Menggunakan Algoritma Genetika	Algoritma Genetika memastikan sistem informasi penjadwalan dengan data yang dihasilkan tidak ada bentrokan.
3.	(Wicaksono & Putra, 2021)	Course Scheduling Information System Using Genetic Algorithms	Membuat penjadwalan matakuliah menjadi lebih efisien dan

No.	Peneliti	Judul Penelitian	Hasil Penelitian
			meminimalkan bentrok
4.	(Muh Syawal et al., 2021)	Implementasi Algoritma Genetika Untuk Penjadwalan Laboratorium Fakultas Ilmu Komputer Universitas Muslim	Membuat jadwal laboratorium sesuai aturan. Metode blackbox memastikan sistem berjalan sesuai tujuan.
5.	(Eka Yulia Sari et al., 2023)	Pemodelan Sistem Optimasi Penjadwalan Matakuliah dengan Algoritma Genetika	Penjadwalan dapat diatasi dengan menggunakan Algoritma Genetika. Pemodelannya memakai UML dan basis data menggunakan ERD
6.	(Kordos et al., 2020)	Optimization of Warehouse Operations With Genetic Algorithms	Sistem mengatasi ruang pencarian dan menganalisis ribuan kemungkinan untuk solusi yang optimal.

2.7. Framework

Framework merupakan suatu kerangka kerja yang terdiri dari kumpulan komponen-komponen program yang memiliki fungsi tertentu untuk membantu menjalankan berbagai perintah, sehingga proses penulisan kode menjadi lebih efektif dan efisien. Meskipun demikian, penggunaan *framework* memerlukan pemahaman yang baik dari pengembang terhadap aturan dan struktur yang telah ditentukan oleh *framework* tersebut (Apriliando, 2021). Dalam konteks web *framework*, kerangka kerja ini menyediakan beragam fitur seperti fungsi, sintaks, *library*, ekstensi, dan *template* siap pakai yang memudahkan dan mempercepat proses pengembangan *website*. Banyak jenis kerangka kerja ditemukan termasuk Laravel. Pada aplikasi berbasis web, pengembang harus mengikuti aturan kerangka kerja yang telah ditetapkan oleh *framework*. *Framework* (dalam hal ini PHP *Framework*), tidak perlu memikirkan

kode perintah dasar/fungsi aplikasi *website* (G. R. U. Sinaga & Samsudin, 2021).

2.8. Laravel

Laravel merupakan kerangka kerja pemrograman berbasis *open source* yang popular dan banyak digunakan oleh pengembang di seluruh dunia. Framework ini menerapkan pola arsitektur *Model-View-Controller* (MVC) yang memisahkan aplikasi menjadi tiga komponen utama, yaitu pengelolaan data (*model*), pengendali logika (*controller*), dan tampilan antarmuka pengguna (*view*) (Awaluddin et al., 2020). Pendekatan ini memberikan keuntungan dalam hal kemudahan pemeliharaan serta peningkatan skala (*scalability*) aplikasi secara lebih efisien.

1. **Model**
   Model mewakili struktur data dan biasanya berisi fungsi-fungsi untuk mengelola basis data, seperti memasukkan, memperbarui, dan menghapus data.
2. ***View***
   View adalah bagian yang mengatur tampilan kepada pengguna, umumnya berupa halaman web.
3. ***Controller***
   Controller adalah komponen yang menghubungkan model dan *view*, mengatur aliran data di antara keduanya.

Filament merupakan paket tambahan (*package*) untuk Laravel yang digunakan untuk membangun admin panel modern, dashboard, dan manajemen data dengan cepat. Dengan menggunakan *Filament* dapat memudahkan untuk membuat dashboard custom, portal pengguna, atau bahkan aplikasi lengkap dengan beberapa panel. *Filament* juga terintegrasi dengan mulus dengan tumpukan *frontend* apa pun dan bekerja sangat baik dengan alat seperti *Livewire* ataupun *Blade* (FilamentPHP, 2025).

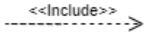
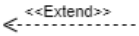
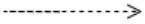
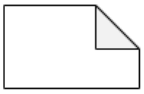
2.9. Unified Modeling Language (UML)

Unified Modeling Language (UML) merupakan bahasa grafis yang digunakan untuk memvisualisasikan, menentukan, membangun, dan mendokumentasikan sistem perangkat lunak yang kompleks. UML menyediakan cara yang standar untuk menulis *blueprint* sistem, yang mencakup aspek-aspek konseptual, kelas yang ditulis dengan bahasa pemrograman tertentu, skema *database*, dan komponen perangkat

lunak yang dapat digunakan kembali. Diagram yang digunakan dalam analisis pemodelan adalah *use case diagram*, *activity diagram*, dan *class diagram* (Ansari & Subairi, 2020).

1. ***Use Case Diagram***

   Use case diagram merupakan salah satu jenis diagram UML yang digunakan untuk menggambarkan fungsi, ruang lingkup, serta hubungan interaksi antara pengguna dan sistem. Diangan ini menunjukkan interaksi antara *actor* dengan sistem yang dikembangkan, serta menjelaskan bagaimana peran dan aktivitas yang dilakukan oleh satu atau lebih *actor* dalam menggunakan sistem yang akan dibuat (Purnasari & Hartiwi, 2022). Berikut simbol *Use Case Diagram* beserta keterangannya:

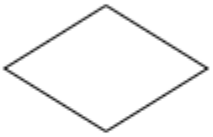
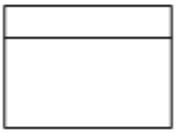
Simbol	Nama	Keterangan
 Actor	Actor	Entitas yang berinteraksi dengan sistem
 Use Case	Use Case	Akttivitas yang dapat dilakukan actor pada sistem
 parent child	Assosiation	Hubungan antara actor dengan use case
	System	Sistem yang sedang dikembangkan
 <<Include>>	Include	Suatu use case termasuk bagian dari use case lain
 <<Extend>>	Extend	Satu use case dapat diperluas dengan use case lain
	Dependency	Ketergantungan antar elemen-elemen diagram
	Generalization	Satu actor atau use case merupakan generalisasi dari yang lain
	Realization	Implementasi dari satu use case oleh yang lain
	Collaboration	Dua atau lebih actor dan use case yang terhubung
	Note	Penjelasan tambahan terkait elemen-elemen diagram
	Anchor	Hubungan teks note dengan elemen-elemen diagram lain

Gambar 2. 2 Simbol Use Case Diagram

2. ***Activity Diagram***

Activity diagram adalah diagram yang dapat memodelkan proses-proses dalam suatu sistem, menggambarkan runtutan

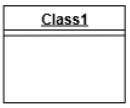
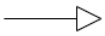
proses secara vertikal. Diagram ini merupakan pengembangan dari *use case diagram* yang menampilkan alur aktivitas dalam sistem. *Activity diagram* dapat membantu memodelkan alur kerja sistem dengan baik, menganalisis *use case* dengan menjelaskan *actor*, tindakan yang diperlukan, dan waktu pelaksanaannya (Dicoding Intern, 2021). Berikut simbol *activity diagram* beserta keterangannya:

Simbol	Nama	Keterangan
	Status awal	Sebuah diagram aktivitas memiliki sebuah status awal
	Aktivitas	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja
	Percabangan/ Desicion	Percabangan dimana ada pilihan aktivitas yang lebih dari satu
	Penggabungan/ Join	Yang mana lebih dari satu aktiviyas lalu digabungkan jadi satu
	Status Akhir	Akhir dari suatu sistem
	Swimlane	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi

Gambar 2. 3 Simbol Activity Diagram

3. *Class Diagram*

Class diagram merupakan representasi hubungan antar kelas dalam suatu sistem dan menggambarkan alur kerja sebuah *database* pada sistem yang akan dikembangkan. Diagram ini terdiri atas kumpulan dari beberapa *class* dan relasinya, yang berfungsi untuk menangkap struktur dari seluruh *class* pembentuk arsitektur sistem. *Class diagram* membantu dalam memvisualisasikan struktur serta keterkaitan antar *class* dan menjadi salah satu jenis diagram yang paling sering digunakan dalam pemodelan sistem. Selain itu, *class diagram* berperan untuk menjelaskan jenis objek dalam sistem serta hubungan antar objek tersebut (Ramdany, 2024). Berikut simbol *class diagram* beserta keterangannya:

Simbol	Nama	Keterangan
	Class	Class adalah blok-blok pembangunan pada pemrograman berorientasi objek.
	Association	Sebuah hubungan yang menunjukkan adanya interaksi antar *class*
	Aggregation	Mengindikasikan keseluruhan bagian relationship dan biasanya disebut relasi
	Compostion	Saat class yang tidak bisa berdiri sendiri dan bergantung pada class lain memiliki relasi Composition. digambarkan dengan garis berujung jajaran genjang solid
	Generalization	sebuah hubungan antar class yang bersifat dari khusus ke umum.
	Dependency	menunjukkan operasi pada suatu class yang menggunakan class yang lain.

Gambar 2. 4 Simbol Class Diagram