BAB 2
TINJAUAN PUSTAKA DAN DASAR TEORI

Bab 2 menjelaskan konsep dasar dan merujuk pada penelitian sebelumnya yang digunakan sebagai landasan untuk mendukung penelitian ini. Referensi-referensi dari berbagai sumber, termasuk buku dan jurnal, menjadi acuan yang penting dalam merumuskan alternatif solusi yang sesuai untuk mengatasi masalah yang dikaji dalam penelitian.

2.1. Tinjauan Pustaka

Pada sub bab tinjauan pustaka berisi cara-cara penelitian terdahulu dalam menyelesaikan masalah serupa. Masalah yang dimaksud adalah penjadwalan yang masih manual atau belum terkomputerisasi sehingga penyusunan jadwal mata kuliahnya relatif lebih lama dan bentrok antara mata kuliah satu dengan yang lainnya. Dengan melakukan tinjauan pustaka, maka diperoleh metode-metode apa yang digunakan serta solusi yang dihasilkan untuk mengatasi masalah tersebut. Untuk memperoleh hasil penelitian terdahulu, dilakukan pencarian jurnal di Scopus dengan kata kunci *University Course Timetabling Problem*.

Dari hasil pencarian tersebut, diperoleh beberapa jurnal penelitian yang dipublikasikan beberapa tahun terakhir. Dalam permasalahan terkait penjadwalan mata kuliah, metode atau algoritma yang digunakan penelitian terdahulu sangat beragam. Seperti pada penelitian Harrabi dkk. (2024), Yavuz dkk. (2023), dan Rivero dkk. (2020) menggunakan *integer programming* untuk memperoleh solusi penjadwalan. Penelitian yang menggunakan cara ini menghasilkan solusi yang lebih cepat secara waktu dan kualitas.

Selain itu, terdapat juga penelitian lain yang menggunakan satu metode atau algoritma saja. Seperti pada penelitian Mazlan dkk. (2019) yang menggunakan algoritma *Ant Colony Optimization* (ACO) untuk memperoleh solusi jadwal. Kemudian dengan cara yang berbeda, diperoleh jadwal menggunakan *Genetic Algorithm* seperti pada penelitian Abduljabbar & Abdullah (2022).

Selanjutnya pada penelitian Goh dkk. (2019), dilakukan perbandingan beberapa algoritma, yaitu *Monte Carlo Tree Search* (MCTS), *Graph Colouring Heuristic* (GCH), dan *Tabu Search* (TS). Algoritma tersebut dibandingkan dan diketahui bahwa secara performa, MCTS lebih unggul dibandingkan GCH tetapi tidak lebih

baik dari TS. Oleh karena itu, penggunaan MCTS pada penelitian tersebut lebih membutuhkan waktu dalam menghasilkan jadwal.

Penelitian yang menggunakan konsep perbandingan juga dilakukan pada penelitian Muklason dkk. (2022), yaitu menggunakan *Late Acceptance Hill Climbing* (LAHC) untuk kemudian dibandingkan dengan *Graph Colouring* dan *Hill Climbing.* Setelah dilakukan perbandingan, diperoleh bahwa penggunaan LAHC lebih baik hasilnya ketimbang *graph coloring* dan *hill climbing*. Perbandingan yang dilakukan dalam penelitian ini bisa saja menghasilkan *output* berbeda tergantung kondisi aturan dan data yang diberikan.

Setelah banyaknya penelitian, dilakukan pengembangan cara untuk memperoleh solusi pada masalah penjadwalan. Adanya kekurangan pada setiap algoritma menghasilkan gagasan baru yang tujuannya untuk mengkombinasikan 2 atau lebih cara untuk pencarian solusi. Idenya adalah kombinasi dilakukan untuk menutup kekurangan dari masing-masing cara atau algoritma.

Penelitian oleh Abdalla dkk. (2019) menggunakan dua tahapan dalam penggunaan *integer programming*. Tahap pertama digunakan untuk pencarian solusi yang *feasible,* sedangkan tahap kedua digunakan untuk memenuhi *soft constraints* yang ada. Hal ini tentu berbeda dengan penelitian yang hanya menggunakan *integer programming* saja tanpa ada cara khusus tertentu karena penelitian ini memiliki pengembangan menjadi 2 proses untuk memperoleh solusi yang lebih baik.

Pada penelitian Ghaffar dkk. (2018), dilakukan kombinasi algoritma *memetic* berbasis *fuzzy multi-objective* yang dibandingkan dengan algoritma genetika, algoritma *tabu search*, dan algoritma *hill climbing* ataupun bentuk *hybrid*-nya. Hasilnya diperoleh solusi yang lebih optimal dengan akurasi mencapai 96,29%. Penggunaan logika *fuzzy* juga diterapkan pada penelitian Babaei dkk. (2019), dikombinasikan *fuzzy multi-criteria comparison algorithm* dengan *local search algorithm* dan *clustering algorithm*. Kombinasi ini dilakukan untuk mencapai tujuan tertentu, yaitu menghasilkan jadwal yang meningkatkan kepuasan dosen umum dan peningkatan performa untuk menangani *soft constraints.*

Selain itu, ada juga yang menggunakan kombinasi algoritma genetika dan algoritma *tabu search* pada penelitian Wong dkk. (2022) untuk menghasilkan solusi secara otomatis dengan adanya peningkatan hasil dari pengurangan *hard constraint* dan *soft constraint* sekitar 54%. Kemudian dengan algoritma yang

serupa dengan tambahan *mixed integer linear programming* pada penelitian Arias-Osorio & Mora-Esquivel (2020), diperoleh bahwa adanya peningkatan secara waktu dibanding cara manual.

Selain dikombinasikan dengan algoritma genetika, *tabu search* digunakan pada penelitian Muklason dkk. (2019) untuk dikombinasikan dengan metode lain. *Tabu search* pada penelitian tersebut di-*hybrid* dengan *variable neighborhood search*. Bentuk kombinasi ini menghasilkan jadwal yang terotomatisasi dan penalti yang berkurang signifikan pada pecobaan menggunakan dua *dataset*.

Di luar penggunaan algoritma tertentu, terdapat juga cara lain yang digunakan pada penelitian Liu dkk. (2023) dan penelitian Kumar & Pandey (2020). Penelitian Liu dkk. (2023) menggunakan model simulasi gabungan *MATLAB-EnergyPlus* untuk menghasilkan jadwal yang hemat energi pada musim panas dan dingin. Berbeda dengan penelitian Liu dkk. (2023), penelitian Kumar & Pandey (2020) menggunakan cara yang berkaitan dengan *SQL database* dan menggunakan bahasa pemrograman C#.

Dalam penelitian secara lokal di Indonesia khususnya di Universitas Atma Jaya Yogyakarta, terdapat juga penelitian yang serupa terkait masalah penjadwalan mata kuliah. Pada penelitian Latumahina (2021), diketahui bahwa sistem penjadwalan yang dirancang menghasilkan jadwal yang cepat dengan metode modifikasi hibrida antara *Genetic Algorithm* dan *Largest Color Degree.* Metode yang digunakan juga menggunakan konsep gabungan seperti penelitian lainnya untuk saling memberikan kelebihan tiap algoritma.

Setelah dilakukan tinjauan-tinjauan terhadap penelitian sebelumnya, maka terdapat sebuah *gap* yang belum dieksplorasi. *Gap* tersebut adalah keinginan dosen untuk berada di sesi atau ruang tertentu yang belum diteliti oleh penelitian sebelumnya. Oleh karena itu, pada penelitian ini *gap* tersebut akan dieksplorasi lebih lanjut.

Setelah membahas referensi penelitian sebelumnya, selanjutnya dilakukan rekapitulasi dalam bentuk tabel. Hasil rekapitulasi dan disajikan pada Tabel 2.1.

Tabel 2.1. Ringkasan Tinjauan Pustaka

Peneliti	Objek Penelitian	Permasalahan	Metode	Hasil penelitian	Penggunaan Informasi
Ghaffar dkk., 2018	University of Management & Technology, Lahore.	Adanya konflik pada jadwal mata kuliah yang dibuat.	Kombinasi dari algoritma memetika berbasis *fuzzy multi-objective*, algoritma genetika, *tabu search*, dan *hill climbing algorithm.*	Menghasilkan solusi yang lebih optimal dengan akurasi 96,29% dalam menyelesaikan konflik dibanding algoritma genetika sederhana ataupun *hybrid.*	Digunakan pada Bab 3 sebagai dasar pembangkitan solusi dan dasar penggunaan metode.
Harrabi dkk., 2024	High Institute of Business Studies, Carthage.	Kesulitan dalam menetapkan jadwal karena kendala yang saling bertentangan.	*Integer linear programming.*	Hasil komputasi menghasilkan performa yang signifikan dibanding proses manual secara waktu dan kualitas.	Digunakan pada Bab 3 sebagai dasar pembangkitan solusi dan dasar penggunaan metode.
Wong dkk., 2022	University Malaysia Sabah-Labuan International Campus.	Adanya masalah penjadwalan pasca proses pendaftaran di universitas terkait.	Kombinasi *genetic algorithm* dan *tabu search with sampling and perturbation.*	Berdasarkan perbandingan hasil otomatis dan manual, hasil jadwal secara otomatis lebih unggul dalam hal pelanggaran *hard constraint* dan *soft constraint* dengan peningkatan sekitar 54%.	Digunakan pada Bab 3 sebagai dasar pembangkitan solusi dan dasar penggunaan metode.

Tabel 2.1. Lanjutan

Peneliti	Objek Penelitian	Permasalahan	Metode	Hasil penelitian	Penggunaan Informasi
Liu dkk., 2023	Universitas di Xi'an.	Adanya konsumsi energi yang tidak perlu pada musim panas dan musim dingin.	Model simulasi gabungan *MATLAB-EnergyPlus.*	Hasil optimasi penjadwalan menghemat energi pada minggu ujian musim panas, minggu non-ujian musim panas, minggu ujian musim dingin, minggu non-ujian musim dingin sebesar 35%, 29.4%, 13.4%, dan 13.4%.	Digunakan pada Bab 3 sebagai dasar pembangkitan solusi dan dasar penggunaan metode.
Yavuz dkk., 2023	Izmir University Bakırcay.	Adanya kesulitan penjadwalan seiring dengan bertambahnya jumlah departemen dan mata kuliah.	*Integer programming.*	Dihasilkan penjadwalan yang optimal terhadap aturan dan keinginan dosen secara otomatis menggunakan bantuan *software* IBM ILOG CPLEX.	Digunakan pada Bab 3 sebagai dasar pembangkitan solusi dan dasar penggunaan metode.
Goh dkk., 2019	Universiti Malaysia Sabah.	Adanya masalah penjadwalan yang bentrok sehingga mahasiswa tidak dapat memilih mata kuliah favorit setiap semester.	*Monte Carlo Tree Search (MCTS), Graph Colouring Heuristic (GCH),* dan *Tabu Search (TS).*	MCTS lebih unggul kinerjanya dibandingkan dengan GCH, tetapi lebih buruk dibandingkan dengan TS dalam pencarian solusi. Selain itu MCTS butuh waktu untuk dapat bekerja secara kompeten dalam menghasilkan jadwal.	Digunakan pada Bab 3 sebagai dasar pembangkitan solusi dan dasar penggunaan metode.

Tabel 2.1. Lanjutan

Peneliti	Objek Penelitian	Permasalahan	Metode	Hasil penelitian	Penggunaan Informasi
Babaei dkk., 2019	Multi departemen dari Islamic Azad University of Ahar branch.	Proses penjadwalan yang menggunakan waktu banyak dan repetitif.	*Fuzzy multi-criteria comparison algorithm*, *local search algorithm*, dan *clustering algorithm.*	Menghasilkan pengurangan preferensi umum dosen karena adanya alokasi ke sumber daya tambahan, meningkatkan konfigurasi bobot *soft constraints* dengan *fuzzy,* dan meningkatkan kepuasan dosen umum dengan menerapkan metode pencarian lokal.	Digunakan pada Bab 3 sebagai dasar pembangkitan solusi dan dasar penggunaan metode.
Kumar & Pandey, 2020	The Institute of Chartered Financial Analysts of India University, Dehradun	Adanya kesulitan dan kerumitan dalam penyusunan jadwal.	Menggunakan *SQL database* dan bahasa C#.	Menghasilkan jadwal bebas konflik dengan menggunakan waktu dan memori utama yang sangat sedikit.	Digunakan pada Bab 3 sebagai dasar pembangkitan solusi dan dasar penggunaan metode.
Muklason dkk., 2022	Departemen Sistem Informasi, Institut Teknologi Sepuluh Nopember.	Waktu pembuatan jadwal yang lama dan rumit karena keterbatasan waktu beberapa dosen.	*Late Acceptance Hill Climbing (LAHC)* dan *Graph Coloring.*	Jadwal yang diperoleh menggunakan LAHC menghasilkan penalti yang lebih optimal, yaitu sebesar 103,9 dibandingkan dari *Graph Coloring* dan *Hill Climbing*, yaitu 753,84 dan 189,59.	Digunakan pada Bab 3 sebagai dasar pembangkitan solusi dan dasar penggunaan metode.

Tabel 2.1. Lanjutan

Peneliti	Objek Penelitian	Permasalahan	Metode	Hasil penelitian	Penggunaan Informasi
Arias-Osorio & Mora-Esquivel, 2020	Industrial University of Santander.	Kesulitan dan kerumitan dalam melakukan penjadwalan.	*Mixed integer linear programming, hybrid genetic algorithm,* dan *tabu search algorithm.*	Waktu komputasi yang lebih cepat dibanding penyusunan manual yang memakan waktu berjam-jam hingga berhari-hari.	Digunakan pada Bab 3 sebagai dasar pembangkitan solusi dan dasar penggunaan metode.
Muklason dkk., 2019	Departemen Sistem Informasi, Institut Teknologi Sepuluh Nopember.	Adanya kesulitan penjadwalan untuk memenuhi permintaan mahasiswa ataupun dosen.	*Tabu search hybridize with variable neighborhood search.*	Jadwal dapat diotomatisasi dan pelanggaran penaltinya berkurang dari 2675 dan 1984 menjadi 820 dan 874 (menggunakan dua *dataset*).	Digunakan pada Bab 3 sebagai dasar pembangkitan solusi dan dasar penggunaan metode.
Mazlan dkk., 2019	Universitas Sultan Zainal Abidin.	Kesulitan penjadwalan karena meningkatnya jumlah siswa dan kompleksitas.	*Ant Colony Optimization (ACO).*	Diperoleh jadwal yang tidak bentrok (*feasible*) dan dapat diandalkan.	Digunakan pada Bab 3 sebagai dasar pembangkitan solusi dan dasar penggunaan metode.
Abduljabbar & Abdullah, 2022	Department of Computer Science, University of Technology, Baghdad.	Kesulitan dalam proses penjadwalan mata kuliah dalam departemen.	*Genetic algorithm.*	Menghasilkan jadwal yang fleksibel dan beragam tergantung masukan yang diberikan.	Digunakan pada Bab 3 sebagai dasar pembangkitan solusi dan dasar penggunaan metode.

Tabel 2.1. Lanjutan

Peneliti	Objek Penelitian	Permasalahan	Metode	Hasil penelitian	Penggunaan Informasi
Abdalla dkk., 2019	University of Malaysia Sabah Labuan International Campus.	Penjadwalan yang rumit karena adanya fitur dan aturan tertentu dari universitas.	Dua tahap *integer programming.*	Diperoleh solusi yang layak pada tahap pertama dan peningkatan sebesar 10,99% dan 8,92% dengan memenuhi *soft constraints* pada tahap kedua.	Digunakan pada Bab 3 sebagai dasar pembangkitan solusi dan dasar penggunaan metode.
Rivero dkk., 2020	Faculty of Engineering of the Autonomous University of Querétaro.	Kesulitan melakukan penjadwalan karena mata kuliah yang banyak dan keterbatasan sumber daya.	*Integer programming.*	Diperoleh solusi jadwal yang awalnya dikerjakan selama 20 hari menjadi hanya setengah jam.	Digunakan pada Bab 3 sebagai dasar pembangkitan alternatif solusi dan dasar penggunaan metode.
Mokhtari dkk., 2021	Industrial Engineering Department of Islamic Azad University, Najafabad Branch (IAUN)	Penjadwalan yang rumit karena faktor jumlah mahasiswa yang terus bertambah dan minimnya fasilitas.	Multi objektif *mixed-integer programming.*	Jadwal yang dihasilkan lebih efisien dan mengurangi waktu tempuh antar kelas dan kelebihan kapasitas kelas.	Digunakan pada Bab 3 sebagai dasar pemilihan alternatif solusi dan dasar penggunaan metode.

Tabel 2.1. Lanjutan

Peneliti	Objek Penelitian	Permasalahan	Metode	Hasil penelitian	Penggunaan Informasi
Latumahina, 2021	Departemen Teknik Industri, UAJY	Penjadwalan yang dilakukan lama dalam memperoleh jadwal *feasible*.	*Modified Hybrid Genetic Algorithm* (MHGA)	Menghasilkan jadwal dengan durasi kurang lebih 1 menit berdasarkan dataset yang digunakan.	Digunakan pada Bab 3 sebagai dasar pembangkitan alternatif solusi dan dasar penggunaan metode.
Sinaga & Aknuranda, 2020	Fakultas Ilmu Komputer, Universitas Brawijaya	Keterlambatan publikasi jadwal mata kuliah	*Quality Evaluation Framework*	Penelitian menunjukkan adanya kekurangan pada SOP saat ini terkait waktu target yang tidak realistis dan sumber daya terbatas sehingga perlu diperbarui.	Digunakan pada Bab 3 sebagai dasar pembangkitan alternatif solusi.

2.2. Dasar Teori

2.2.1. Penjadwalan Mata Kuliah

Menurut Leung (2004), penjadwalan dalam dunia pendidikan merupakan salah satu aktivitas administratif utama dalam kebanyakan universitas. Permasalahan penjadwalan dalam universitas atau sering disebut juga *University Course Timetabling Problem* (UCTP) diklasifikasikan dalam 2 bagian yang perlu dijadwalkan, yaitu penjadwalan mata kuliah dan penjadwalan ujian. Dalam melakukan penjadwalan, sejumlah mata kuliah dialokasikan ke dalam ruang dan sesi yang tersedia tergantung dengan *constraint* yang ada.

Terkait dengan *constraints* (batasan), biasanya *constraint* dibedakan menjadi dua jenis, yaitu *hard constraints* dan *soft constraint*. *Hard constraints* merupakan batasan yang tidak boleh dilanggar oleh universitas sehingga jenis batasan ini wajib untuk tercapai. Jika nantinya jadwal yang diperoleh tidak ada pelanggaran *hard constraints* maka dapat disebut sebagai jadwal yang *feasible*. Pada umumnya, *hard constraints* terdiri dari hal-hal sebagai berikut:

a. Tidak boleh ada orang yang dialokasikan untuk berada di lebih dari satu tempat pada satu waktu.
b. Total sumber daya yang dibutuhkan pada periode tertentu tidak boleh lebih besar dari sumber daya yang tersedia.

Kemudian terdapat juga yang disebut sebagai *soft constraints*. *Soft constraints* merupakan batasan-batasan yang ingin dipenuhi, tetapi batasan tersebut tidak penting atau hanya sebatas ingin dipenuhi saja. Dalam penerapannya di dunia universitas, umumnya sangat mustahil untuk dapat memenuhi semua *soft constraints* yang ada. Akan tetapi, kualitas jadwal yang *feasible* dapat dinilai berdasarkan seberapa baik *soft constraints* dapat dipenuhi. Meskipun begitu, beberapa masalah penjadwalan di universitas terkadang sangat kompleks sehingga sulit untuk menemukan solusi yang *feasible.*

Di banyak universitas, terdapat banyak sekali jumlah mata kuliah yang harus dijadwalkan. Dalam hal ini, jumlah yang harus dijadwalkan bisa mencapai ratusan atau bahkan ribuan mata kuliah/ujian. Oleh karena itu, kendala yang harus dipertimbangkan dalam pembuatan jadwal tersebut menjadi tugas yang sangat sulit. Solusi yang dilakukan secara manual membutuhkan usaha yang sangat besar. Adanya perkembangan metode otomatis dalam melakukan penjadwalan

menjadi tantangan yang menarik terutama dalam ilmu *Operational Research* dan *Artificial Intelligence*.

Berbagai macam pendekatan terhadap masalah penjadwalan telah dilakukan dan diuji pada data dari universitas terkait mata kuliah dan ujian. Pendekatan-pendekatan ini dapat diklasifikasikan dalam kriteria yang berbeda. Berdasarkan klasifikasi pendekatan yang disajikan dalam penelitian (Carter & Laporte, 1998), terdapat beberapa pendekatan yang telah berkembang, yaitu sebagai berikut:

a. Pendekatan berbasis *Mathematical Programming* yang mana pendekatan ini memiliki kebutuhan *computational* yang tinggi, tetapi kurang atraktif untuk jumlah penjadwalan yang banyak. Meskipun begitu, pendekatan ini tetap digunakan secara *hybrid* dengan metode lain.
b. Pendekatan berbasis pewarnaan grafik yang mana masalah penjadwalan dapat direpresentasikan dalam bentuk grafik. Untuk bagian simpul menandakan mata kuliah/ujian, sedangkan bagian sisi menunjukkan konflik kegiatannya seperti ketika mahasiswa diharuskan untuk menghadiri kedua kegiatan tersebut). Contoh beberapa pendekatan pewarnaan grafik dalam penjadwalan secara heuristik adalah:
    i. *Largest degree first* yang memprioritaskan *events* yang memiliki jumlah konflik terbesar dibanding *events* lain.
    ii. *Largest weighted degree* yang menghitung konflik antar *events* kemudian diberi bobot berdasarkan jumlah mahasiswa yang terlibat
    iii. *Saturation degree* yang mana dilakukan penjadwalan terlebih dahulu *events* yang memiliki periode valid terkecil.
    iv. *Color degree* yang mana kegiatan diurutkan berdasarkan jumlah konflik yang dimiliki suatu *event* dengan *events* yang telah dijadwalkan.
c. Metode kluster yang mana metode ini membagi terlebih dahulu himpunan *event* ke dalam kelompok sehingga *events* yang ada di dalam kelompok dapat memenuhi *hard constraints.*
d. Pendekatan berbasis *constraint* yang mana metode ini memodelkan sejumlah *events* yang ingin dijadwalkan sebagai kumpulan variabel yang nilai-nilainya harus ditetapkan sesuai jumlah kendala.
e. Metode metaheuristik yang mana metode ini dimulai dengan memperoleh beberapa solusi awal lalu menggunakan strategi pencarian untuk menghindari terjebak dalam optimum lokal.

f. Pendekatan multikriteria yang mana pada pendekatan ini setiap kriteria menghitung pelanggaran dari suatu kendala yang sesuai. Kriteria-kriteria ini memiliki tingkat kepentingan yang berbeda dalam situasi yang berbeda dan untuk institusi yang berbeda.
g. Pendekatan *case-based reasoning* yang mana digunakan solusi-solusi sebelumnya sebagai bagian untuk membentuk solusi penjadwalan yang baru.

2.2.2. Metode Metaheuristik

Dalam kasus UCTP, salah satu pendekatan untuk penyelesaiannya adalah menggunakan metode metaheuristik. Metode metaheuristik adalah sebuah pendekatan dalam optimasi dan digunakan untuk menyelesaikan permasalahan-permasalahan yang kompleks yang sulit diselesaikan dengan metode eksak atau solusi matematis secara langsung. Menurut Babaei dkk. (2015), pendekatan metaheuristik dalam kasus UCTP memiliki dua macam metode utama, yaitu metode *population based* dan metode *single-solution*.

Metode *population based* adalah metode yang menggunakan populasi dalam mencari solusi. Metode *population based* ini memerlukan populasi awal yang berisi sejumlah solusi terlebih dahulu. Kemudian dilakukan iterasi melalui pendekatan metaheuristik untuk menyeleksi sehingga diperoleh solusi terbaik dari populasi saat ini. Setelah itu, dilakukan beberapa perubahan terhadap solusi-solusi yang tidak terpilih sehingga terjadi perbaikan. Kemudian hasil perubahan ini menggantikan solusi sebelumnya yang tidak terpilih. Prosedur ini berlanjut terus hingga kemudian diperoleh solusi yang optimum meskipun tidak dapat dipastikan bahwa solusi tersebut merupakan solusi optimum global.

Kemudian terdapat beberapa algoritma yang memiliki basis *population based.* Algoritma tersebut terdiri dari algoritma *evolutionary* dan *genetic*, algoritma *ant colony*, algoritma *memetic*, dan algoritma *harmony search*. Algoritma *evolutionary* dan *genetic*. Untuk *Evolutionary Algorithm* (EA) memiliki inspirasi dari mekanisme evolusi pada kehidupan nyata. EA ini bekerja mencari solusi di dalam populasi dengan 3 langkah, yaitu seleksi, regenerasi, dan pergantian. Bagian dari EA yaitu *Genetic Algorithm* (GA) merupakan salah satu algoritma yang populer digunakan dalam penelitian.

Selain EA dan GA, terdapat juga algoritma lain yang termasuk dalam pendekatan metaheuristik berbasis populasi. Algoritma tersebut adalah algoritma *ant colony* (AC). Algoritma ini terinspirasi dari dari perilaku semut yang berusaha mencari rute

antara sarang semut dan sumber makanan. Semut menggunakan feromon sebagai penunjuk jalur dan melakukan eksplorasi berdasarkan jejak feromon yang mereka tinggalkan. Berdasarkan prinsip tersebut, kemudian diimplementasikan ke dalam algoritma dan digunakan untuk mencari jalur terbaik/optimal menuju sumber makanan/tujuan yang ingin dicapai.

Selanjutnya, terdapat juga algoritma *memetic* yang sekilas mirip dengan GA. Akan tetapi, yang membedakan dalam prosesnya adalah terdapat tahap tahap tambahan yang dapat melakukan perbaikan lokal sehingga algoritma ini dapat meningkatkan solusi yang ada melalui pengoptimalan lokal. Menurut (Obit, 2010), algoritma *memetic* adalah kombinasi antara genetika dan *hill climbing* (pencarian lokal).

Lalu, ada juga yang disebut dengan algoritma *harmony search*. Algoritma ini terinspirasi dari proses penciptaan harmoni dalam dunia musik. Prinsipnya adalah menciptakan sebuah harmoni atau solusi terbaik dengan menggabungkan berbagai elemen dari solusi yang telah ada. Adapun langkah-langkah yang digunakan pada algoritma ini, yaitu inisialisasi harmoni dengan membentuk sekumpulan solusi acak, improvisasi dengan menggabungkan elemen-elemen ke dalam solusi, penyaringan untuk menyempurnakan solusi yang dihasilkan, dan *update memory* untuk memperbarui dengan solusi yang lebih baik.

Selain itu metode *population based,* terdapat juga metode *single solution*. Menurut Obit (2010), Metode ini menggunakan solusi tunggal yang terpilih berdasarkan beberapa kriteria. Kemudian selama proses penyelesaian masalah, solusi dimanipulasi dan dipindahkan hingga diperoleh solusi akhir yang lebih baik. Terminasi terjadi ketika kondisi akhir dan semua kriteria penyelesaian telah terpenuhi. Adapun algoritma yang memiliki basis *single solution*, yaitu *Simulated Annealing* (SA) dan *Tabu Search*.

SA terinspirasi dari proses pendinginan logam dalam metalurgi. Prinsipnya adalah mencari solusi terbaik dengan menggunakan simulasi suhu. Awalnya algoritma ini menerima solusi awal dan suhu yang tinggi yang kemudian suhunya menurun seiring iterasi mulai berjalan. Semakin suhu menurun, maka algoritma memiliki peluang yang lebih rendah untuk mendapatkan solusi yang lebih buruk.

Berbeda dengan SA, *tabu search* menggunakan pendekatan berbasis memori. Algoritma ini menggunakan daftar tabu untuk mencatat langkah yang telah dilalui untuk mencegah langkah yang sama terulang kembali. Selain itu, algoritma ini juga

berusaha untuk mencari solusi yang lebih baik di sekitar solusi saat ini dengan mencoba melewati batasan-batasan.

2.2.3. Algoritma Genetika

Algoritma genetika atau *genetic algorithm* (GA) merupakan salah satu pendekatan dari metode metaheuristik yang digunakan untuk meningkatkan kualitas populasi berdasarkan hasil seleksi yang dilakukan. Menurut Goldberg (1989), algoritma ini mengkombinasikan kelangsungan hidup yang terkuat dari berbagai struktur atau entitas dengan melakukan pertukaran informasi yang terstruktur namun juga membawa unsur acak. Berdasarkan pernyataan tersebut, maka dapat dikatakan bahwa algoritma genetika bekerja dengan melakukan kombinasi (mutasi) dan kawin silang (*crossover*) dari hasil sebelumnya yang sudah baik untuk memperoleh hasil yang lebih baik lagi.

Secara teknis, algoritma ini mengenal yang namanya kromosom yang terdiri dari berbagai elemen. Setiap koromosom memiliki kualitasnya tersendiri dan kualitas baik atau buruk dari setiap kromosom dapat diukur menghasilkan yang namanya *fitness value. Fitness value* yang tinggi menandakan bahwa solusi yang diperoleh mendekati solusi optimal yang diinginkan. Akan tetapi, perlu diingat bahwa *fitness value* yang tinggi juga perlu memperhatikan apakah telah memenuhi semua *constraints* atau belum.

Menurut Goldberg (1989), terdapat populasi dalam proses berjalannya algoritma. Populasi yang besar atau kecil dapat mengindikasikan apakah performa dari GA baik atau tidak. Secara umum, populasi yang besar akan semakin baik karena pencarian solusinya juga menjadi lebih luas. Akan tetapi, hal ini akan memicu waktu komputasi yang memiliki durasi yang lama. Namun, apabila populasi diatur terlalu kecil maka akan memicu terjadinya *premature convergence. Premature convergence* adalah peristiwa berhentinya proses pencarian solusi terlalu cepat sebelum solusi yang lebih baik ditemukan.

Untuk prosedur dari algoritma genetika sendiri dimulai dari tahap memperoleh populasi awal terlebih dahulu. Gen pembentuk kromosom akan dimunculkan secara *random* untuk menjadi bagian dari populasi. Cara untuk memunculkan tiap gen pembentuk kromosom dapat dilakukan dengan bantuan dari algoritma lain seperti yang diterapkan pada penelitian ini.

Setelah memperoleh kromosom dalam populasi, maka selanjutnya dilakukan evaluasi terhadap nilai *fitness value*. Setiap kromosom dalam populasi diukur *fitness value*-nya dan dievaluasi berdasarkan kriteria yang ditentukan. Jika sudah terdapat kromosom yang memiliki *fitness value* paling tinggi yang memenuhi batasan atau setelah iterasi mencapai jumlah maksimalnya, maka algoritma berhenti melakukan iterasi.

Jika hasil tidak ada yang memenuhi kriteria atau batasan, maka akan dilakukan beberapa tahapan. Tahap pertama adalah melakukan seleksi dengan menggunakan kromosom yang memiliki *fitness value* tertinggi. Kemudian kromosom terpilih akan dijadikan sebagai *parent* dan jumlahnya minimal 2 *parent* agar dapat dilakukan eksploitasi. Tujuannya adalah agar kromosom tersebut selalu menjadi yang paling baik atau lebih baik di generasi selanjutnya.

Lalu tahap kedua adalah melakukan *crossover* atau kawin silang antar kromosom. Pada tahap ini dilakukan percobaan kombinasi antar kromoson *parent* agar dapat dimunculkan kromosom yang baru hasil kawin silang yang biasa disebut *offspring.* Adanya proses *crossover* ini bertujuan untuk mengeksplorasi kemungkinan-kemungkinan baru yang harapannya memperoleh kromosom baru yang *fitness value*-nya lebih baik. Nantinya, *crossover* ini akan dikontrol melalui nilai *crossover rate.*

Setelah dilakukan *crossover*, selanjutnya dilakukan mutasi terhadap kromosom hasil *crossover*. Mutasi ini merupakan bagian dari eksplorasi dengan cara memindahkan gen yang ada di kromosom itu sendiri. Untuk proses mutasi ini nantinya juga akan dikontrol dengan nilai tertentu. Akan tetapi, proses mutasi mengandung proses *random* sehingga resiko terjadi penurunan *fitness value* bisa saja terjadi. Oleh karena itu, ada pendekatan yang membuat kromosom dengan *fitness value* paling baik tidak ikut dalam mutasi yang dikenal dengan sebutan *elitism.* Selain pendekatan tersebut, terdapat juga cara lain yang dapat melakukan perbaikan jika hasil mutasi memiliki *fitness value* yang lebih rendah dari sebelumnya yang disebut dengan *repair.*

Setelah anak kromosom mengalami *crossover* dan mutasi, maka anak kromosom tersebut dapat masuk menggantikan sebagian ataupun keseluruhan dari kromosom populasi awal. Proses pergantian ini disebut dengan proses regenerasi. Adapun kromosom yang diregenerasi adalah kromosom yang memiliki nilai *fitness value* terkecil yang ada dalam populasi. Dengan adanya regenerasi, maka akan

dilakukan evaluasi kembali. Proses seleksi, *crossover,* mutasi, dan regenerasi akan kembali terjadi jika hasil evaluasi menunjukkan bahwa tidak ada solusi yang memenuhi kriteria atau batasan yang ditentukan.

2.2.4. Bahasa Pemrograman

Menurut Minnick & Holland (2016), program komputer merupakan kumpulan-kumpulan instruksi yang dapat dimengerti dan diikuti oleh sebuah komputer. Program komputer saat ini juga dikenal dengan istilah *software* atau perangkat lunak. Komputer tidak dapat melakukan sesuatu dengan sendirinya dan membutuhkan program untuk mengetahui apa yang dilakukan. Orang yang menuliskan kode agar komputer dapat melakukan sesuatu dikenal dengan *computer programmers.*

Setiap program komputer dibuat dengan menggunakan bahasa pemrograman. Bahasa pemrograman memungkinkan *programmer* untuk menuliskan instruksi yang dapat diterjemahkan menjadi bahasa mesin. Bahasa pemrograman ini kemudian diubah dalam bentuk kode biner yang menggunakan angka nol dan satu untuk membentuk huruf, angka, dan simbol yang disatukan untuk menjalankan tugas.

Bahasa pemrograman adalah bahasa yang berbeda dengan bahasa yang umumnya digunakan oleh manusia. Terdapat beberapa sintaksis yang memiliki struktur sendiri yang membedakan dengan bahasa pada umumnya. Adapun bahasa pemrograman dikategorikan dalam beberapa level, yakni:

a. Bahasa level rendah, yaitu bahasa yang sulit dipahami oleh manusia karena instruksi yang dibuat mendekati bahasa mesin dan spesifik tergantung mesinnya. Bahasa mesin dalam hal ini berbentuk kode-kode mesin sebagai instruksinya dan dapat berinteraksi langsung dengan perangkat keras. Contoh bahasa yang masuk kategori ini adalah bahasa Assembly.
b. Bahasa level menengah, yaitu bahasa yang sudah mulai dapat dipahami oleh manusia. Selain itu, bahasa level ini dapat berinteraksi langsung dengan perangkat keras dengan lebih efisien dalam penggunaan memori dan kecepatan eksekusi. Salah satu bahasa yang masuk ke dalam kategori level menengah ini adalah bahasa C, C++, Rust, dan lain-lain.
c. Bahasa level tinggi, yaitu bahasa yang mendekati bahasa manusia dan umumnya dirancang untuk memudahkan dalam penulisan, pembacaan, dan pemeliharaan. Bahasa pada level ini tidak dapat berinteraksi atau kontak

langsung dengan perangkat keras. Bahasa yang masuk dalam level ini adalah Java, Python, PHP, JavaScript, dan lain-lain.